www.cloudskillsboost.google /games/4423/labs/28618

Troubleshooting Data Models in Looker | Google Cloud Skills Boost

Qwiklabs : 20-26 minutes

GSP1019

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 how to troubleshoot and diagnose your LookML code by using SQL Runner, LookML Validator, Explores, and Content Validator.

What you'll learn

In this lab, you learn how to:

- Use SQL Runner to explore data tables and troubleshoot SQL queries.
- Use the LookML Validator to validate syntax for defined objects and relationships.
- Diagnose and resolve error messages from Explore queries.
- Diagnose and resolve error messages from running the Content Validator.

Prerequisites

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

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.

Tools for troubleshooting LookML code and common use cases

In this section, you learn about the different tools and methods you can use to troubleshoot your LookML code and common use cases for each of them.

Note: This lab simulates errors in syntax and definitions of LookML objects throughout each task to help you learn how to troubleshoot errors that you may encounter in your Looker instance. Be sure to complete the *entire* lab to see the correct syntax and fully resolve errors before pushing your changes to production.

LookML Validator

The LookML Validator is used to perform a full model validation. Some errors, such as an invalid field reference due to a missing join, require a holistic look at the model and therefore are only surfaced when the LookML Validator is run. The LookML Validator checks all of the LookML code in a model, such as the syntax of object definitions (for example, dimensions and measures) and defined relationships (for example, joins). However, it does not check the SQL parameters of LookML objects (for example, SQL derived tables).

Running queries in the Explore

After defining new LookML objects, you can run queries in the Explore. This helps troubleshoot your LookML code because it displays SQL errors provided by the underlying database (for example, inadequate permissions, incorrect SQL object references, or invalid aggregations).

SQL Runner

SQL Runner provides a way to directly access your database and is also a useful tool for checking SQL errors in queries. This is where you can test the custom SQL that you want to include within SQL parameters of LookML objects. You can also see a list of database tables, run ad hoc queries,

write queries for SQL derived tables, etc.

Content Validator

The Content Validator validates all references that your Looks and dashboards make to your LookML models, Explores, views, and fields, and displays an error for any references your content makes to an unknown LookML object. It also checks the Looks and dashboards that were created in the instance to ensure that their references to LookML objects are valid (for example, the name of a specific dimension or measure the Explore might have changed over time).

Looker error catalog

Are you still not sure about where you might see a particular error? The Looker error catalog provides a list of common error messages, their underlying causes, and where in Looker the message is displayed.

Task 1. Use SQL Runner to explore available data and troubleshoot SQL queries

SQL Runner provides direct access to the underlying tables in your database connection. Within SQL Runner, the data tables and columns that are available are listed, and you can run custom SQL queries on your data. In this task, you create a new SQL derived table in SQL Runner after exploring available data and troubleshooting an SQL query.

Connect to a BigQuery dataset in SQL Runner

1. Click the toggle button to enter **Development mode**.
2. On the **Develop** tab, select **SQL Runner**.
3. Click **Settings** (⚙), and then click **Search public projects**.
   The box for Project is now empty.
4. Type **cloud-training-demos**, and press ENTER.
5. For Dataset, select **looker_ecomm**.
   A list of the available tables in this BigQuery dataset is displayed.
6. Add the following query to the SQL Query window:

SELECT orders.user_id as user_id ,COUNT(*) as lifetime_orders ,SUM(orders.order_price) as lifetime_sales FROM cloud-training-demos.looker_ecomm.orders GROUP BY user_id LIMIT 10 **Note:** The provided SQL query contains incorrect information, which you troubleshoot in the next steps.

7. Click **Run**.
   The following error message is displayed:

Query execution failed: - Not found: Table cloud-training-demos:looker_ecomm.orders was not found in location US

Identify the correct table names for SQL queries

1. In the list of table names under **Tables**, select the table that contains order information.

2. Update the table name in the SQL query:

SELECT order_items.user_id as user_id ,COUNT(*) as lifetime_orders ,SUM(order_items.order_price) as lifetime_sales FROM cloud-training-demos.looker_ecomm.order_items GROUP BY user_id LIMIT 10

3. Click **Run**.
   A new error message is displayed:

Query execution failed: - Name order_price not found inside order_items at [5:19]

4. Under **Tables**, click **order_items**.
   A list of the table columns is displayed.

5. Update the column name in the SQL query:

SELECT order_items.user_id as user_id ,COUNT(*) as lifetime_orders ,SUM(order_items.sale_price) as lifetime_sales FROM cloud-training-demos.looker_ecomm.order_items GROUP BY user_id LIMIT 10

6. Click **Run**.
   The query results are returned successfully. You can now save this query as a SQL derived table.

Save the query as a SQL derived table

1. Click **Settings** (⚙) next to **Run**, and then click **Add to Project**.

2. For Project, select **qwiklabs-ecommerce**.

3. For View Name, type **user_order_lifetime**.

4. Click **Add**.

5. In the File Browser, drag **user_order_lifetime.view** to the **views** folder.

6. In the **user_order_lifetime.view** file, delete the code line for `LIMIT 10` from the sql parameter.

Note: You are not defining a `primary_key` for the view at this time.

7. Click **Save Changes**, and then click **Validate LookML**.
   No LookML errors were found, and your file should resemble the following:

8. Navigate to the **training_ecommerce.model** file.

9. In the `explore: events` definition, on a new line before `join: event_session_facts`, use the following code to define a new join:

join: user_order_lifetime { type: left_outer sql_on: ${events.user_id} = ${user_order_lifetime.user_id};; relationship: many_to_one }

10. Click **Save Changes**, and then click **Validate LookML**. There are no LookML errors.

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. Use SQL Runner to explore available data and troubleshoot SQL queries

Task 2. Use the LookML Validator to test syntax and validate relationships defined in the model

The LookML Validator helps you test the syntax for defined objects (for example, dimensions and measures) and validate the relationships (for example, joins) in the model. In this task, you create a new dimension with incorrect LookML syntax that references another view that has not been joined to the base view of the Explore. You then use the LookML Validator to identify and correct the two issues.

Create a new dimension by referencing dimensions in another view

1. In the **qwiklabs-ecommerce** project, open **users.view**.

2. Find the last dimension, and add the following code (around line 88) to create a new dimension:

dimension: average_sales { type: ${number} sql: user_order_lifetime.lifetime_sales / user_order_lifetime.lifetime_orders ;; value_format_name: usd }

Notice that the new dimension references dimensions from the newly created view called `user_order_lifetime`.

Note: This LookML code contains incorrect syntax, which you troubleshoot in the next steps.

3. Click **Save Changes**, and then click **Validate LookML**.
   Four different errors are displayed:

- The first item identifies the location of the syntax error as *“Invalid LookML syntax near line 89”*.
- The last item identifies the specific syntax error: *“Expecting ‘keyword’, ‘}’, got ‘identifier’*

Because the view now contains invalid syntax, it is no longer considered a valid view; thus, additional errors are identified within the model file because the view cannot be found:

- *“Join name must match a view name”* and *“Could not find a field named users.id”*

After the syntax for the new dimension in the view file is corrected, these model file errors are also resolved because the view is again valid.

Identify and correct syntax errors in new dimensions

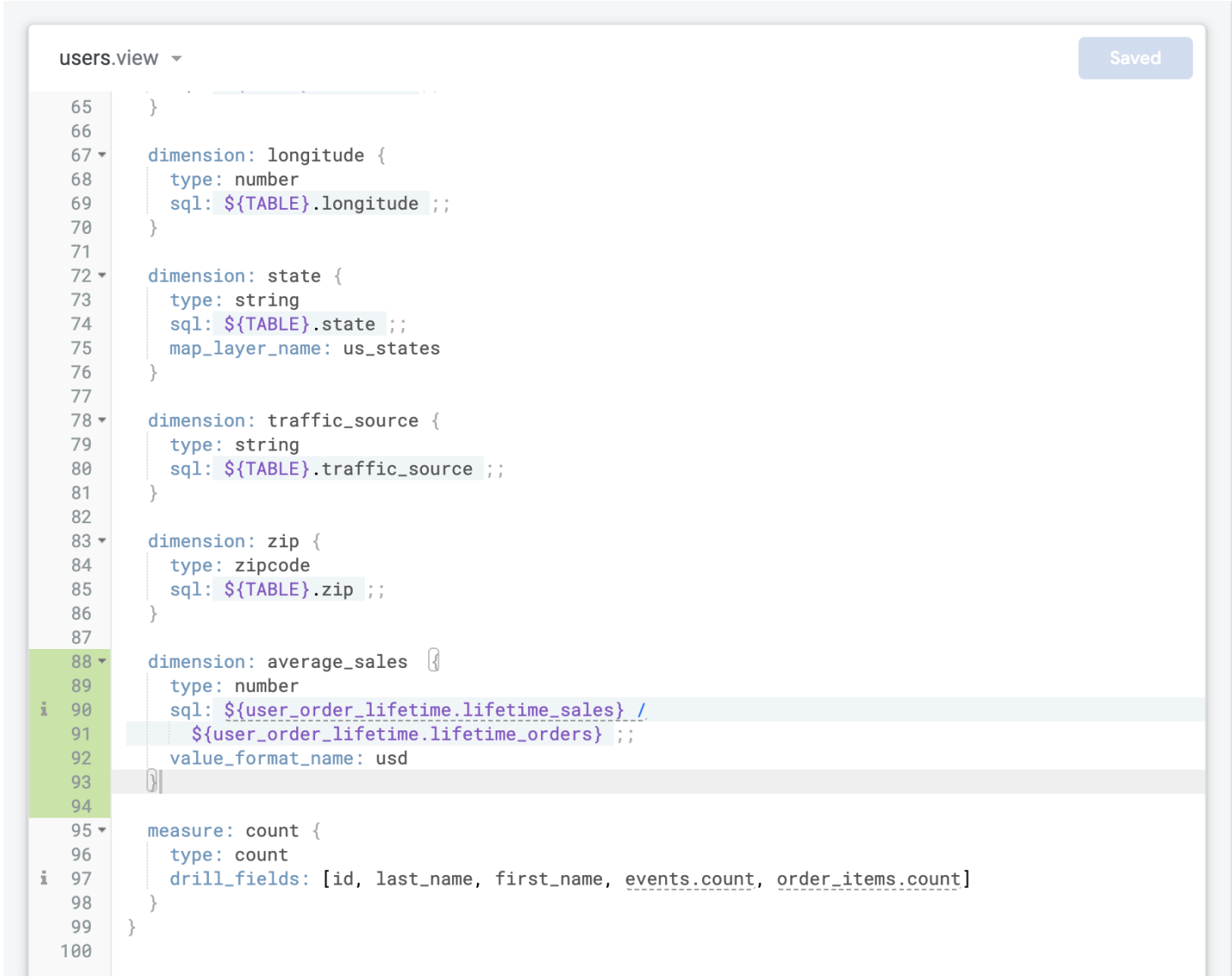
You now troubleshoot the errors in the LookML code.

1. Review the other dimensions in the **users.view** file.
2. Review the documentation on Dimension, filter and parameter types.
3. Update the LookML code for the dimension:

dimension: average_sales { type: number sql: ${user_order_lifetime.lifetime_sales} / ${user_order_lifetime.lifetime_orders} ;; value_format_name: usd }

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

   Because the syntax was updated, the view is valid again, and the model errors have also been resolved. However, there is now new error:



5. Review the error message for **Inaccessible view** in the Looker error catalog.
   There are a few possible options to investigate:
   - The view doesn't exist.
   - The view is not joined correctly to the explore.
6. Review the list of views in the File Browser. Notice that the view called **user_order_lifetime** actually exists in the File Browser.

7. Open and review **training_ecommerce.model**.

   Notice that the new view called **user_order_lifetime** is not joined to the base views of the Explores in the model file. Also notice that the **users.view** is joined to both the **order_items** and **event** Explores. For this reason, the new view for **user_order_lifetime** must also be joined to both Explores in order for the new dimension to be defined successfully within **users.view**.

8. In the `explore: order_items` definition, on a new line before `join: users`, use the following code to define a new join:

join: user_order_lifetime { type: left_outer sql: ${order_items.user_id} = ${user_order_lifetime.user_id};; relationship: many_to_one } **Note:** One of these joins is incorrectly defined, which you troubleshoot in the next section.

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

Leave the browser tab for the IDE open as you begin the next task.

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. Create a new dimension

Task 3. Use the Explore query window to diagnose missing objects and error messages

An easy way to test your changes of LookML code is to run a query in the Explore so that you can see how business users will view and interact with the modified code. By running Explore queries, you can identify missing or invalid LookML objects (for example, a missing `primary_key`) and see SQL errors provided by the underlying database, such as inadequate permissions or incorrect SQL object references (for example, an incorrectly defined join).

In this task, you run queries in the Explore to identify incorrectly defined joins and missing measures. You also correct the LookML for the newly defined joins and add a `primary_key` to the new view to ensure that aggregations (that is, measures) are successful.

Review a new view within the Explore

1. Open a new Looker window in a new tab.
2. Navigate to **Explore > Order Items**.
3. Expand the view for **User Order Lifetime**.

Three dimensions are displayed, **lifetime_orders**, **lifetime_sales**, **user_id**, but no measures.

Leave this browser tab open as you continue to the next steps.

Identify and correct missing parameters for aggregations within a view

1. Return to the browser tab for the Looker IDE, and navigate back to **user_order_lifetime.view**. One measure called `count` should be displayed in the Explore.
2. Review the documentation on requirements for symmetric aggregates. The first requirement is that all views involved in the join need to have a `primary_key` defined.
3. Review **user_order_lifetime.view** again.
   There is no `primary_key` defined.
4. Within **user_order_lifetime.view**, update the `user_id` dimension to define it as the `primary_key` for the view:

dimension: user_id { primary_key: yes type: number sql: ${TABLE}.user_id ;; }

5. Click **Save Changes**, and then click **Validate LookML**.
6. Return to the browser tab for the Order Items Explore, and refresh the page.
7. Expand the view for **User Order Lifetime**.
   After you define a `primary_key` for the view, the Count measure is now displayed.

In the next steps, you remain in the Order Items Explore to test the new dimension defined in **users.view** (Average Sales) that relies on the dimensions in **user_order_lifetime.view**.

Run Explore queries to test a new dimension

1. Under **Users > Dimensions**, click **Average Sales**, and then click **Run**.

An error message is displayed along with the SQL query that the Explore sent to the underlying database. The error message identifies the problem at line 13: *Query execution failed: - Syntax error: Expected end of input but got identifier "order_items" at [13:1]*.

2. In the **Data** pane, open the **SQL** tab to more easily review the failed query, and review line 13:

order_items.user_id =user_order_lifetime.user_id

Although not much information is given about the error, remember that you also joined this new view to Events Explore. In the next steps, you run the same query in the Events Explore to test the view in that Explore.

3. Leave this browser tab for the **Order Items** Explore open, and open a new Looker window in a new tab.

4. Navigate to **Explore > Events**.

5. Under **Users > Dimensions**, click **Average Sales**, and then click **Run**.

6. In the Data pane, open the **SQL** tab to see the successful query.

7. Review line 13 in this query.

   Unlike the query in the **Order Items** Explore, the query syntax at line 13 specifies a **join** between `events` and `user_order_lifetime`:

LEFT JOIN user_order_lifetime ON events.user_id = user_order_lifetime.user_id

Identify and correct invalid parameters in a model

1. Return to the browser tab for the Looker IDE, and open **training_ecommerce.model**.

2. Review the join for **user_order_lifetime** in both the **order_items** Explore and the **events** Explore.

3. Review parameters by referring to LookML parameter reference by function.

4. In the `explore: order_items` definition, update the join for `user_order_lifetime`:

join: user_order_lifetime { type: left_outer sql_on: ${order_items.user_id} = ${user_order_lifetime.user_id};; relationship: many_to_one }

5. Click **Save Changes**, and then click **Validate LookML**.
   There are no LookML errors.

6. Return to the browser tab for the **Order Items** Explore, and refresh the page.

Now that you have correctly defined the join for `user_order_lifetime` within the **order_items** explore, the query runs successfully.

Save an Explore query as a Look

1. Click once on the column for **Average Sales** to sort in *descending* order.

2. Expand the query by clicking on additional dimensions: **ID**, **State**, **Country**, and **Age**.

3. For **Row Limit**, enter: **10**.

4. Click **Run**.

5. Expand the Visualization pane, and select the **Table** visualization.

6. Click **Settings** (⚙).

7. Click **Save > As a Look**.

8. Name the Look `Top 10 Users With Highest Average Sales`.

9. Click **Save & View Look.**
   Your visualization should resemble the following:

	Average Sales	ID	State	Country	Age
1	$999.00	88,111	Louisiana	USA	54
2	$903.00	69,736	Iowa	USA	16
3	$903.00	148,226	Shropshire	UK	24
4	$903.00	151,555	Oklahoma	USA	16
5	$903.00	128,086	California	USA	25
6	$903.00	23,020	New York	USA	19
7	$903.00	150,197	California	USA	16
8	$903.00	150,140	Virginia	USA	16
9	$903.00	143,173	Hackney	UK	41
10	$903.00	74,489	Oklahoma	USA	16

10. Close the other tab for the Explore, and leave this browser tab open as you begin the next task.

Click **Check my progress** to verify the objective. Create a look

Task 4. Use the Content Validator to test and update content after changes to LookML objects

The Content Validator helps you check the Looks and dashboards that were created in the instance to ensure that their references to LookML objects are valid. This is especially helpful if the names of dimensions, measures, views, Explores, or models have been modified. Review the documentation, Before using the Content Validator for more information on how the tool may affect content and objects in your instance.

In this task, you modify the name of a LookML object (for example, a dimension) to be more user friendly, and then use the Content Validator to validate and update the references to the LookML object within existing content (for example, a Look).

Modify the name of existing dimensions

1. Open **users.view**, find the dimension called `average_sales`, and modify the name of the dimension to be more specific for business users:

dimension: average_order_price { type: number sql: ${user_order_lifetime.lifetime_sales} / ${user_order_lifetime.lifetime_orders} ;;
value_format_name: usd }

2. Click **Save Changes**, and then click **Validate LookML**.
   There are no LookML errors.

3. Leave this browser tab open for the IDE, return to the browser tab for the Look, and refresh the page.

Notice that there is now a warning: *'users.average_sales' no longer exists on Order Items, or you do not have access to it, and it will be ignored*.

4. Open a new Looker window in a new tab.

5. Navigate to **Develop > Content Validator**.

6. Click **Validate**.

The Error tab is active, and there is an error for "`Unknown field "users.average_sales"` for the Look called **Top 10 Users With Highest Average Sales**, which you created in the previous task.

7. Click **Find & Replace in All Content**.

8. For **Type**, select **Field**.

9. For **Field Name**, type **users.average_sales**.

10. For **Replacement Field Name**, type **users.average_order_price**.

11. Click **Replace Field Name**.

12. Click **OK**.

13. Click **Validate**.
    The Error tab is now empty because the name of the dimension has been updated in all the content that referenced it (in this case, the Look named **Top 10 Users With Highest Average Sales**).

14. Return to the browser tab for the Look, and refresh the page.
    The Look has been updated and is rendering the visualization successfully, so you can now push your LookML changes to production.

15. Return to the browser tab for the IDE.

16. Click **Validate LookML**. There should be no LookML errors.

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. Modify the name of existing dimensions

Congratulations!

In this lab, you first used the SQL Runner to troubleshoot problematic SQL queries and used the LookML Validator to validate syntax for defined objects and relationships. You then used the Explore query window to diagnose missing objects and error messages and used the Content Validator to test and update content after changing a LookML object.

Finish your quest

This self-paced lab is part of the Manage Data Models in Looker skill badge quest. A quest is a series of related labs that form a learning path. Completing a 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 a quest and get immediate completion credit if you've taken this lab. See other available quests.

Next steps / Learn more

- LookML quick reference
- LookML terms and concepts
- Looker Community
- Additional LookML basics

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 February 9, 2023

Lab Last Tested February 9, 2023