Employing Best Practices for Improving the Usability of LookML Projects | Google Cloud Skills Boost

Qwiklabs : 29-37 minutes

GSP1020

Overview

Looker is a modern data platform in Google Cloud that you can use to analyze and visualize your data interactively. You can use Looker to analyze data in depth, integrate insights across different data sources, build actionable data-driven workflows, and create custom data applications.

In this lab, you learn the best practices for writing LookML code that enhances the experience of both business users and developers. This includes improving the usability and sustainability of LookML projects by reusing existing objects and applying descriptive naming conventions. You also use LookML parameters to make Explores easier to use by providing additional context and customizing the objects that are visible to business users.

Prerequisites

Familiarity with LookML is necessary. We recommend that you complete the Understanding LookML in Looker quest before you begin this lab.

Objectives

In this lab, you:

- Use descriptive field names to define dimensions and measures.
- Leverage additional parameters to organize and provide more context for your data.
- Create and surface the fewest number of fields and Explores possible.
- Write sustainable and modular LookML code that can easily be updated, maintained, and reused.

Setup and requirements

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).

Note: Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

- Time to complete the lab---remember, once you start, you cannot pause a lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab to avoid extra charges to your account.

How to start your lab and sign in to Looker

1. When ready, click 

.

   A new panel will appear with the temporary credentials that you must use for this lab.

   If you need to pay for the lab, a pop-up will open for you to select your payment method.

2. Note your lab credentials in the left pane. You will use them to sign in to the Looker instance for this lab.

   Note: If you use other credentials, you will get **errors or incur charges**.

3. Click **Open Looker**.

4. Enter the provided Username and Password in the Email and Password fields.

   Important: You must use the credentials from the Connection Details panel on this page. Do not use your Google Cloud Skills Boost credentials. If you have your own Looker account, do not use it for this lab.

5. Click **Log In**.

   After a successful login, you will see the Looker instance for this lab.

Best practices for writing LookML code

This section provides an overview of the top five best practices you can employ to write LookML code that enhances the experience of both business users and developers. For the purposes of this lab, you will focus on the first four items.

Use descriptive field names

Using descriptive field names to define dimensions and measures helps both business users and developers find what they need (#1 in the best practices). Some examples of this include:

- Name measures by their aggregate function or with common terms: `total_[FIELD]` for sum, `count_[FIELD]`, `avg_[FIELD]`, etc.
- Name ratios descriptively: `percent_orders_by_returning_customer` is clearer than `percent_returning`.
- Name **yesno** fields clearly: `order_is_returned` or `is_order_returned`, instead of `returned`.
- Avoid using the words "date" or "time" in a dimension group. Looker already appends each timeframe to the end of the dimension name. For example, `created_date` would become `created_date_date`.

Leverage additional parameters

Leveraging additional parameters can help organize and provide more context for the data exposed to business users. You should include labels and descriptions to help business users identify which fields and Explores to use for their workflows (#1 and #4 in best practices).

You should also use the `group_label` parameter to group similar fields and Explores into logical categories for easier navigation (#2 in best practices).

Finally, you should use `drill_fields` to curate the additional options that a business user sees when they click on the value of a table cell while exploring data.

Create and surface the fewest number of fields and Explores possible

Creating and surfacing the fewest number of fields and Explores possible while still allowing business users to easily access the answers they need is a very beneficial best practice to implement (#3 in best practices). The optimal number of fields and Explores is different for every business, but having too many of each tends to confuse end users.

You can use the `fields` parameter to limit fields surfaced to the business user in an individual Explore and use the `hidden` parameter to hide a field or Explore across the entire model.

Write sustainable and modular LookML code

Writing sustainable and modular LookML code that can easily be updated, maintained, and reused is key to maintaining functional projects.

Although the organization and quality of the underlying LookML code might not be immediately apparent, it can also affect the overall business user experience. Specifically, if you (as a LookML developer) can spend less time writing and maintaining your base code, you can more quickly modify or implement attributes and features that are requested by business users.

Here are a few ideas for writing sustainable and modular LookML code:

- Use substitution operators throughout your code to minimize hard-coded references to your underlying database.

- Identify appropriate cases for SQL-derived versus native-derived tables.
    - Native derived tables promote code reusability by leveraging existing LookML objects in your model to define new structures or aggregations that do not yet exist in your underlying database.
    - SQL derived tables are used for more custom or complex aggregations that are not easily accomplished with native derived tables.
- Use extends or refinements to expand on existing views and Explores.
    - You can extend a view or Explore to make a copy of the original object and apply modifications. Check out the Modularizing LookML Code with Extends lab for more information and practice.
    - You can also create a refinement to adapt an existing view or Explore without editing the LookML file that contains it. Check out the Optimizing Performance of LookML Queries lab for more information and practice

Optimize the performance of Explore queries

To optimize the performance of Explore queries for business users:

- Avoid joining extraneous views in an Explore.
- Declare a primary key in the view file, and define a join relationship using the `relationship` parameter to ensure that correct aggregates are produced.
- Define `many_to_one` joins between views from the most granular level to the highest level of detail.
- Persist derived tables for complex queries that take a long time to run or for queries that are used frequently by a large number of users or applications.

For more information about these topics, see Looker Dos and Don'ts and Optimizing Performance of LookML Queries.

Task 1. Create fields that leverage existing fields and descriptive naming conventions

When creating new dimensions and measures, always review the existing LookML objects in your model for potential reuse. To facilitate code updates, a best practice is to use substitution operators throughout your code to minimize hard-coded references to objects in your underlying database.

In addition to leveraging existing objects, also choose descriptive names when creating new objects, which can help developers and business users find fields for their code or analysis. In this task, you create a new **yesno** dimension to identify canceled orders by leveraging the existing status dimension, create new measures that leverage existing objects to calculate the percentage of revenue lost from canceled orders, and apply descriptive naming conventions to new dimensions and measures.

Create a yesno dimension from an existing dimension

1. Click the toggle button to enter **Development mode**.

2. From the **Develop** menu, select the **qwiklabs_ecommerce** project.

3. Open **order_items.view** under **views**.

4. Review the dimension named **status**.

   Notice that the syntax uses the substitution operator `${TABLE}.column_name`, which references a column in the table identified in the `sql_table_name` parameter at the top of a view file.

   In this case, `${TABLE}.status` references the status column within the `cloud-training-demos.looker_ecomm.order_items` table. You can leverage this existing dimension to create a new **yesno** dimension that identifies whether an order has a canceled status.

Both `order_is_canceled` and `is_order_canceled` adhere to the best practice of descriptive names because they clearly identify the canceled object as an order. The choice between the two options is a naming convention decision that could be decided within your team.

5. Under the `status` dimension, add the following code for a new yesno dimension:

dimension: order_is_canceled { type: yesno sql: ${status} = "" ;; }

Notice that the syntax uses the substitution operator `${field_name}`, which references an existing *dimension or measure* within the current view. In this case, `${status}` references the `status` dimension within the **order_items** view.

Also notice that this code does not yet include a value for status. Although you can leverage the existing status dimension, you still need to know which value in the dimension is used to identify canceled orders.

Technically, all of these options could work to identify the correct value. However, the educated guess option is inefficient and could potentially result in an incorrect value; you would have to keep testing your code by running a query on the new dimension in the Explore until you receive the expected result.

Running a query on the status dimension in the Order Items View of the Order Items Explore would yield its dimension values, but the values displayed in the Explore could potentially reflect additional formatting applied with LookML code.

Running a query on the `order_items` table *directly* in SQL Runner allows you to see the raw data values in the status column.

6. Leave the browser tab for the IDE open, and open a new Looker window in a new browser tab.

7. Navigate to **Develop** > **SQL Runner**.

8. Next to connections click **Settings** (⚙), and then click **Search public projects**.

9. For **Project**, type **cloud-training-demos**, and then press ENTER.

10. For **Dataset**, select **looker_ecomm**.
    A list of the available tables in this BigQuery dataset is displayed.

11. To select the distinct values in the `status` column, add the following query to the SQL Query window, and then click **Run**:

SELECT distinct(status) FROM cloud-training-demos.looker_ecomm.order_items ORDER BY status **Note:** Notice that the values include *Cancelled*, which uses the British English spelling with two letter Ls, instead of *Canceled* in the American English spelling.

12. Close the browser tab for SQL Runner, and return to the browser tab for the IDE.

13. Complete the LookML code for the new **yesno** dimension:

dimension: order_is_canceled { type: yesno sql: ${status} = "Cancelled" ;; }

14. Click **Save Changes**.
    Leave this browser tab open for the Looker IDE.

15. Open a new Looker window in a browser tab.

16. Navigate to **Explore > Order Items**.

17. In the Data pane, click on the **SQL** tab.

18. Under **Order Items > Dimensions**, select:

    - **Order ID**
    - **Order Is Canceled (Yes/No)**

Before running the query, notice that the `CASE` statement returns the *Yes or No* result, depending on whether the value of `order_item.status` is equal to **Cancelled**:

CASE WHEN order_items.status = "Cancelled" THEN 'Yes' ELSE 'No' END

19. Click **Run**.

20. Open the **Results** tab to see the results.

21. Close the browser tab for the Explore, and return to the browser tab for the IDE.

Create new measures based on an existing dimensions and measures

1. Open **order_items.view**.

2. Review the measure called `total_revenue_from_completed_orders`.

   Notice that the name follows best practices for sum measure because it begins with “total” to clearly indicate that the value is a *summed* or *total value*. Also, notice that it contains a filter on the existing status dimension where the value is equal to *Complete*.

Although `total_canceled_orders` begins with *total*, it does not clearly identify that the sum is the total of the revenue, not the total of the orders. In contrast, `revenue_from_canceled_orders` does include descriptive details but is missing *total* at the beginning of the name.

3. Following this measure, add the following code to create two new measures:

measure: total_revenue_from_canceled_orders { type: sum sql: ${sale_price} ;; filters: [order_is_canceled: "Yes"] value_format_name: usd } measure: percent_revenue_canceled_orders { type: number value_format_name: percent_2 sql: 1.0*${total_revenue_from_canceled_orders} /NULLIF(${total_revenue}, 0) ;; }

The first measure named `total_revenue_from_canceled_orders` follows the same best practice for naming sum measures by beginning with the word *total*, and it clearly identifies that the sum is for revenue from canceled orders.

However, notice that the filter is different. Instead of using the **status** dimension, `total_revenue_from_canceled_orders` uses the **yesno** dimension that you created in the previous steps. Both options follow best practices of reusing existing LookML objects, depending on which objects are available in your model.

The second measure follows the best practice of naming ratios descriptively by including the words *percent* and *canceled orders*. The measure also leverages two existing measures to calculate the percent revenue attributed to canceled orders: the first measure and the `total_revenue` measure.

4. Click **Save Changes**.
   Leave this browser tab open for the Looker IDE.
5. Open a new Looker window in a browser tab.
6. Navigate to **Explore > Order Items**.
7. In the Data pane, click on the **SQL** tab.
8. Under **Order Items > Measures**, select:
   - **Total Revenue From Canceled Orders**
   - **Total Revenue**
   - **Percent Revenue Canceled Orders**

Before running the query, notice the **CASE** statement now used in conjunction with **SUM** to calculate the total of `order_items.sale_price` when the `order_items.status` is equal to *Cancelled*, and then divided by the total of all values in `order_items.sale_price`:

SUM(CASE WHEN order_items.status = "Cancelled" THEN order_items.sale_price ELSE NULL END), 0) / NULLIF(COALESCE(SUM(order_items.sale_price), 0), 0)

9. Click **Run**.
10. Open the **Results** tab to see the results.
11. Close the browser tab for the **Explore** query, and return to the browser tab with the Looker IDE.
12. Click **Project Health** ( ).

13. In the *Project Health* > *LookML validation* section, click **Validate LookML**.

No LookML errors should be found.

Commit changes and deploy to production

1. Click **Validate LookML** and then click **Commit Changes & Push**.
2. Add a commit message and click **Commit**.
3. Lastly, click **Deploy to Production**.

Remain in the browser tab for the Looker IDE as you continue to the next task.

Click *Check my progress* to verify the objective. Create fields that leverage existing fields and descriptive naming conventions

Task 2. Provide context to fields and Explores with labels and descriptions

Adding labels and descriptions to LookML objects is an easy way to help business users identify which fields and Explores to use for their workflows. In this task, you add labels and descriptions to the new **yesno** dimension and measures created in the previous task. You also add a label and description to the existing Order Items Explore that exposes these fields to business users.

Add labels and descriptions to dimensions

1. Open a new Looker window in a new browser tab.
2. Navigate to **Explore > Order Items**.
3. Expand **Order Items**, and hold the pointer over the dimension called **Order Is Canceled (Yes/No)** to see additional options.
4. Click **Info** (ⓘ) to see the details of this dimension.

   The **Info** control provides details such as the SQL parameter, but for business users with limited SQL knowledge, there is no additional description to help them easily understand the intended use of the dimension. In the next steps, you add the description parameter to the **Order Is Canceled** dimension to provide additional context for business users.
5. Leave the browser tab for the Explore open, and return to the browser tab for the Looker IDE.
6. Open **order_items.view**.
7. Add a description for the `order_is_canceled` dimension you created earlier:

dimension: order_is_canceled { description: "A value equal to Yes means that the order has a canceled status. A value equal to No means that the order does not have a canceled status." type: yesno sql: ${status} = "Cancelled" ;; }

8. Click **Save Changes**, and then click **Validate LookML**.
9. Return to the browser tab for the Order Items Explore, and refresh the page.
10. Expand **Order Items**, and hold the pointer over the dimension called **Order Is Canceled (Yes/No)** to see additional options.
11. Click **Info** (ⓘ) to see the details of this dimension.

    A clear description now explains to business users how to interpret the values in the dimension.
12. Leave the browser tab for the Explore open, and return to the browser tab for the Looker IDE.
13. Review the LookML code for `order_is_canceled` dimension again.

    Notice that you did not include the label parameter because it was not needed. The dimension name was already clearly displayed in the Explore as “Order Is Canceled (Yes/No)”.

    In the next steps, you add both a description and a label to the measures you created in the previous task so that you can modify how the measure names are displayed in the Explore and include additional details about how to interpret the measure values.

Add labels and descriptions to measures

1. In **order_items.view**, add descriptions and labels to the measures called `total_revenue_from_canceled_orders` and `percent_revenue_canceled_orders`:

measure: total_revenue_from_canceled_orders { label: "Total Revenue Lost From Canceled Orders" description: "Sum of sale price for orders with canceled status." type: sum sql: ${sale_price} ;; filters: [order_is_canceled: "Yes"] value_format_name: usd } measure: percent_revenue_canceled_orders { label: "% Revenue Lost From Canceled Orders" description: "Total revenue lost from canceled orders as a percentage of the total revenue from all orders." type: number value_format_name: percent_2 sql: 1.0*${total_revenue_from_canceled_orders} /NULLIF(${total_revenue}, 0) ;; }

2. Click **Save Changes**, and then click **Validate LookML**.

   Adding labels to these measures helps emphasize that canceled orders represent lost revenue, and the descriptions provide additional context for how to interpret the measure type and SQL parameter.
3. Return to the browser tab for the Order Items Explore, and refresh the page.
4. Expand **Order Items**, and notice the revised labels for the two measures: *% Revenue Lost From Canceled Orders* and *Total Revenue Lost From Canceled Orders*.
5. Click **Info** (ⓘ) for each measure to see the details.

In addition to the labels, useful descriptions now explain to business users how to interpret the values in these measures.

6. Under **Order Items > Dimensions**, select: **Order Is Canceled (Yes/No)**.
7. Under **Order Items > Measures**, select:
    - **Total Revenue Lost From Canceled Orders**
    - **Total Revenue**
    - **% of Revenue Lost from Canceled Orders**
8. In the Data pane, hover over the column names to see the same description provided for the Info button for **Order Is Canceled**, **Total Revenue Lost From Canceled Orders**, and **% of Revenue Lost from Canceled Orders**.
9. Click **Run**.

    The query returns two rows based on the **Order Is Canceled** dimension: one row for *Yes* and one row for *No*. Business users who have any questions about these results can now refer to the descriptions of these dimensions and measures for more context.

Add a label and description to an Explore

1. Close the browser tab for the Explore, and return to the browser tab for the Looker IDE.
2. Open **training_ecommerce.model** under **models**.
3. Add a label and description to the existing Order Items Explore, before the join for `users`:

```
explore: order_items {
  label: "Orders and Users"
  description: "Use this Explore to review details for orders and users, including information on inventory, products, and distribution centers."
  join: users {
    type: left_outer
    sql_on: ${order_items.user_id} = ${users.id} ;;
    relationship: many_to_one
  }
```

4. Click **Save Changes**, and then click **Validate LookML**.
5. Leave the browser tab for the IDE open, and open a new Looker window in a new browser tab.
6. Click **Explore** to see the menu list of Explores.

The Order Items Explore has a new label of **Orders and Users**. Also notice that there is now an **Info** control next to the Explore name.

7. Hold the pointer over **Info** (ⓘ) to see the details of the Explore.

    A clear description with details about the data available in the Explore is now available so that business users can easily determine whether this Explore is appropriate for their workflows.

8. Close the browser tab for the Explore, and leave the browser tab for the IDE open.

Commit changes and deploy to production

1. Click **Validate LookML** and then click **Commit Changes & Push**.
2. Add a commit message and click **Commit**.
3. Lastly, click **Deploy to Production**.

Click *Check my progress* to verify the objective. Provide context to fields and Explores with labels and descriptions

Task 3. Limit fields to only those needed in a specific Explore

When creating new LookML objects in your model, be mindful of which fields and Explores should be directly accessed by business users. Surfacing only a specifically curated number of fields and Explores allows business users to quickly find the data they need.

In this task, you use the `hidden` parameter to hide extra fields across the entire training_ecommerce model and leverage the `fields` parameter to limit fields surfaced to business users in the existing Order Items Explore.

Hide a field from all Explores in a LookML model

1. Open a new Looker window in a new browser tab.
2. Navigate to **Explore > Events**.
3. Expand **Users**.
4. Open another new Looker window in a new browser tab.
5. Navigate to **Explore > Orders and Users**.
6. Expand **Users**, and compare the two Explores.

Notice that you see the same dimensions for Users in the Events Explore as in the Orders and Users, including the *Latitude* and *Longitude* dimensions, which you may not always want to expose to business users.

Hint: you can do a search for each parameter in the Looker documentation.

7. Leave the browser tabs open for both Explores, and return to the Looker IDE.
8. Open **users.view** under **views**.
9. Add the `hidden` parameter to the latitude and longitude dimensions:

dimension: latitude { hidden: yes type: number sql: ${TABLE}.latitude ;; } dimension: longitude { hidden: yes type: number sql: ${TABLE}.longitude ;; }

10. Click **Save Changes**, and then click **Validate LookML**.

11. Return to the browser tab for the **Orders and Users** Explore, and refresh the page.
12. Expand **Users**, and review the available dimensions.

    Notice that the *latitude* and *longitude* dimensions are no longer visible in the Orders and Users Explore.
13. Return to the browser tab for the **Events** Explore, and refresh the page.
14. Expand **Users**, and review the available dimensions.

    The *latitude* and *longitude* dimensions are no longer visible in either Explore. However, imagine that instead of hiding a dimension or measure from all Explores, you want to hide it from only specific Explores. For example, the Events Explore contains all of the same user information as the Orders and Users Explore, but many of these fields provide more personally identifiable information than may actually be needed in the Events Explore.

    In the next steps, you hide most of the dimensions and measures in the Users view but only in the Events Explore, where only minimal information is needed to select users or to identify general trends.

Selectively hide fields from specific Explores

1. Leave the browser tabs open for the Explore, and return to the browser tab for the IDE.
2. Open **training_ecommerce.model**.
3. Add the `fields` parameter to the existing **Events** Explore, before the *join* for `event_session_facts`:

```
explore: events { fields: [ALL_FIELDS*, -users.city, -users.email, -users.first_name, -users.gender, -users.last_name, -users.state] join: event_session_facts { type: left_outer sql_on: ${events.session_id} = ${event_session_facts.session_id} ;; relationship: many_to_one }
```

The syntax for the `fields` parameter indicates that all fields will remain visible in the Explore except for those identified with a minus sign (“`-`”) before the field name, such as `users.city`, which will be hidden in the Explore.

4. Click **Save Changes**, and then click **Validate LookML**.
5. Return to the browser tab for the **Orders and Users** Explore, and refresh the page.
6. Expand **Users**, and review the available dimensions.

    Notice that no additional dimensions have been hidden in the Orders and Users Explore.
7. Return to the browser tab for the **Events** Explore, and refresh the page.
8. Expand **Users**, and review the available dimensions.

    Notice that only a few dimensions for Users remain available in the Events Explore, which are those that were not explicitly identified with a minus sign (“`-`”) in the fields parameter.

The Events Explore now supports more general analyses of users and events, and the Orders and Users Explore contains more specific user details, such as personally identifiable information.

9. Under **Users > Dimensions**, select **Country**.
10. Under **Users > Measures**, select **Count**.
11. Under **Events > Dimensions**, select **Event Type**.
12. Click **Run**.

    The query returns the number of users per country for each event type, including users with null country values.

13. Close both of the browser tabs open for the Explore, and leave the browser tab for the IDE open.

Commit changes and deploy to production

1. Click **Validate LookML** and then click **Commit Changes & Push**.
2. Add a commit message and click **Commit**.
3. Lastly, click **Deploy to Production**.

Click *Check my progress* to verify the objective. Limit fields to only those needed in a specific Explore

Task 4. Group similar fields or Explores into useful categories

As a LookML developer, you can use the `group_label` parameter to facilitate navigation of Explores by grouping similar fields or Explores into logical categories. In this task, you group the various location dimensions in **users.view** and create separate groups for the Orders and Users Explore and Events Explore in **training_ecommerce.model** under different headings labeled by team.

Group similar fields in a view

1. Open **users.view** and review the available dimensions.

Notice that several dimensions contain location information such as city, country, state, and ZIP.

2. Add the `group_label` parameter to the **city**, **country**, **state**, and **zip** dimensions:

dimension: city { group_label: "Location" type: string sql: ${TABLE}.city ;; } dimension: country { group_label: "Location" type: string map_layer_name: countries sql: ${TABLE}.country ;; } dimension: state { group_label: "Location" type: string sql: ${TABLE}.state ;; map_layer_name: us_states } dimension: zip { group_label: "Location" type: zipcode sql: ${TABLE}.zip ;; }

3. Click **Save Changes**, and then click **Validate LookML**.
4. Leave the browser tab open for the IDE, and open a new Looker window in a new browser tab.

5. Navigate to **Explore > Orders and Users**.

6. Expand **Users** and review the available dimensions.

7. Under **Users > Dimension > Location**, select:

    - **City**
    - **Country**
    - **State**
    - **Zip**

8. Click **Run** to see the results.

9. Close the browser tab for the **Orders and Users** Explore, and return to the browser tab for the IDE.

Create groups of Explores under different headings

1. Open **training_ecommerce.model**.

Notice that a label parameter at the model level on line 15 has a value of “E-Commerce Training”. This is the current heading displayed in the Explore menu. Both the Orders and Users Explore and Event Explore are organized under this heading in the Explore menu.

2. Before the label parameter, add a `group_label` called “E-commerce - Inventory Team” to the **Order Items** Explore:

explore: order_items { group_label: "E-commerce - Inventory Team" label: "Orders and Users"

3. Before the fields parameter, add a `group_label` called “E-commerce - Marketing Team” to the **Events** Explore:

explore: events { group_label: "E-commerce - Marketing Team" fields: [ALL_FIELDS*, -users.city, -users.email, -users.first_name, -users.gender, -users.last_name, -users.state]

4. Click **Save Changes**, and then click **Validate LookML**.

5. Open a new Looker window in a new browser tab.

6. Expand the **Explore** menu and review the options.

Notice that each Explore is now under its own heading. When adding new Explores, you can use the same `group_label` value to continue expanding the existing groups, or assign new `group_label` values to create new groups with their own headings.

Commit changes and deploy to production

1. Click **Validate LookML** and then click **Commit Changes & Push**.

2. Add a commit message and click **Commit**.

3. Lastly, click **Deploy to Production**.

Click *Check my progress* to verify the objective. Group similar fields or Explores into useful categories

Congratulations!

In this lab, you improved the usability and sustainability of LookML projects by reusing existing objects, applying descriptive naming conventions, and using additional parameters such as `fields` and `hidden`. You then limited fields to only those needed in a specific Explore and used the `group_label` parameter to facilitate navigation of Explores by grouping similar fields and Explores into logical categories.

Finish your quest

This self-paced lab is part of the Manage Data Models in Looker quest. A quest is a series of related labs that form a learning path. Completing this quest earns you a badge to recognize your achievement. You can make your badge or badges public and link to them in your online resume or social media account. Enroll in any quest that contains this lab and get immediate completion credit. See the Google Cloud Skills Boost catalog to see all available quests.

Next steps/Learn more

- LookML quick reference
- LookML terms and concepts
- Looker Community
- Additional LookML basics
- Best Practice: Create a Positive Experience for Looker Users

Google Cloud training and certification

...helps you make the most of Google Cloud technologies. Our classes include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. Certifications help you validate and prove your skill and expertise in Google Cloud technologies.

Manual last updated August 24, 2023

Lab last tested August 24, 2023