



Introduction to Looker and LookML

Welcome to the **Introduction to Looker and LookML** module.

In this module, we will explore the Looker platform as LookML developers and review the key terminology and tools for writing LookML code.

Agenda

- 01 Introduction to the Looker Platform
- 02 Understanding your Users' Experience
- 03 LookML Project Hierarchy
- 04 Example 1: The Looker Development Environment



We will begin with an overview of the Looker platform and architecture to understand how Looker can play a key role in your organization's data workflows.

Next, we will complete a walkthrough of the Looker User Interface to understand how your business users leverage Looker to explore and analyze data.

Then, we will review the LookML project hierarchy to understand its key components and the role that each plays in the overall LookML structure.

Last, we will explore the Looker development environment and review its key features for your workflow as a LookML developer.

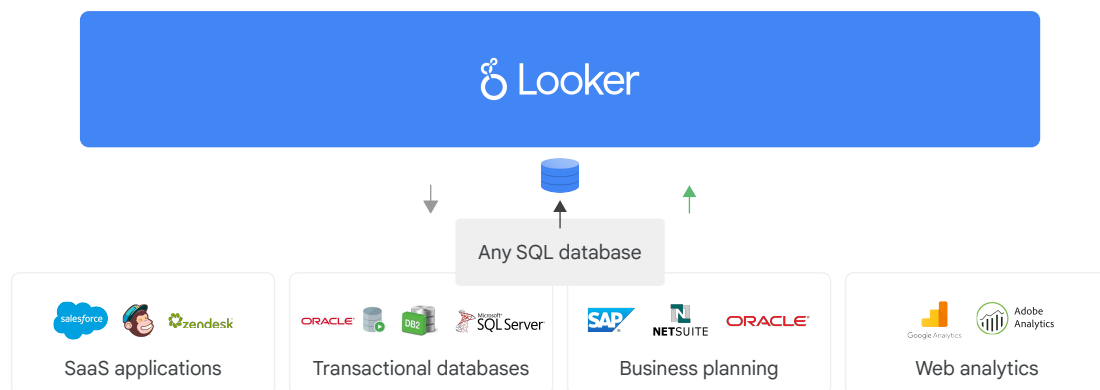
Introduction to Looker and LookML

- 01 Introduction to the Looker Platform
- 02 Understanding your Users' Experience
- 03 LookML Project Hierarchy
- 04 Example 1: The Looker Development Environment



Looker is a Business Intelligence (BI) software and big data analytics platform that helps business users to explore, analyze and share real-time data analytics easily.

Architecture: Looker in the data stack



Google Cloud

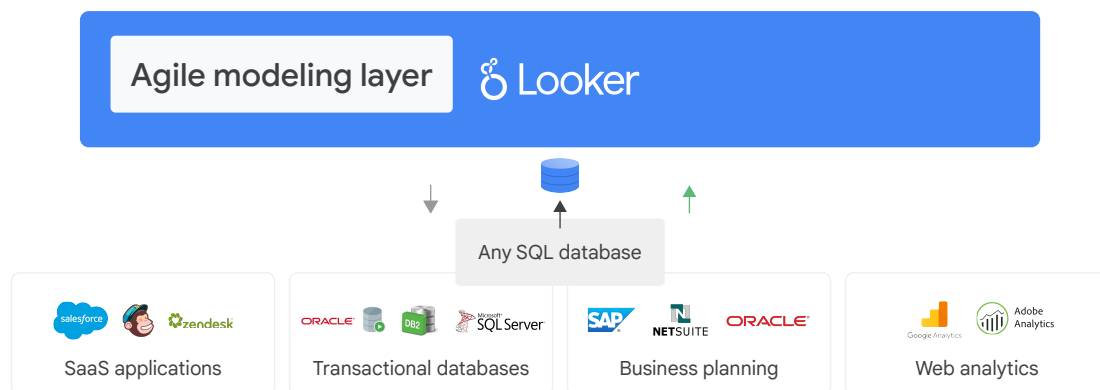
As a browser-based, Software as a Service (SaaS) platform, Looker connects directly to SQL databases.

For example, you can connect Looker to:

- Other SaaS applications such as Salesforce, Mailchimp, and Zendesk.
- Heavy read-write operations in transactional databases such as Oracle, IBM Db2, and Microsoft SQL Server.
- Business planning tools such as SAP, NetSuite, and Oracle.
- And web analytics products such as Google Analytics or Adobe Analytics.

These are just a few examples. Looker is multi-cloud and [supports over 65 database dialects](#).

Architecture: Looker's agile modeling layer

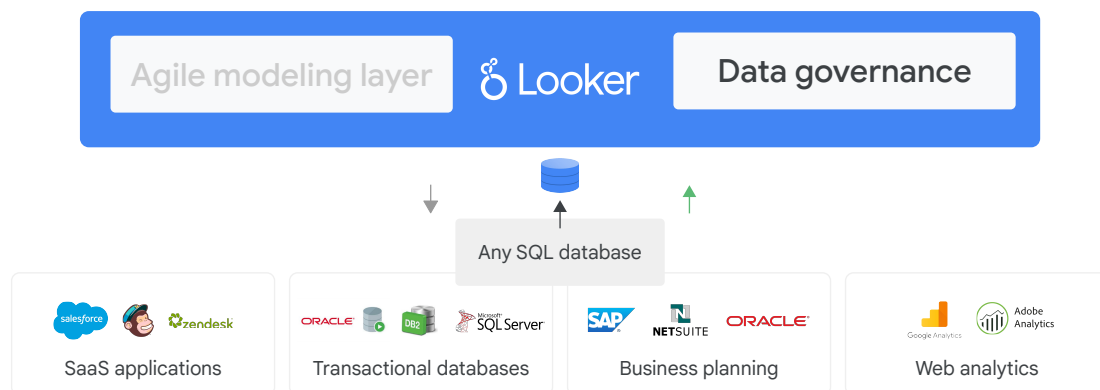


One major benefit of Looker is its agile modeling layer, which can save data teams and business analysts time that would otherwise be spent manually writing and editing SQL queries. Looker's agile modeling layer allows developers to define, through Looker Modeling Language or LookML, how the database is structured and how the tables and columns relate to each other.

A useful way to think about LookML is that it is an abstraction layer for SQL that developers use to tell Looker what data to use from the connected database and how it should interpret that data.

As users explore and analyze the data, Looker uses the defined LookML models to automatically generate SQL **SELECT** queries to send to the database and return the appropriate results.

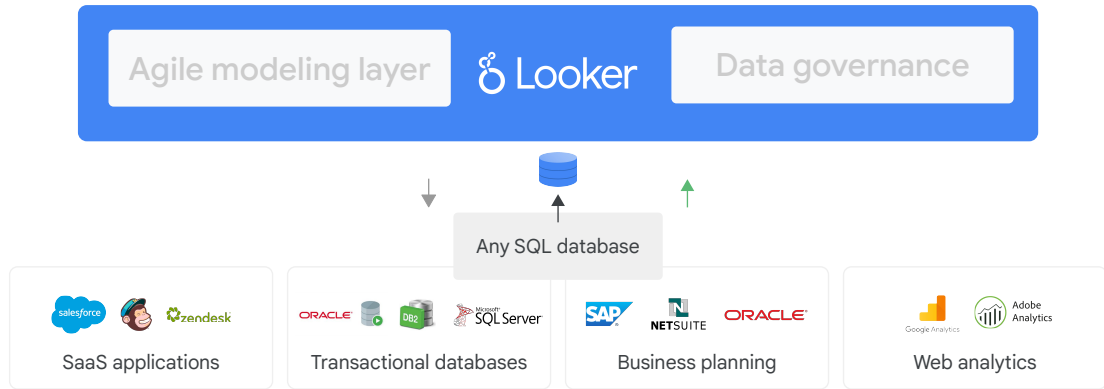
Architecture: Looker data governance



Another benefit of Looker is data governance, which means that you can define a single source of truth for data that everyone in the organization can understand and trust.

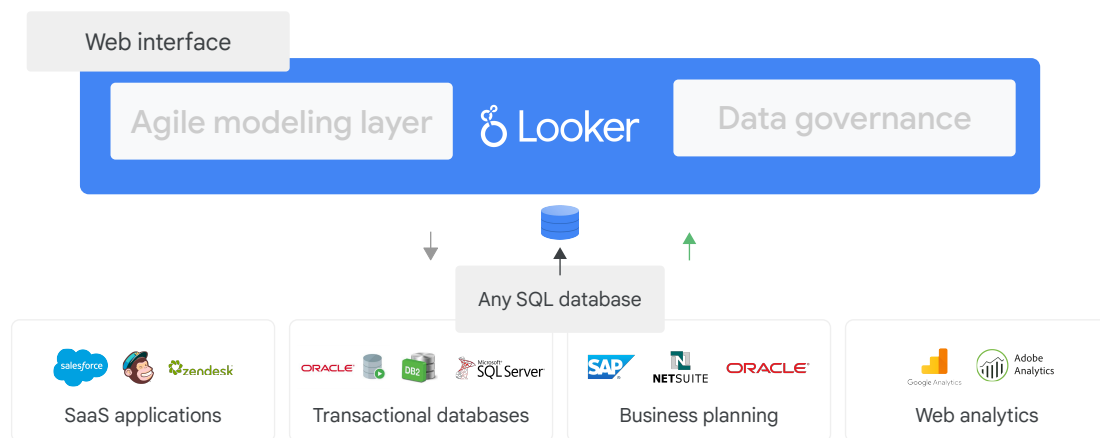
In Looker, you can enforce various types of data security and governance through the Looker user interface (UI), such as assigning specific user roles, as well as through LookML, such as providing access to specific fields or rows of data.

Architecture: Looker data democratization



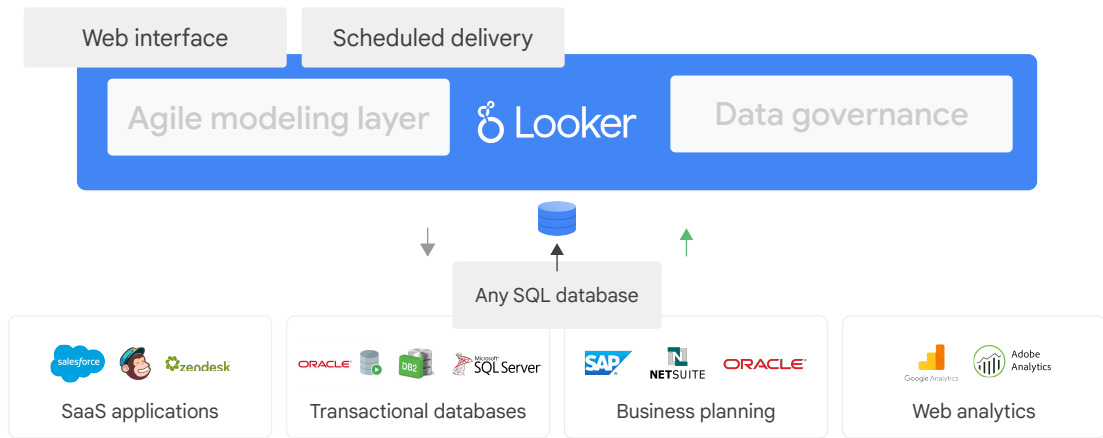
To help organizations disseminate data, Looker can surface and expose query results in several ways.

Architecture Looker's user interface



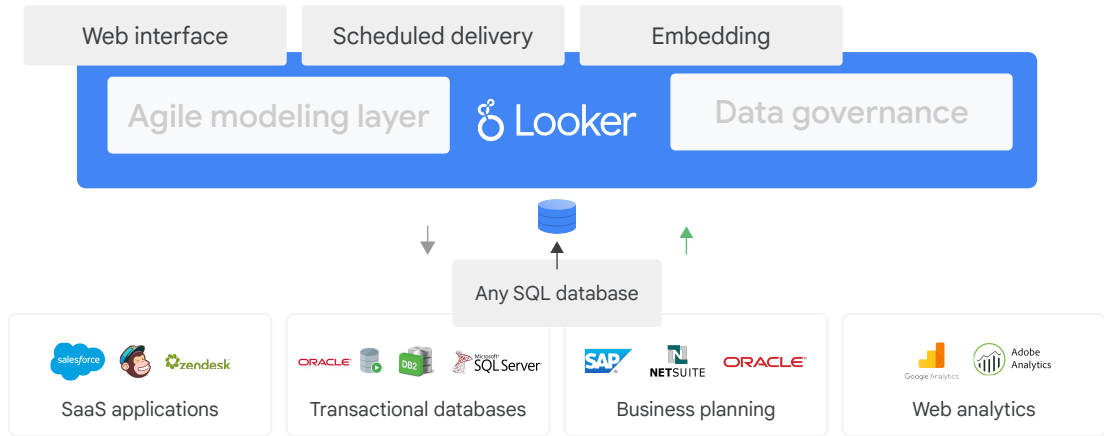
The first is through the web interface. This can be in the form of Explores (which are report-builder interfaces), Looks (which are standalone reports or visualizations), and dashboards (which contain multiple visualizations).

Architecture: Looker's data scheduling and delivery



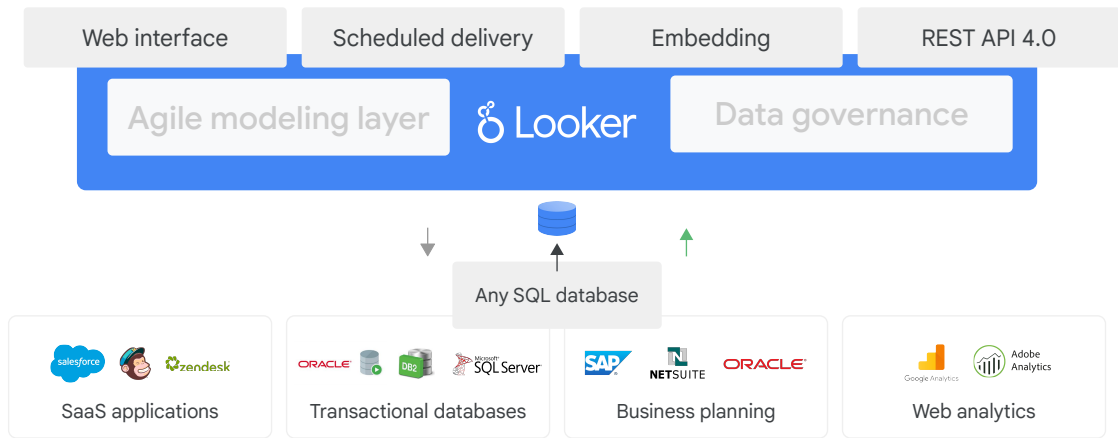
Another way is through scheduled data deliveries, such as sending Looks and dashboards to specific email addresses or Cloud Storage buckets on a one-time or recurring basis.

Architecture: Embed Looker content in your applications



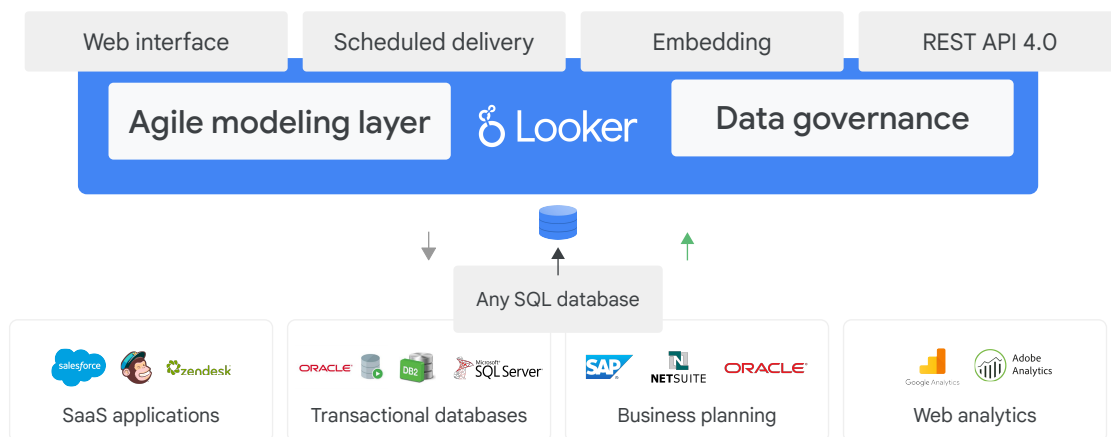
Explores, dashboards, and Looks can also be embedded within other websites or applications.

Architecture: The Looker API



Last, Looker provides a REST API that allows you to retrieve, analyze, and transform data and metadata directly from the Looker platform.

Architecture: Looker's rich development framework



Looker's unique architecture provides a rich development framework that is built to support enterprise-grade workflows and help your users and tools access the most accurate and up-to-date version of your organization's data.

With this unified view into your organization's data, you, as a LookML developer, can curate data experiences to ensure that both people and systems get the data they need, how and when they need it.

Introduction to Looker and LookML

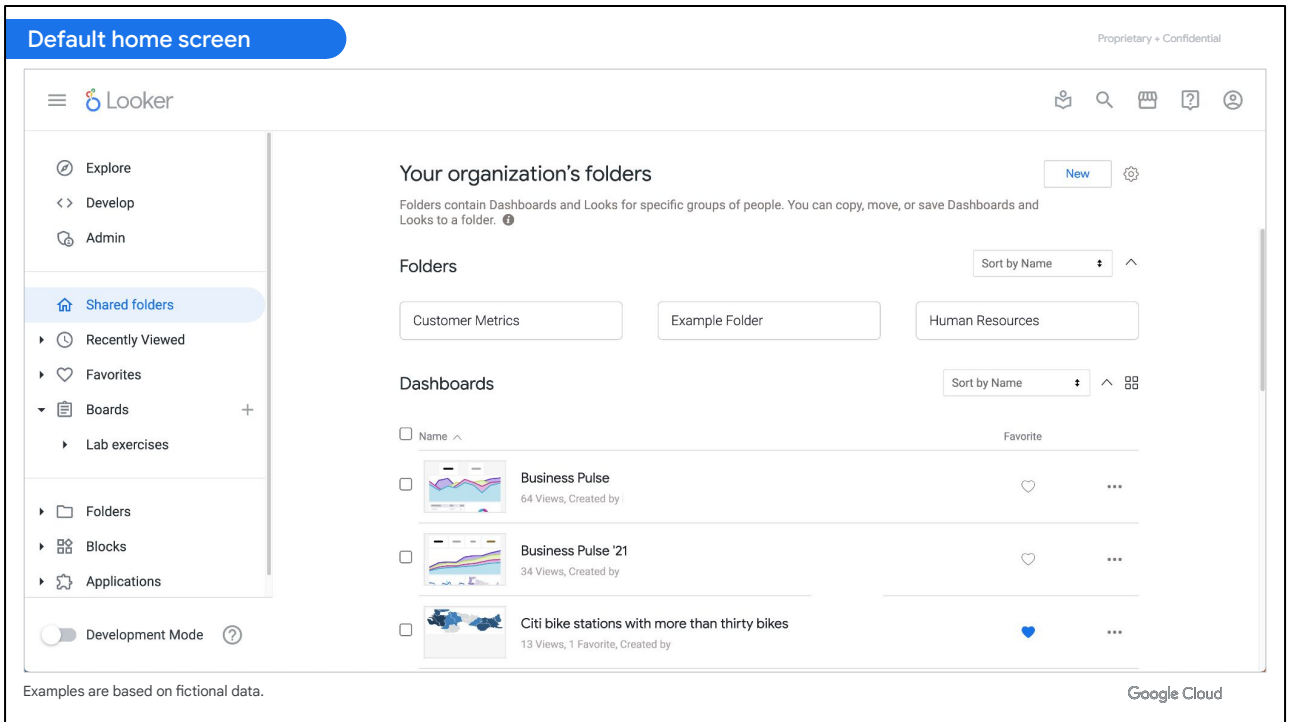
- 01 Introduction to the Looker Platform
- 02 [Understanding your Users' Experience](#)
- 03 LookML Project Hierarchy
- 04 Example 1: The Looker Development Environment



To understand how you as a LookML developer can support business users, it's important to become familiar with the business user experience and how they use the Looker platform on a daily basis to answer data-driven questions.

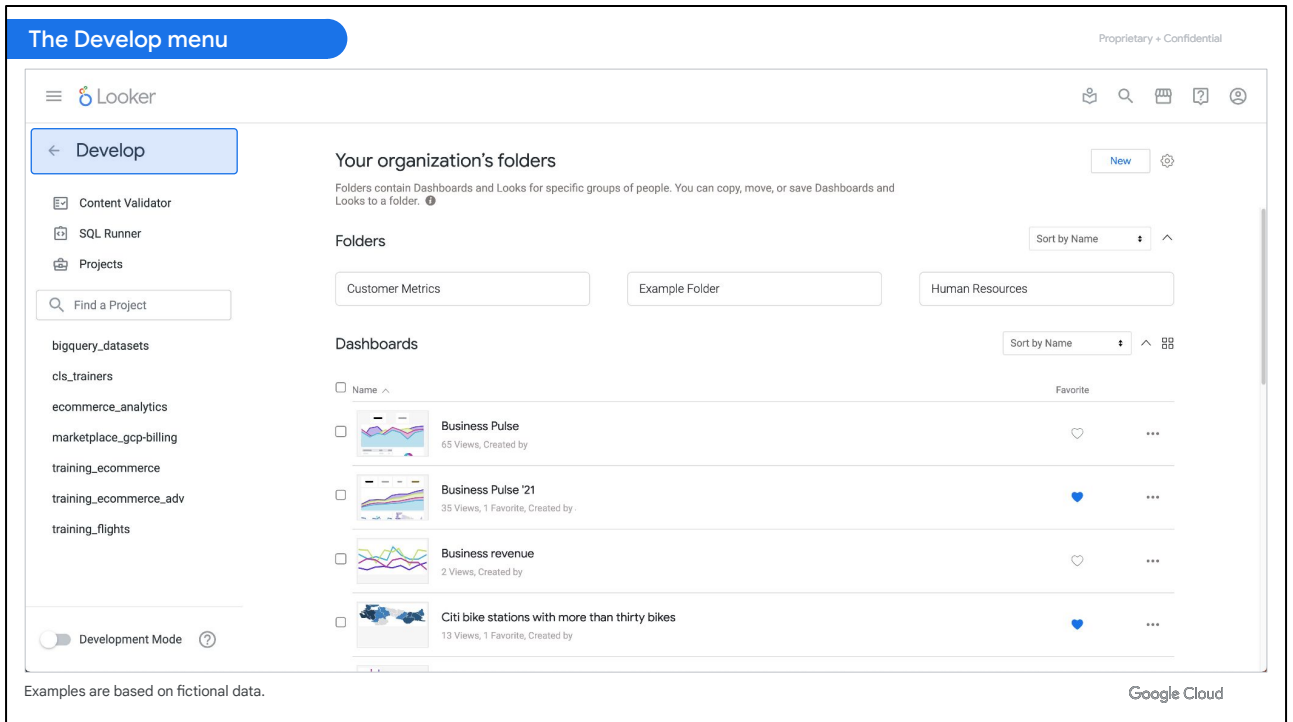
Furthermore, the ability to navigate the Looker platform as a business user will help you to fully test your changes to LookML code by reviewing how results appear when accessed by business users.

Let's explore the business user experience by walking through an example Looker instance.



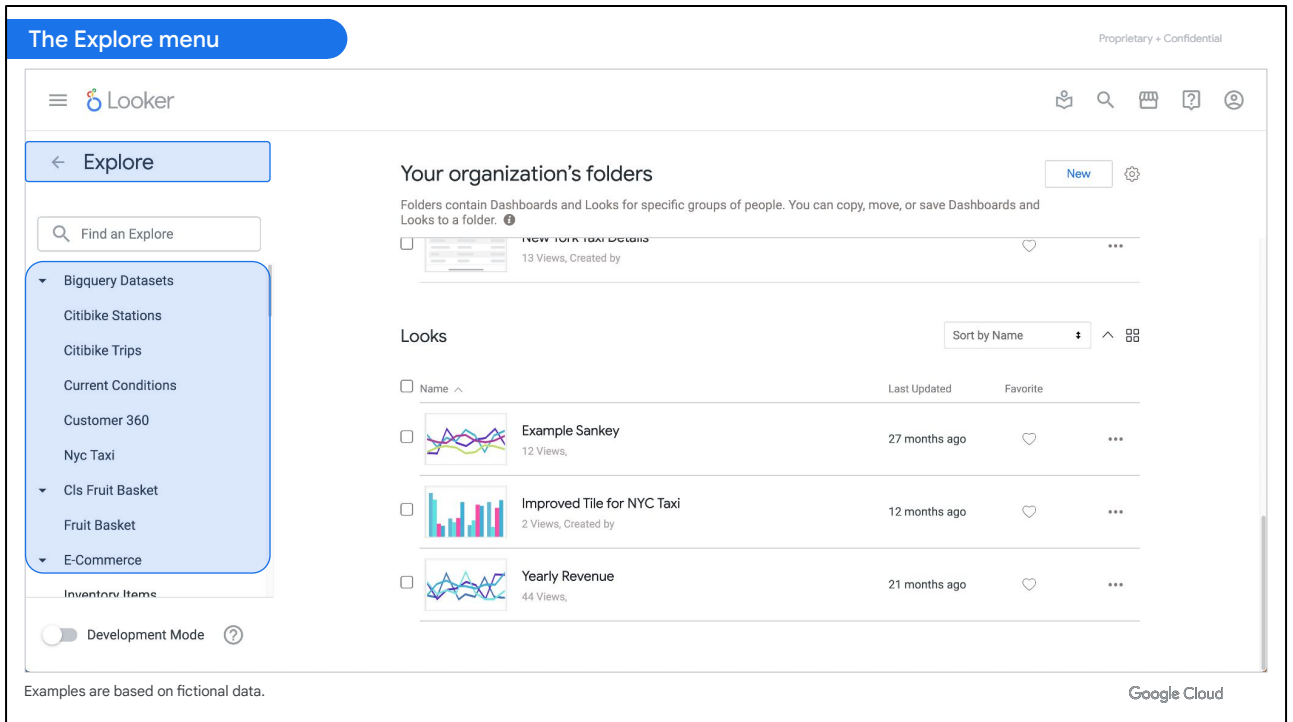
When you first log in to the Looker platform, your home page may vary depending on what your company's Looker administrator has configured.

In this example instance, we will begin on the **Shared** folders page. Folders in Looker are where content lives, just as files in your computer or Google Drive are stored in folders.



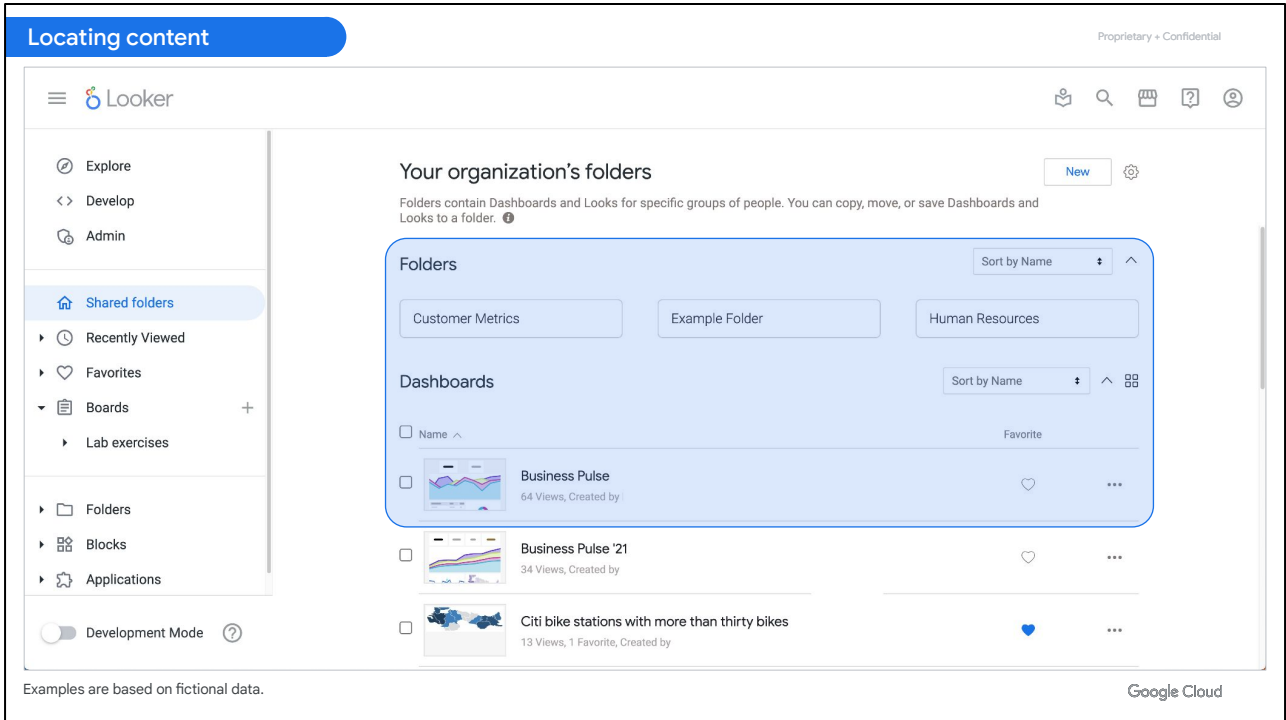
As you may know, Looker has two primary categories of users: business users and developers.

As a LookML developer, you use the **Develop** environment to curate report-builder interfaces called Explores that are used by business users and to configure other aspects in your Looker instance such as rules for caching and data security.

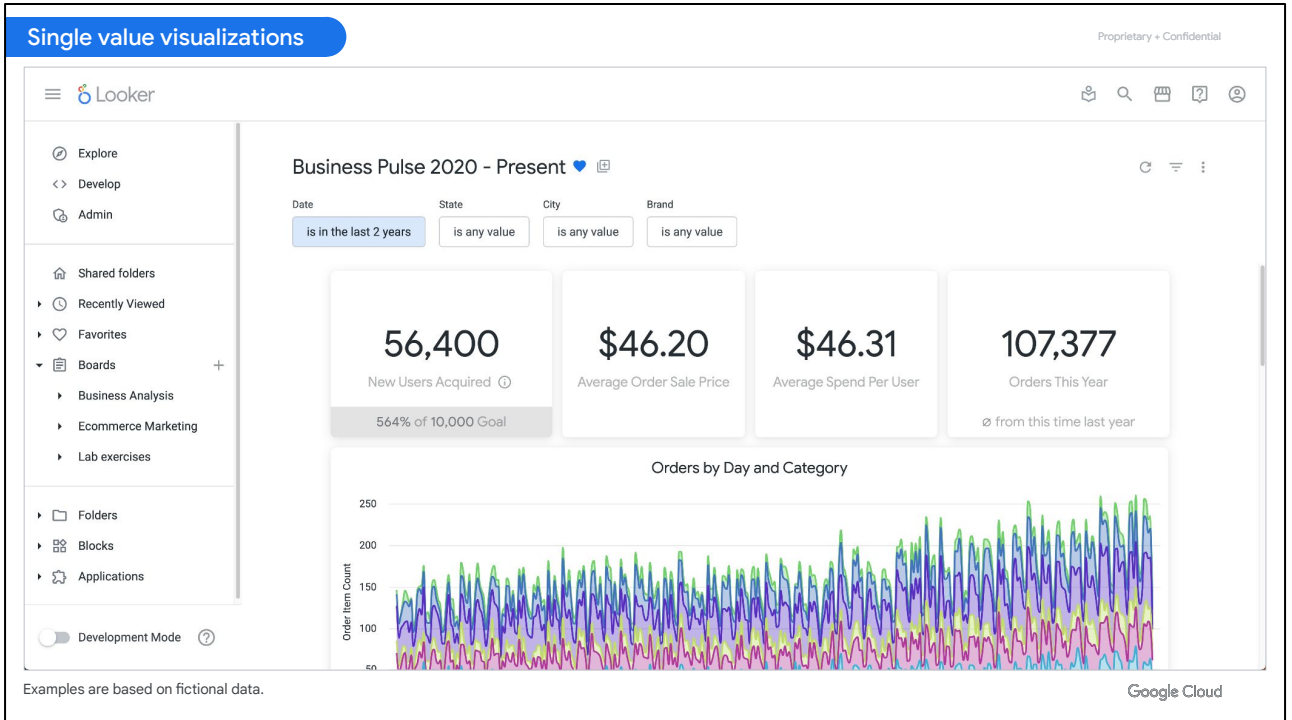


Business users can click on Explore to see a list of the custom Explores that LookML developers have modeled for them. Within the Explores, they can analyze and visualize data to answer business questions and save their results as visualizations and reports.

This example Looker instance has many Explores including one for Order Items under the E-Commerce Training header and another for Flights under the FAA header.



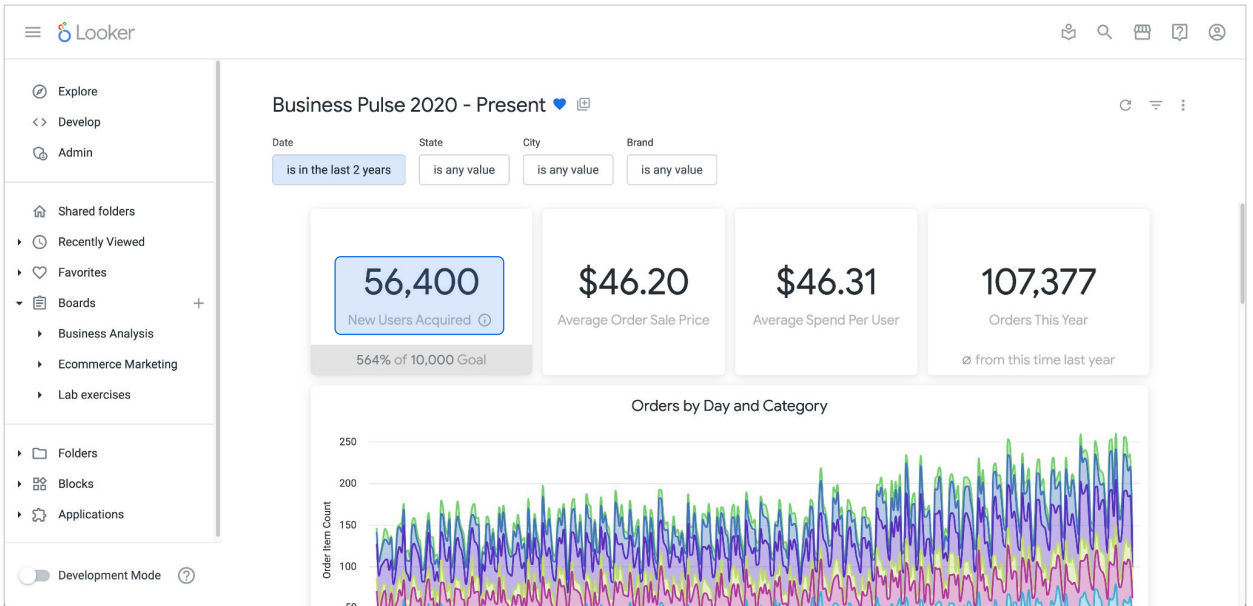
As a business user working at a hypothetical ecommerce company, you can also navigate through the various folders of the Looker instance to find content that has already been built using the available Explores, such as one of the Business Pulse dashboards.



For business users, this example Business Pulse dashboard provides some high-level key performance indicators (KPIs) that a typical ecommerce company might care about, such as the number of new users acquired and the average order amount.

In Looker, these are called single value visualizations.

Scrolling down, you can see other visualizations such as the number of orders by category over time in an area chart.



Let's say you want to learn more about new users your company has acquired. You can click on the number provided in the New Users Acquired tile to drill down to the underlying data.

The screenshot shows the Google Looker interface. At the top left, there is a blue header with the text 'Drilling into data'. In the top right corner, it says 'Proprietary + Confidential'. The main content area is a white window titled 'New Users Acquired' with 'Explore' and 'Download' buttons. Below the title, it says 'ORDER ITEMS (5 Filters)'. The table has four columns: 'ID', 'Last Name', and 'First Name'. The data rows are numbered 1 through 16, with the 17th row partially visible. The table is set against a dark background with a colorful line chart at the bottom.

ID	Last Name	First Name
1 96,209	Dipietro	Virginia
2 96,212	Carpenter	Tonya
3 96,213	Branch	Marlene
4 96,214	Olson	Jacob
5 96,215	Graves	David
6 96,216	Moore	Lynn
7 96,217	Gallegos	Shelly
8 96,219	Rockwood	Todd
9 96,221	Saunders	Jessica
10 96,222	Tucker	Peter
11 96,223	Merritt	Troy
12 96,224	Marcum	Lisa
13 96,225	Pietrzak	James
14 96,226	Williams	Susan
15 96,227	Houston	Connie
16 96,228	Rose	Ashley
17 96,229

Examples are based on fictional data.

Google Cloud

After drilling down on the New Users Acquired tile, you can view the granular rows of data that constitute the overall number of new users acquired.

For example, for this tile, you can see each user's ID, first name, and last name. In Looker, these data attributes are called dimensions.

The screenshot shows a Google Looker interface. At the top left, there is a navigation menu with the Looker logo. The main content area displays a table titled "New Users Acquired" with the subtitle "ORDER ITEMS (5 Filters) v". The table has three columns: "ID", "Last Name", and "First Name". The data is as follows:

ID	Last Name	First Name
1 96,209	Dipietro	Virginia
2 96,212	Carpenter	Tonya
3 96,213	Branch	Marlene
4 96,214	Olson	Jacob
5 96,215	Graves	David
6 96,216	Moore	Lynn
7 96,217	Gallegos	Shelly
8 96,219	Rockwood	Todd
9 96,221	Saunders	Jessica
10 96,222	Tucker	Peter
11 96,223	Merritt	Troy
12 96,224	Marcum	Lisa
13 96,225	Pietrzak	James
14 96,226	Williams	Susan
15 96,227	Houston	Connie
16 96,228	Rose	Ashley

At the top right of the table area, there are buttons for "Explore" (with a refresh icon), "Download", and a close "X" button. The bottom of the interface shows a "Development mode" indicator and a Google Cloud logo.

Examples are based on fictional data.

Google Cloud

If you want to learn even more about these users, you can click on the **Explore** link.

The Looker Explore

Proprietary + Confidential

Looker

Explore 500 rows - from cache - 9m ago Run

Order Items

Find a Field

Start typing to search

All Fields In Use

- Custom Fields + Add
- Distribution Centers
- Inventory Items
- Order Items 1
- Products 1
- Users 6

99 fields Go to LookML

Filters (5) Order Items Created Date is in the past 2 years Products Brand is any value Users City is any value Users Created Date is in the past 2 years Users State is any value

Visualization

Data Results SQL Add calculation Row Limit 500 Totals Subtotals

Row limit reached. Results may be incomplete

Users ID ↑	Users Last Name	Users First Name
1	96209 Dipietro	Virginia
2	96212 Carpenter	Tonya
3	96213 Branch	Marlene
4	96214 Olson	Jacob
5	96215 Graves	David
6	96216 Moore	Lynn
7	96217 Gallegos	Shelly
8	96219 Rockwood	Todd
9	96221 Saunders	Jessica
10	96222 Tucker	Peter
11	96223 Merritt	Troy
12	96224 Marcum	Lisa
13	96225 Pietrzak	James
14	96226 Williams	Susan
15	96227 Houston	Connie
16	96228 Rose	Ashley
17	96229 Bieker	John

Examples are based on fictional data.

Google Cloud

This takes you out of the dashboard and into the Explore, which is the report-builder interface that has been curated by a LookML developer.

In this example, the overall Explore is called **Order Items**, but there are several expandable groups of fields in the field picker found in the left-side panel. These are called views.

An Explore is composed of one or more views. For example, when you want to analyze order information, you also might want to include fields from other related views such as **Users**.

In SQL terms, each view represents a database table, and the tables are pre-joined in the LookML model file to define the overall Explore, like this **Order Items** Explore.

Working with Explores

Proprietary + Confidential

The screenshot shows the Looker Explore interface. On the left, the 'Order Items' view is selected, and the 'Users' dimension is expanded to show 'Age' as a dimension. The main area displays a table of 17 rows of user data. A yellow warning banner indicates that the row limit of 500 has been reached. The table columns are Users ID, Users Last Name, Users First Name, and Users Age.

Users ID ↑	Users Last Name	Users First Name	Users Age
1	96209 Dipietro	Virginia	71
2	96212 Carpenter	Tonya	22
3	96213 Branch	Marlene	16
4	96214 Olson	Jacob	16
5	96215 Graves	David	41
6	96216 Moore	Lynn	12
7	96217 Gallegos	Shelly	43
8	96219 Rockwood	Todd	12
9	96221 Saunders	Jessica	37
10	96222 Tucker	Peter	37
11	96223 Merritt	Troy	36
12	96224 Marcum	Lisa	16
13	96225 Pietrzak	James	74
14	96226 Williams	Susan	42
15	96227 Houston	Connie	51
16	96228 Rose	Ashley	12
17	96229 Bleker	John	54

Examples are based on fictional data.

Google Cloud

Using this Explore, you can ask questions about the newly acquired users. For example, maybe you are curious about the age of each of these users.

To see age information about these users, you can expand the **Users** view, and click on the **Age** dimension to add it to your results set. Then, you can click **Run** to view the results.

Modifying the Explore

Proprietary + Confidential

The screenshot shows the Looker interface for an 'Explore' view. On the left, the 'Order Items' view is selected, and the 'Total Revenue' measure is highlighted in the 'MEASURES' section. The main table displays 17 rows of data with columns: Users ID, Users Last Name, Users First Name, Users Age, and Order Items Total Revenue. A yellow warning banner at the top of the table reads 'Row limit reached. Results may be incomplete'. The 'Run' button is visible in the top right corner.

Users ID ↑	Users Last Name	Users First Name	Users Age	Order Items Total Revenue	
1	96209	Dipietro	Virginia	71	\$16.50
2	96212	Carpenter	Tonya	22	\$90.99
3	96213	Branch	Mariene	16	\$29.99
4	96214	Olson	Jacob	16	\$149.98
5	96215	Graves	David	41	\$21.99
6	96216	Moore	Lynn	12	\$73.99
7	96217	Gallegos	Shelly	43	\$232.93
8	96219	Rockwood	Todd	12	\$59.95
9	96221	Saunders	Jessica	37	\$68.00
10	96222	Tucker	Peter	37	\$45.00
11	96223	Merritt	Troy	36	\$39.95
12	96224	Marcum	Lisa	16	\$12.58
13	96225	Pietrzak	James	74	\$68.65
14	96226	Williams	Susan	42	\$168.00
15	96227	Houston	Connie	51	\$39.99
16	96228	Rose	Ashley	12	\$39.87
17	96229	Bleker	John	54	\$64.00

Examples are based on fictional data.

Google Cloud

If you want to know how much each of these new users has already spent, you can expand the **Order Items** view, and click on the **Total Revenue** measure, which aggregates all of the purchases for each user.

Once again, you need to click **Run** to view the new results.

The screenshot shows the Looker interface with the 'Order Items' explore selected. The 'SQL' tab is active, displaying a query that filters for users created in the last 2 years and orders created in the last 2 years. The query selects user details and calculates the total revenue for each user. The interface includes a sidebar with field lists, a top navigation bar, and a bottom status bar.

```
SELECT
  users.id AS users_id,
  users.last_name AS users_last_name,
  users.first_name AS users_first_name,
  users.age AS users_age,
  COALESCE(SUM(order_items.sale_price ), 0) AS order_items_total_revenue
FROM `cloud-training-demos.looker_ecomm.order_items`
  AS order_items
LEFT JOIN `cloud-training-demos.looker_ecomm.users`
  AS users ON order_items.user_id = users.id
WHERE ((( order_items.created_at ) >= ((TIMESTAMP(DATETIME_ADD(DATETIME(TIMESTAMP_TRUNC(TIMESTAMP_TRUNC(CURRENT_TIMESTAMP(), DAY), YEAR)), INTERVAL
-1 YEAR)))) AND ( order_items.created_at ) < ((TIMESTAMP(DATETIME_ADD(DATETIME(TIMESTAMP_TRUNC(TIMESTAMP_TRUNC(CURRENT_TIMESTAMP(), DAY), YEAR)), INTERVAL
2 YEAR)))))) AND ((( users.created_at ) >= ((TIMESTAMP(DATETIME_ADD(DATETIME(TIMESTAMP_TRUNC(TIMESTAMP_TRUNC(CURRENT_TIMESTAMP(), DAY), YEAR)), INTERVAL
-1 YEAR)))) AND ( users.created_at ) < ((TIMESTAMP(DATETIME_ADD(DATETIME(TIMESTAMP_TRUNC(TIMESTAMP_TRUNC(CURRENT_TIMESTAMP(), DAY), YEAR)), INTERVAL
2 YEAR))))))
GROUP BY
  1,
  2,
  3,
  4
ORDER BY
  1
LIMIT 500
```

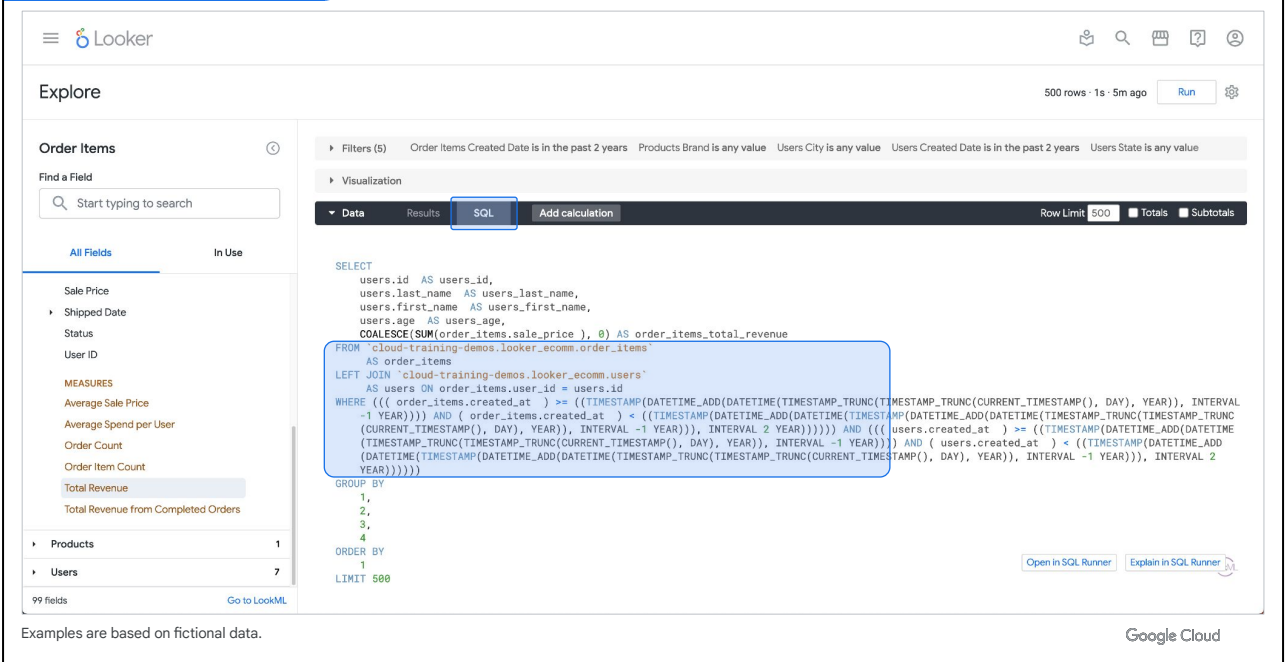
Examples are based on fictional data.

Google Cloud

Behind the scenes, Looker has automatically generated a SQL query for the drill-down results and is updating it with the new selections.

As a LookML developer, you can click on the SQL tab—typically hidden from business users—to see that Looker is **SELECTING** `users.id`, `users.first_name`, `users.last_name`, and `users.age`, which are the dimensions selected in the Explore.

Looker is also executing a **SUM** of `order_items.sale_price` for `total_revenue`, which is the measure selected to aggregate the purchase data for each user.



The screenshot shows the Looker interface with the following components:

- Header:** Looker logo and navigation icons.
- Explore View:** Shows 500 rows, 1s, 5m ago, and a Run button.
- Filters (5):** Order Items Created Date is in the past 2 years, Products Brand is any value, Users City is any value, Users Created Date is in the past 2 years, Users State is any value.
- Visualization:** SQL tab is selected.
- SQL Query:**

```
SELECT
  users.id AS users_id,
  users.last_name AS users_last_name,
  users.first_name AS users_first_name,
  users.age AS users_age,
  COALESCE(SUM(order_items.sale_price ), 0) AS order_items_total_revenue
FROM `cloud-training-demos.looker_ecomm.order_items`
  AS order_items
LEFT JOIN `cloud-training-demos.looker_ecomm.users`
  AS users ON order_items.user_id = users.id
WHERE ((( order_items.created_at ) >= ((TIMESTAMP(DATETIME_ADD(DATETIME(TIMESTAMP_TRUNC(TIMESTAMP_TRUNC(CURRENT_TIMESTAMP(), DAY), YEAR)), INTERVAL
-1 YEAR)))) AND ( order_items.created_at ) < ((TIMESTAMP(DATETIME_ADD(DATETIME(TIMESTAMP_TRUNC(TIMESTAMP_TRUNC(CURRENT_TIMESTAMP(), DAY), YEAR)), INTERVAL
2 YEAR)))))) AND ((( users.created_at ) >= ((TIMESTAMP(DATETIME_ADD(DATETIME(TIMESTAMP_TRUNC(TIMESTAMP_TRUNC(CURRENT_TIMESTAMP(), DAY), YEAR)), INTERVAL
-1 YEAR)))) AND ( users.created_at ) < ((TIMESTAMP(DATETIME_ADD
(DATETIME(TIMESTAMP_TRUNC(TIMESTAMP_TRUNC(CURRENT_TIMESTAMP(), DAY), YEAR), INTERVAL -1 YEAR)), INTERVAL 2
YEAR))))))
GROUP BY
  1,
  2,
  3,
  4
ORDER BY
  1
LIMIT 500
```
- Buttons:** Open in SQL Runner, Explain in SQL Runner.
- Footer:** Examples are based on fictional data. Google Cloud.

You can also see that the data comes **FROM** a table called **looker_ecomm.order_items**, with a **LEFT JOIN** to **looker_ecomm.users**.

Finally, there's a big **WHERE** condition that is identifying only the new users from the past 180 days.

The screenshot shows the Looker 'Explore' interface. On the left, the 'Order Items' view is selected, showing a list of fields including 'Sale Price', 'Shipped Date', 'Status', 'User ID', and 'MEASURES' like 'Average Sale Price' and 'Total Revenue'. The 'Total Revenue' field is highlighted. The main area displays a table with 17 rows of data. The table has columns for 'Users ID', 'Users Last Name', 'Users First Name', 'Users Age', and 'Order Items Total Revenue'. A yellow warning banner at the top of the table indicates 'Row limit reached. Results may be incomplete'. The 'Run' button is visible in the top right corner.

Users ID ↑	Users Last Name	Users First Name	Users Age	Order Items Total Revenue
1	DiPietro	Virginia	71	\$16.50
2	Carpenter	Tonya	22	\$90.99
3	Branch	Mariene	16	\$29.99
4	Olson	Jacob	16	\$149.98
5	Graves	David	41	\$21.99
6	Moore	Lynn	12	\$73.99
7	Gallegos	Shelly	43	\$232.93
8	Rockwood	Todd	12	\$59.95
9	Saunders	Jessica	37	\$68.00
10	Tucker	Peter	37	\$45.00
11	Merritt	Troy	36	\$39.95
12	Marcum	Lisa	16	\$12.58
13	Pietrzak	James	74	\$68.65
14	Williams	Susan	42	\$168.00
15	Houston	Connie	51	\$39.99
16	Rose	Ashley	12	\$39.87
17	Bleker	John	54	\$64.00

Examples are based on fictional data.

Google Cloud

Rather than writing this SQL query manually, it was so easy for the business user to click on some fields and get this output automatically. This is all because of the agile modeling layer in LookML.

A LookML developer or team of developers had already created this **Order Items** Explore, specified which views should be in the Explore, specified which dimensions and measures should be in each view, and defined the SQL logic for each dimension and measure.

Save as a Look

Proprietary + Confidential

The screenshot shows the Looker 'Explore' interface. On the left, the 'Order Items' table is selected, and a list of fields is visible, including 'Total Revenue' and 'Total Revenue from Completed Orders'. The main area displays a table with columns for ID, Last Name, First Name, and a numerical value. A 'Save...' menu is open in the top right corner, with the 'As a Look' option highlighted. The menu also includes options like 'As a new dashboard', 'To an existing dashboard', 'Download', 'Open in Data Studio', 'Send', 'Save and schedule', 'Share', 'Get LookML', 'Merge results', 'Remove fields and filters', and 'Clear cache and refresh'. A warning message 'Row limit reached. Results may be incomplete' is visible above the table.

ID	Last Name	First Name	
1	Dipietro	Virginia	71
2	Carpenter	Tonya	22
3	Branch	Marlene	16
4	Olson	Jacob	16
5	Graves	David	41
6	Moore	Lynn	12
7	Gallegos	Shelly	43
8	Rockwood	Todd	12
9	Saunders	Jessica	37

Users ID ↑	Users Last Name	Users First Name	Users Age	Order Items T	
1	96209	Dipietro	Virginia	71	
2	96212	Carpenter	Tonya	22	
3	96213	Branch	Marlene	16	\$29.99
4	96214	Olson	Jacob	16	\$149.98
5	96215	Graves	David	41	\$21.99
6	96216	Moore	Lynn	12	\$73.99
7	96217	Gallegos	Shelly	43	\$232.93

Examples are based on fictional data.

Google Cloud

Now, you can save these results as a standalone report, or Look, for ongoing reference by clicking on the gear icon in the top right, select **Save**, and then choose **As a Look** from the fanout menu.

Looker

- Explore
- Develop
- Admin

- Shared folders
- Recently Viewed
- Favorites
- Boards
 - Business Analysis
 - Ecommerce Marketing
 - Lab exercises
- Folders
- Blocks
- Applications

Details for new users acquired 500 rows · from cache · 26m ago

Filters (5)
Order Items Created Date is in the past 2 years
Products Brand is any value
Users City is any value
Reset Look
Explore from Here

Visualization

ID	Last Name	First Name	Age	Total Revenue
1	Dipietro	Virginia	71	\$16.50
2	Carpenter	Tonya	22	\$90.99
3	Branch	Marlene	16	\$29.99
4	Olson	Jacob	16	\$149.98
5	Greves	David	41	\$21.99
6	Moore	Lynn	12	\$73.99
7	Gallegos	Shelly	43	\$232.93
8	Rockwood	Todd	12	\$59.95
9	Saunders	Jessica	37	\$68.00
10	Tucker	Peter	37	\$45.00
11	Merritt	Troy	36	\$39.95
12	Marcum	Lisa	16	\$12.58
13	Pietrzak	James	74	\$68.65
14	Williams	Susan	42	\$168.00
15	Houston	Connie	51	\$39.99
16	Rose	Ashley	12	\$39.87
17	Bieker	John	54	\$64.00
18	Perales	Barton	41	\$78.00
19	Deal	Pete	44	\$33.21

Details

Description: None Edit

Scheduled: No Create Schedules

On Dashboards: No Add To Dashboard

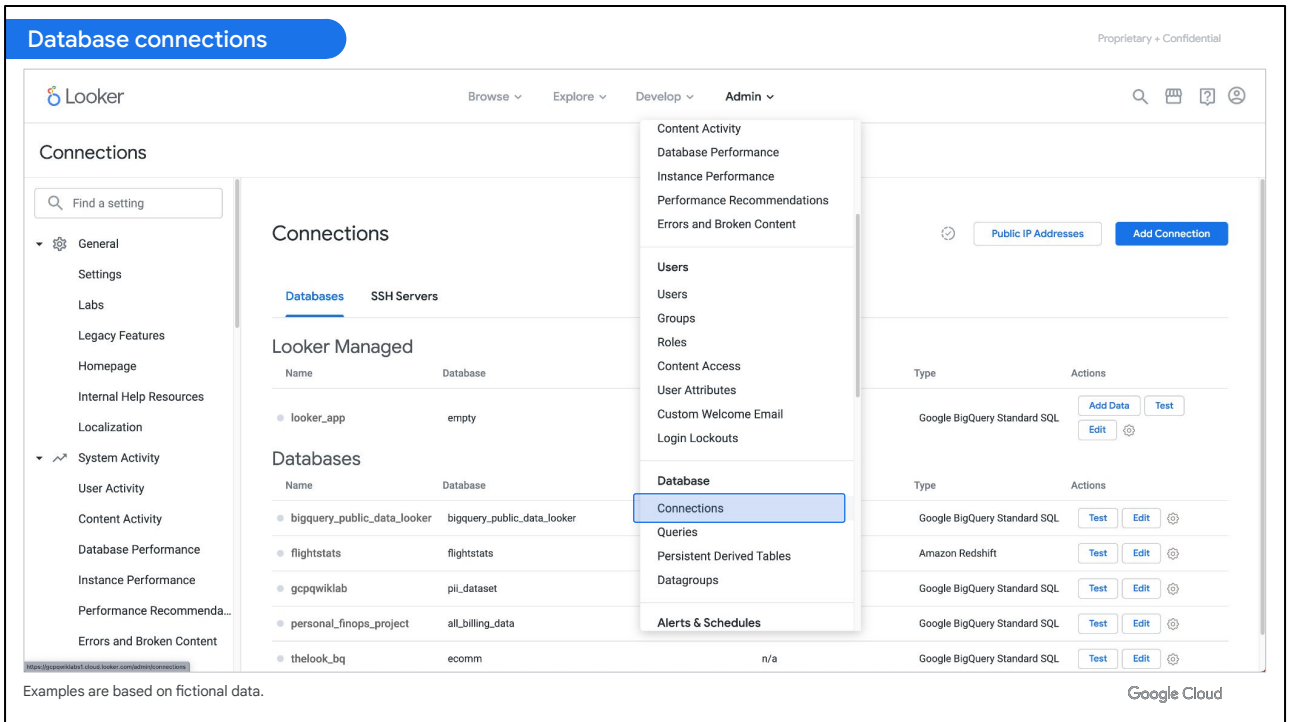
Created by: on 2022-11-29

Updated by: on 2022-11-29

Examples are based on fictional data.

Google Cloud

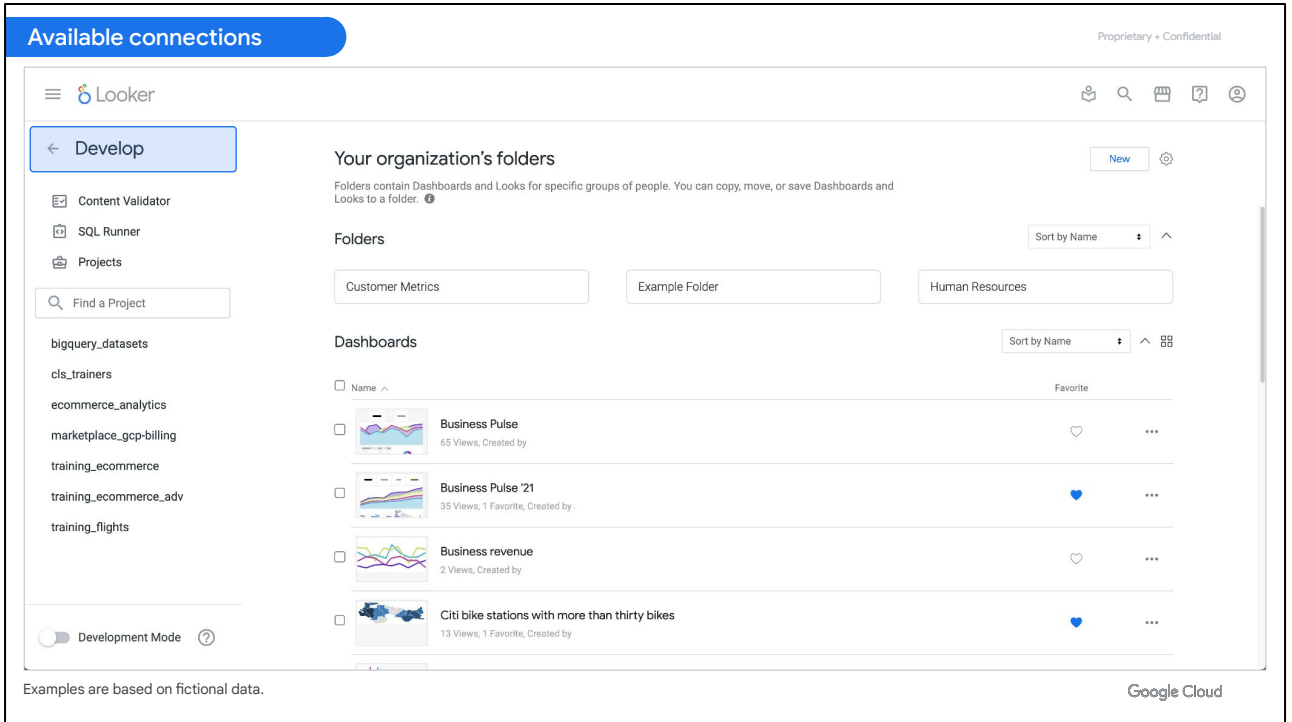
That was a quick demonstration of the significance and value of LookML for empowering business users to explore and analyze data.



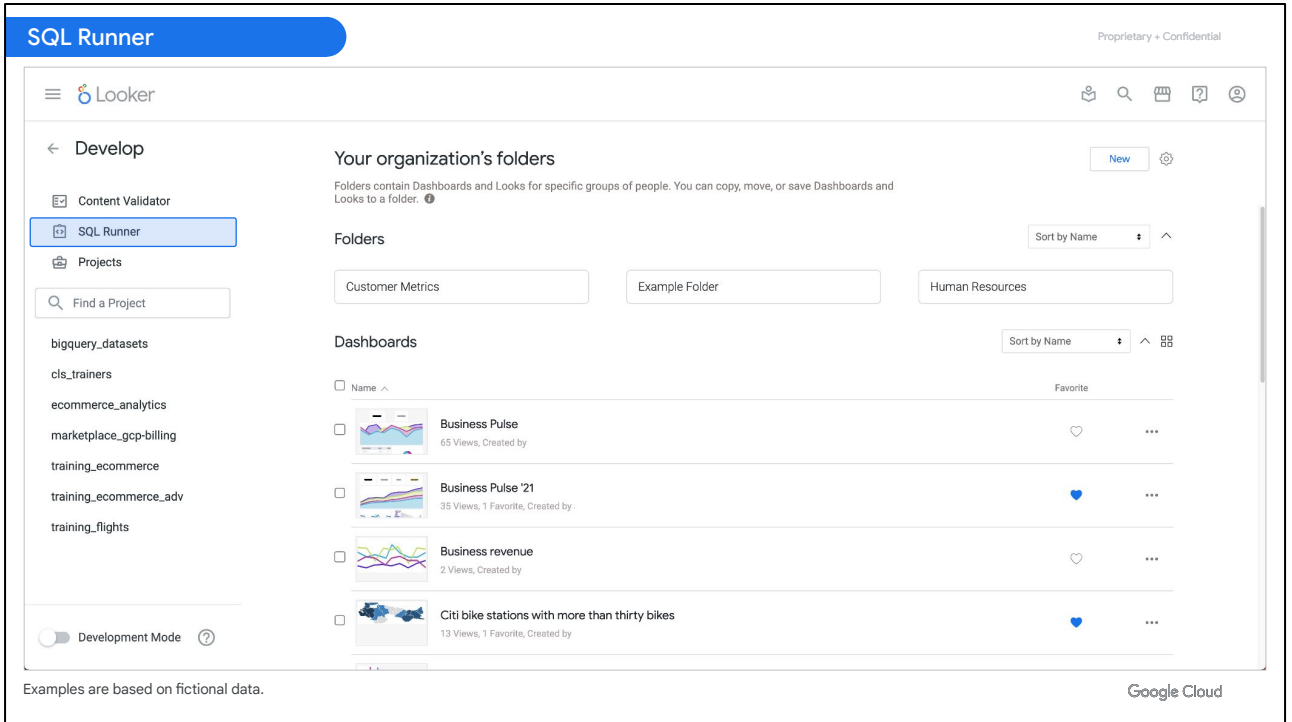
If you're completely new to Looker, there's a fundamental prerequisite to be aware of regarding the underlying data that is used by business users.

A Looker administrator needs to create a connection to a specific data source such as your company's primary database, in order for LookML developers to access and model that data to curate Explores for business users.

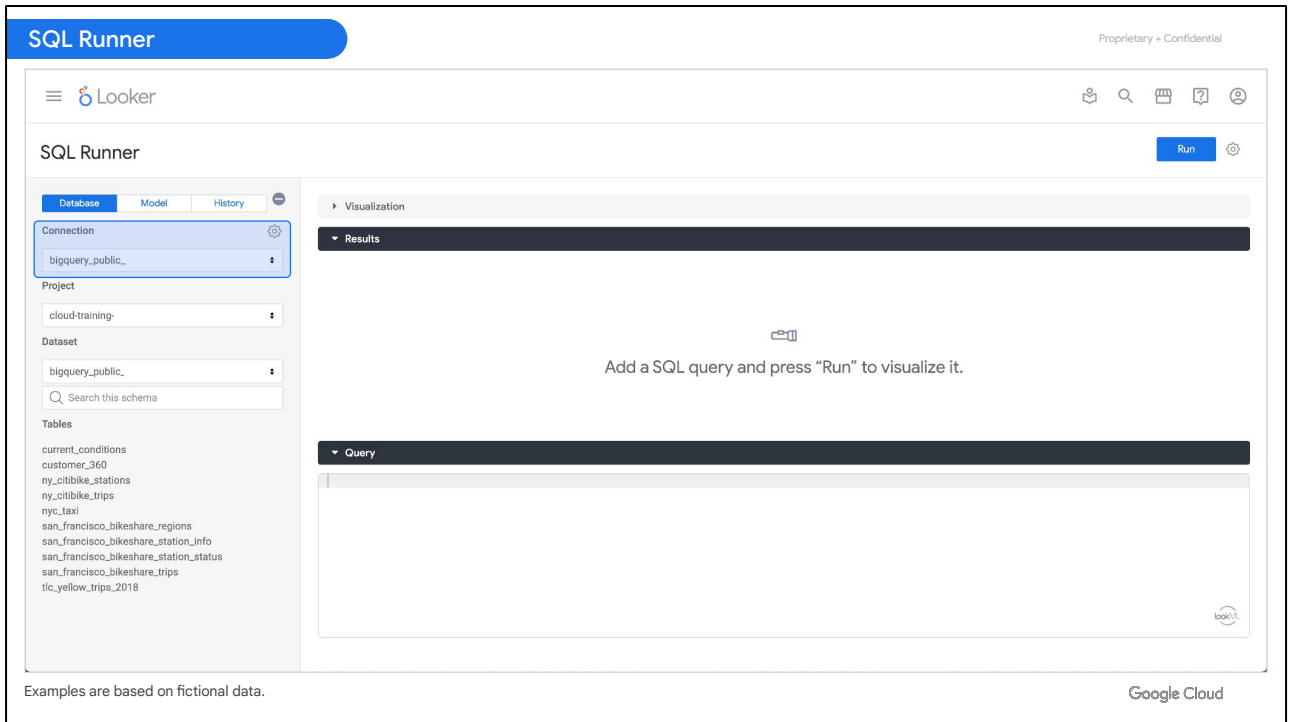
Admins can find setup and configuration details in the [Database configuration details](#) section of Looker's dialects documentation page.



As a LookML developer, you can see available database connections in your organization's Looker instance by clicking on the **Develop** menu in left-side navigation panel of the Looker UI...



... then, selecting **SQL Runner**.



SQL Runner is a console or portal to the databases that have been connected to your organization's Looker instance. You can see all of the database connection options under **Connection**.

The screenshot displays the Looker interface. On the left is a navigation sidebar with options: Explore, Develop, Admin, Shared folders (highlighted), Recently Viewed, Favorites, Boards, Lab exercises, Folders, Blocks, and Applications. At the bottom of the sidebar is a 'Development Mode' toggle. The main content area is titled 'Your organization's folders' and includes a 'New' button and a settings icon. Below this is a 'Folders' section with three folder cards: 'Customer Metrics', 'Example Folder', and 'Human Resources'. A 'Sort by Name' dropdown is positioned to the right. The 'Dashboards' section features a 'Sort by Name' dropdown and a table of dashboard items. The table has columns for 'Name' and 'Favorite'. The items listed are: 'Business Pulse' (64 Views, Created by), 'Business Pulse '21' (34 Views, Created by), and 'Citi bike stations with more than thirty bikes' (13 Views, 1 Favorite, Created by). Each item has a checkbox, a thumbnail, and a 'Favorite' column with a heart icon and a three-dot menu.

Examples are based on fictional data.

Google Cloud

Now, after this brief overview of the Looker business user experience, we hope that you are excited to explore Looker further as a LookML developer.

Introduction to Looker and LookML

- 01 Introduction to the Looker Platform
- 02 Understanding your Users' Experience
- 03 [LookML Project Hierarchy](#)
- 04 Example 1: The Looker Development Environment



To prepare yourself for writing LookML code, there are some key terms and overall project structure that you need to know.

Let's begin with an easy one - what does the "ML" in LookML stand for?

What is LookML?

Proprietary + Confidential

The screenshot displays the Looker web interface in development mode. The main content area shows the LookML code for a model named 'training_ecommerce.model'. The code is as follows:

```
1 connection: "bigquery_public_data_looker"
2
3 # include all the views
4 include: "/views/**/*view"
5 include: "/z_tests/*lkml"
6
7 datagroup: training_ecommerce_default_datagroup {
8   # sql_trigger: SELECT MAX(id) FROM etl_log;;
9   max_cache_age: "1 hour"
10
11
12 persist_with: training_ecommerce_default_datagroup
13
14 label: "E-Commerce Training"
15
16 explore: order_items {
17   join: users {
18     type: left_outer
19     sql_on: ${order_items.user_id} = ${users.id};;
20     relationship: many_to_one
21   }
22
23   join: inventory_items {
24     type: left_outer
25     sql_on: ${order_items.inventory_item_id} = ${inventory_items.id};;
26     relationship: many_to_one
27   }
28
29   join: products {
30     type: left_outer
```

On the right side of the interface, there is a 'Quick Help' section with the following text:

A `model` references a combination of related explores. Unlike other LookML elements, a model is not declared explicitly with the `model` keyword.

```
model: {
  access_grant: identifier
  case_sensitive: yes or no
  connection: "string"
  datagroup: identifier
  explore: identifier
  fiscal_month_offset: number
  include: "string"
  label: possibly-localized-string
  map_layer: identifier
  named_value_format: identifier
  persist_for: "string"
  persist_with: datagroup-ref
  test: identifier
  view: identifier
  week_start_day: monday or ...
}
```

At the bottom of the interface, there is a footer that reads 'Examples are based on fictional data.' and the Google Cloud logo.

LookML stands for Looker Modeling Language. It is Looker's proprietary language that establishes an abstraction layer for SQL. Developers use LookML to tell Looker what data to use from the connected database and how it should interpret that data.

Specifically, LookML acts as the modeling layer between the connected SQL database and your business users. Looker uses the LookML code written by developers to define how business users interact with a connected database and to construct SQL queries against that particular database.

The screenshot displays the Looker development environment. At the top, a blue header contains the text 'SQL and LookML' on the left and 'Proprietary + Confidential' on the right. Below the header, a status bar indicates 'You are in Development Mode' and 'Exit Development Mode'. The main interface is divided into several sections:

- File Browser:** Located on the left, it shows a tree view with folders for 'models', 'views', and 'z_tests'. The 'training_ecommerce.model' file is selected.
- Code Editor:** The central area displays the LookML code for 'training_ecommerce.model'. The code includes a connection to 'bigquery_public_data_looker', includes all views and z-tests, defines a datagroup, and sets up joins and explores for 'order_items', 'inventory_items', and 'products'. A blue rounded rectangle highlights the 'explore' and 'join' sections of the code.
- Quick Help / Metadata:** On the right, there is a 'Quick Help' section with a 'Metadata' tab. It provides a definition of a 'model' and a JSON-like structure for a 'model' object with various metadata fields.

At the bottom left, a note states 'Examples are based on fictional data.' At the bottom right, the 'Google Cloud' logo is visible.

Developers use LookML to define many items from the connected SQL database including data attributes called dimensions, aggregates of dimensions called measures, data relationships such as how to join tables, and custom tables and fields.

A key concept of LookML to remember is: If it's possible in your SQL dialect, it should be possible in Looker. If you can go to your database console and hand-write a **SELECT** statement that does something in the database, you can also code LookML that Looker can use to accomplish the same task.

Hierarchy of LookML objects



For full comprehension of the key LookML terms, developers need to understand where each object fits into the overall hierarchy of a LookML project.

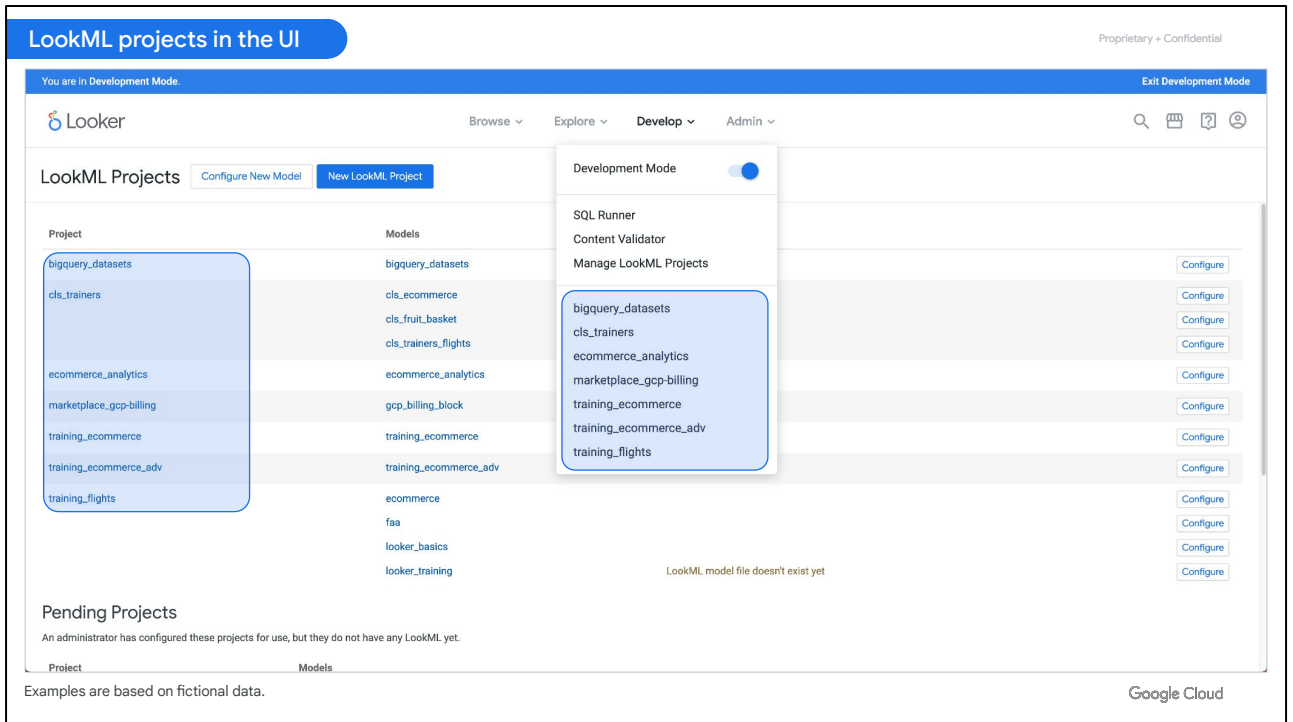
Hierarchy of LookML objects



Project

Git repository

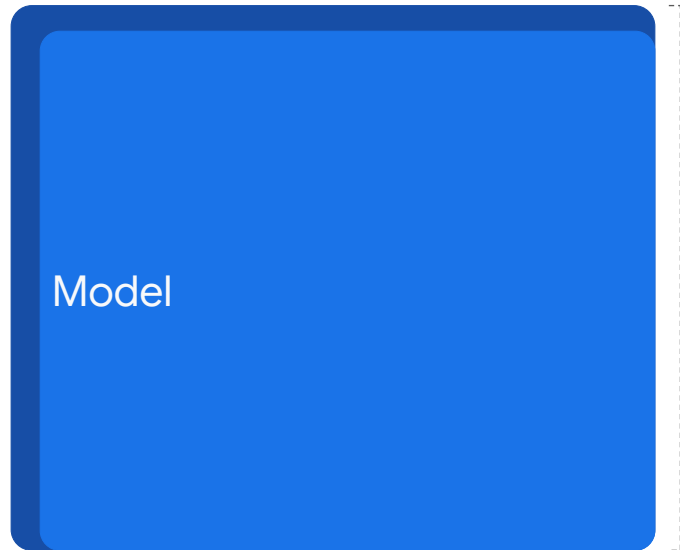
The highest level is the LookML project itself, which is a library of self-contained LookML code. Because Looker uses Git for version control, a best practice is for each project to map one-to-one with a dedicated Git repository.



A project is essentially a library of code for a specific data source or database connection and contains one or more data models. You can think of each project as an semi-independent or mini-instance of Looker, and each project should map one-to-one to a Git repository for version control.

Data that cannot be joined together should be separated into different projects because there is no relation to be made across the data.

Hierarchy of LookML objects



Model

Git repository

The next level in the hierarchy are models. As previously mentioned, a LookML project is composed of one or more models.

A model specifies a database connection and the data views that utilize that connection.

You are in Development Mode. Exit Development Mode

training_ecommerce dev- personal branch Recheck Errors Pull from Production

File Browser

- models
- training_ecommerce.model
- views
- z_tests

```
1 connection: "bigquery_public_data_looker"
2
3 # include all the views
4 include: "/views/**/*.view"
5 include: "/z_tests/*.lkm"
6
7 datagroup: training_ecommerce_default_datagroup {
8   # sql_trigger: SELECT MAX(id) FROM eci_log;;
9   max_cache_age: "1 hour"
10 }
11
12 persist_with: training_ecommerce_default_datagroup
13
14 label: "E-Commerce Training"
15
16 explore: order_items {
17   join: users {
18     type: left_outer
19     sql_on: ${order_items.user_id} = ${users.id} ;;
20     relationship: many_to_one
21   }
22
23   join: inventory_items {
24     type: left_outer
25     sql_on: ${order_items.inventory_item_id} = ${inventory_items.id} ;;
26     relationship: many_to_one
27   }
28
29   join: products {
30     type: left_outer
```

Quick Help Metadata

A **model** references a combination of related explores. Unlike other LookML elements, a model is not declared explicitly with the **model** keyword.

```
model: {
  access_grant: identifier
  case_sensitive: yes or no
  connection: "string"
  datagroup: identifier
  explore: identifier
  fiscal_month_offset: number
  include: "string"
  label: possibly-localized-string
  map_layer: identifier
  named_value_format: identifier
  persist_for: "string"
  persist_with: datagroup-ref
  test: identifier
  view: identifier
  week_start_day: monday or ...
}
```

Shift + ? for keyboard shortcuts

Examples are based on fictional data. Google Cloud

Specifically, a model file is used to define:

- The database connection.
- The view files that are accessible to this model.
- The Explores (which are pre-joined views) and their join logic.

Models can also be used to separate and organize Explores by business area.

Hierarchy of LookML objects



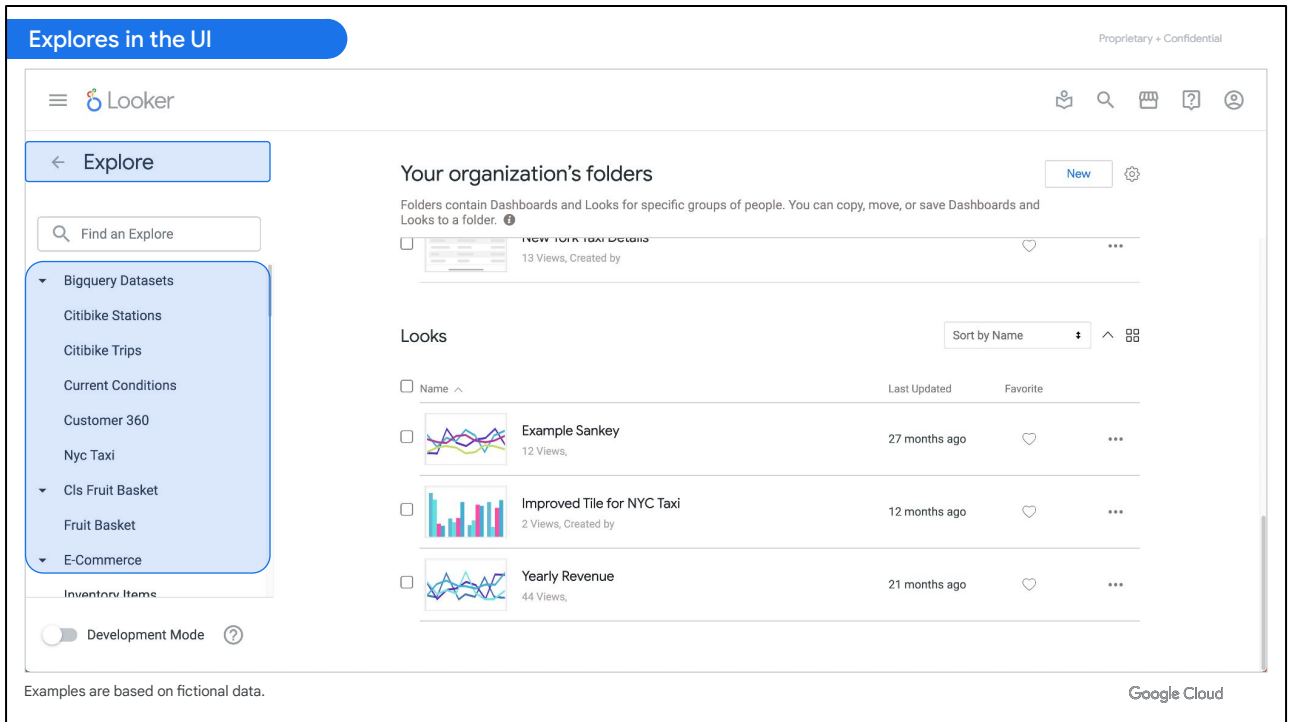
Explore

The diagram illustrates the hierarchy of LookML objects. On the left, a blue rounded rectangle represents an 'Explore' object. To its right, a dashed vertical line indicates the boundary of a 'Git repository'. The 'Explore' object is contained within this repository boundary.

Git repository

Next on the hierarchy are Explores, which are sets of pre-joined views organized by business area defined within the model files.

For example, you might create a model file containing multiple Explores pertaining to customer purchases.

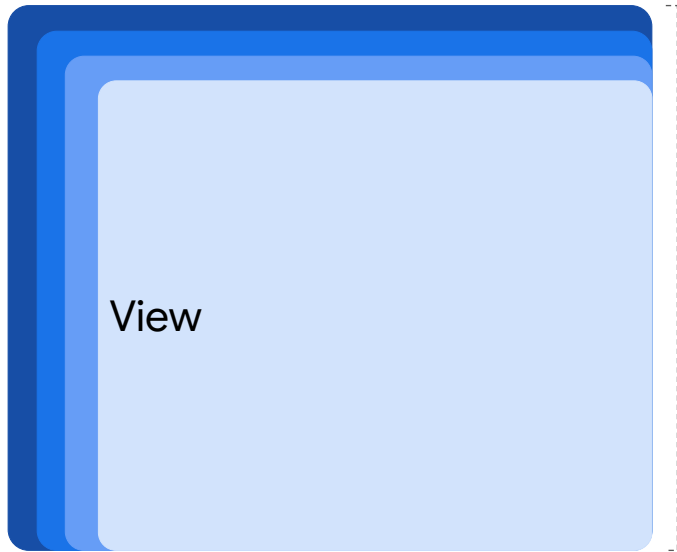


Explores are the central component of Looker that allow business and data analysts to conduct self-serve data exploration, analysis, and visualization.

Within a model file, developers define Explores that join one or more views together to target specific questions that business users may have. Business users then use the predefined Explores to run queries and create reports and visualizations to answer their questions.

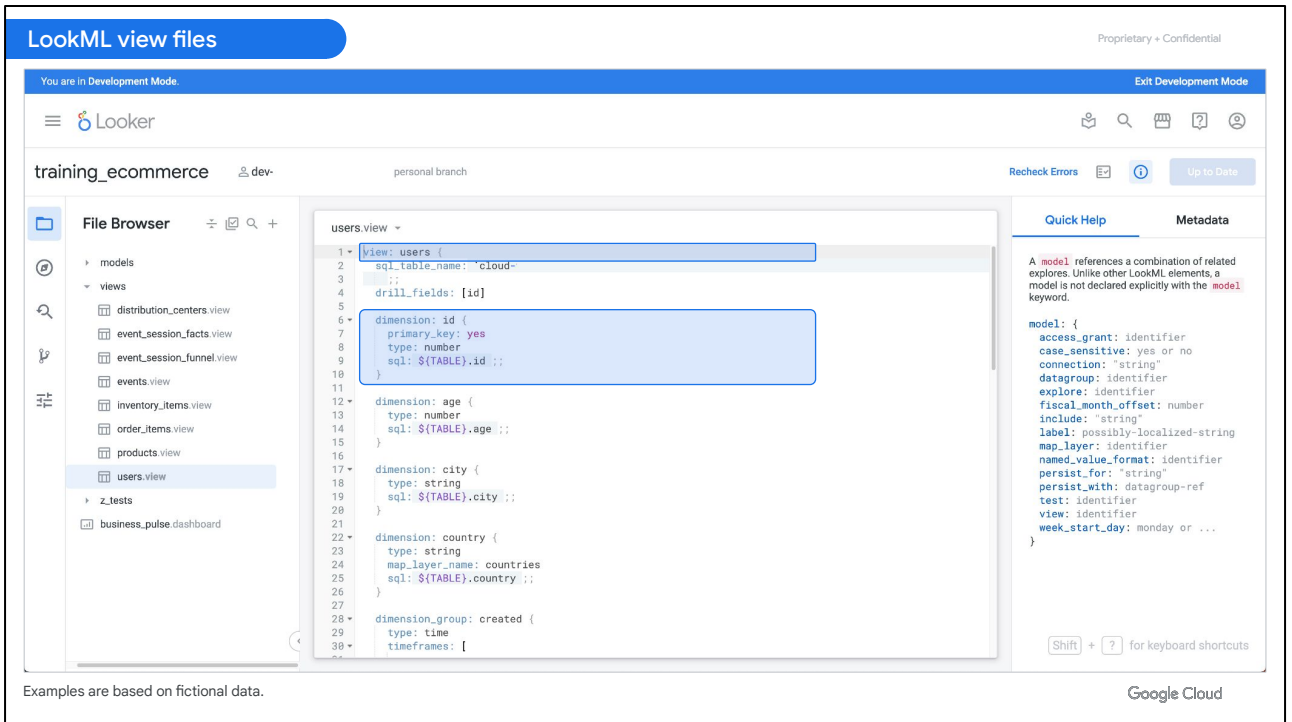
So you can think of an Explore as a predefined set of tables that would frequently be joined for business inquiries and use cases.

Hierarchy of LookML objects



Git repository

Next, the views that are joined in the Explores represent the underlying database tables.



Views are the building blocks of Explores used by business users.

Within view files, developers can define the dimensions (or data attributes) and measures (or aggregations of attributes) to provide to business users in the Explore interface.

The view names that are joined to create an Explore are also the headers that business users see in the Explore. For example, in an Explore called **Order Items**, business users may see the **Users** view, which contains dimensions on users, such as their age and city of residence, and measures, such as their total purchases.

Hierarchy of LookML objects



Last, but not least on the hierarchy, are the dimensions and measures defined in the view files.

The SQL of dimensions

Proprietary + Confidential

The screenshot shows the Looker Explore interface. On the left, the 'Order Items' table is selected, and a list of fields is shown. The main area displays a SQL query with the following structure:

```
SELECT
  users_state,
  users_count
FROM
  (SELECT
    users.state AS users_state,
    COUNT(DISTINCT users.id ) AS users_count,
    COUNT(DISTINCT order_items.order_id ) AS order_items_order_count
  FROM `cloud-training-demos.looker_ecomm.order_items`
  AS order_items
  LEFT JOIN `cloud-training-demos.looker_ecomm.users`
  AS users ON order_items.user_id = users.id
  WHERE (users.country ) = 'UK'
  GROUP BY
    1
  HAVING order_items_order_count > 5) AS t3
ORDER BY
  1 DESC
LIMIT 10
```

Buttons for 'Open in SQL Runner' and 'Explain in SQL Runner' are visible at the bottom right of the SQL editor.

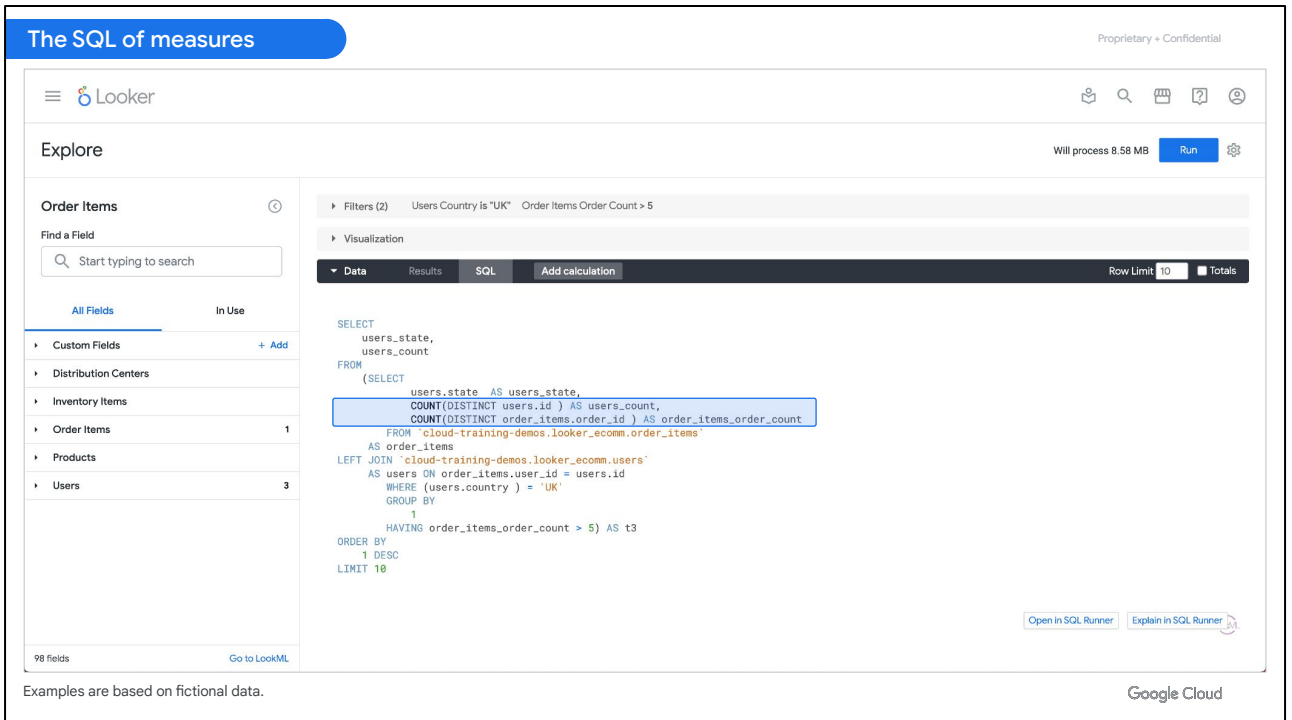
Examples are based on fictional data.

Google Cloud

Dimensions are data attributes and represent the fields or column of a database table.

When the view files are generated from a table by Looker, dimensions are automatically created for any columns that already exist within your database tables. You can also create additional dimensions that would serve as logical representations of table columns.

All dimensions appear in the **SELECT** and **GROUP BY** clause of a SQL statement that is generated by Looker based on the business user's selections in the Explore.



The screenshot shows the Looker 'Explore' interface. On the left, the 'Order Items' dimension is selected. The main area displays a SQL query with the following structure:

```
SELECT
  users.state,
  users_count
FROM
  (SELECT
    users.state AS users_state,
    COUNT(DISTINCT users.id) AS users_count,
    COUNT(DISTINCT order_items.order_id) AS order_items_order_count
  FROM "cloud-training-demos.looker_ecomm.order_items"
  AS order_items
  LEFT JOIN "cloud-training-demos.looker_ecomm.users"
  AS users ON order_items.user_id = users.id
  WHERE (users.country) = 'UK'
  GROUP BY
    1
  HAVING order_items_order_count > 5) AS t3
ORDER BY
  1 DESC
LIMIT 10
```

The query filters for users in the UK and counts distinct order items per user, ordered by count descending. The interface also shows a 'Run' button and a 'Row Limit' of 10.

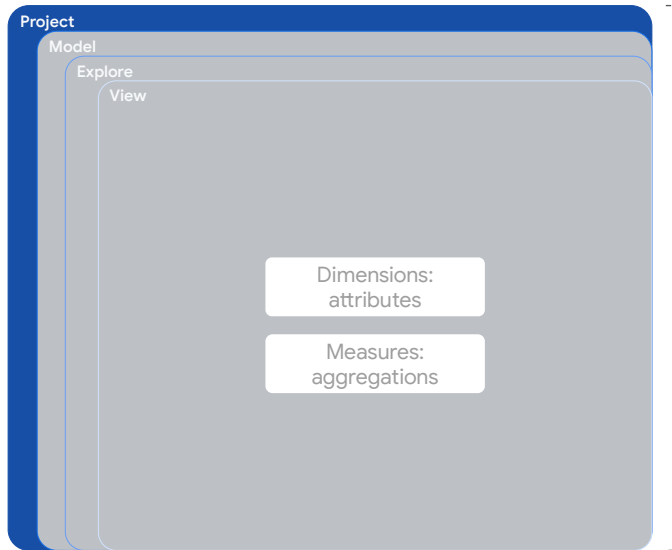
Examples are based on fictional data.

Google Cloud

Measures are aggregates of dimensions and do not live explicitly in your database tables. They must be created using LookML. They aggregate dimensions into values like sums or counts.

Note that they do not appear in the **GROUP BY** statement of the SQL generated by Looker. Instead, they depend on dimensions to determine that grouping, so they appear as aggregate functions in the **SELECT** statement of the SQL query.

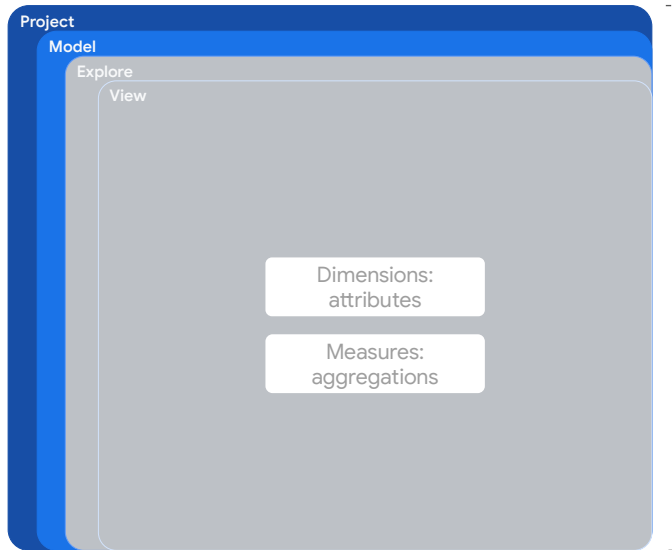
Hierarchy of LookML objects



Git repository

To recap, a LookML project is a library of code that models a data source and should map 1:1 to a Git repository for version control.

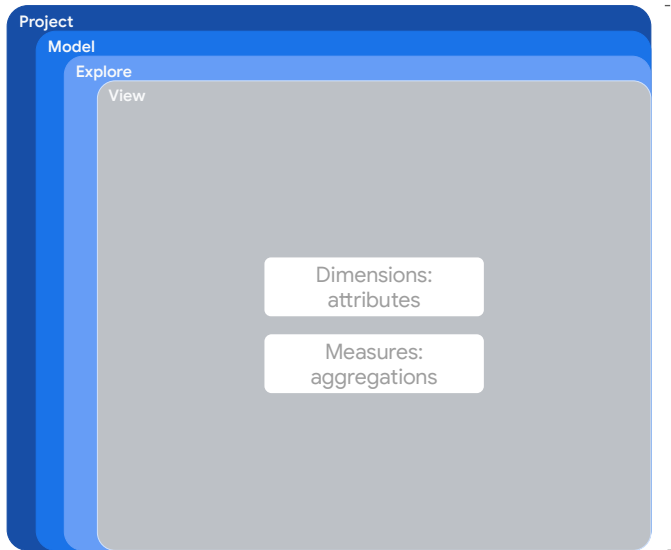
Hierarchy of LookML objects



Git repository

Projects contain model files, which define the Explores that should be packaged together, how those Explores work, and which views are joined in which Explores.

Hierarchy of LookML objects

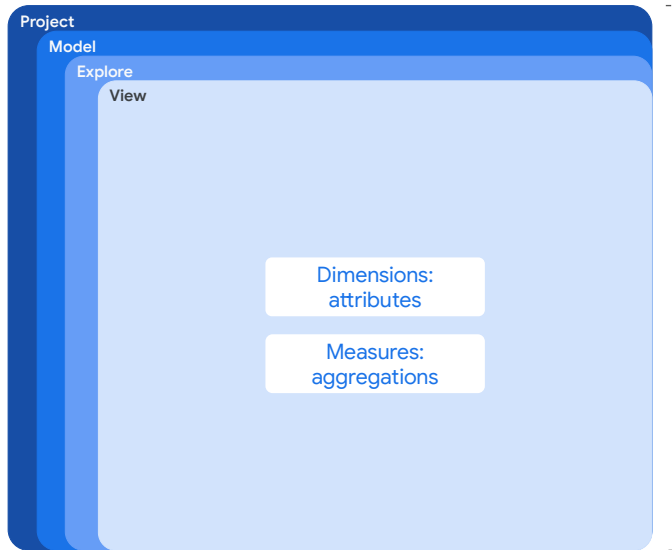


Git repository

View files describe database tables or logical representations of them and are joined together to define Explores in the model files.

Views are then accessed by business users through the Explore to query data and create reports and visualizations.

Hierarchy of LookML objects



Git repository

Dimensions and measures are defined within view files. Dimensions are attributes of data and represent fields or columns in the database tables, while measures are aggregates of dimensions such as a **COUNT** or **SUM**.

After this introduction to the LookML key terms and hierarchy, you are now ready to start exploring your organization's Looker instance as a LookML developer.

Introduction to Looker and LookML

- 01 Introduction to the Looker Platform
- 02 Understanding your Users' Experience
- 03 LookML Project Hierarchy
- 04 **Example 1: The Looker Development Environment**



Looker provides an integrated development environment (IDE) that developers can use to modify existing and to model new dimensions, measures and Explores for business users to employ in their data analysis approaches.

In this section, we will explore the overall Looker development environment and review the key components for your workflow as a LookML developer.

Production Mode vs. Development Mode

Production Mode

- Where business users explore data in Looker.
- The latest production version of the data model is accessed by all users; LookML files are read-only.
- In Git terms, production is the main branch.

Development Mode

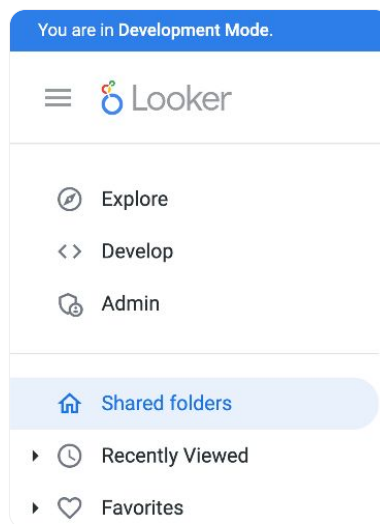
- Where developers make and test changes.
- A separate version of the data model that only the developer can see and edit.
- In Git terms, development is handled by a separate branch.

- In Looker, business users interact with and experience the Looker instance in production mode... while developers make changes and test new features through development mode.
- Production mode uses the latest production version of Looker. Everyone using a Looker instance in production mode accesses its projects, Explores, and content in the same state. LookML project files are read-only in this mode. As a LookML developer, you can use development mode to make changes to projects without affecting anyone else. Development mode accesses a completely separate version of your project files that only you can see and edit.
- If you are familiar with Git, production mode uses the main branch of the Git repository of the LookML project... while development mode uses a separate branch created from the main branch.

After testing, changes made by LookML developers in the separate branch can be merged into production, so that all users of the Looker instance can access the updates.

Development Mode

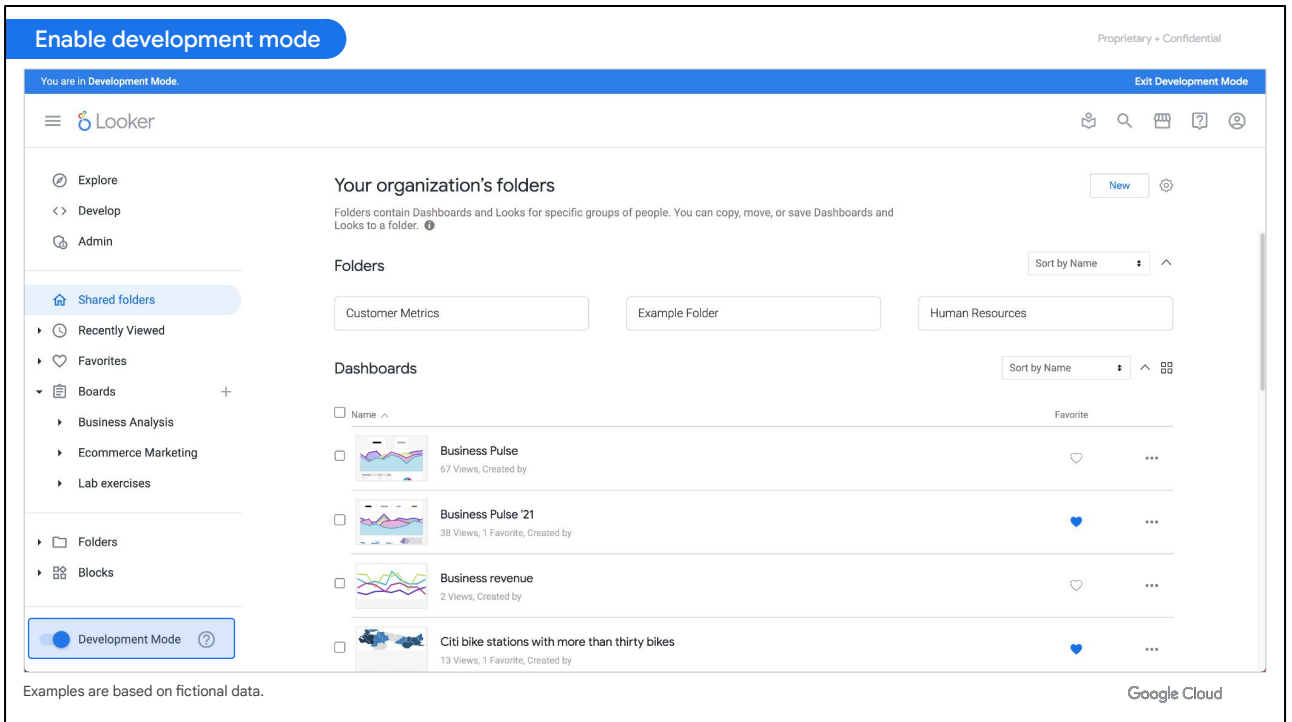
Development Mode allows developers to make and test LookML changes without affecting other users.



By default, LookML projects open in Production Mode. This means that they show the latest version of the project that has been merged to production.

To make changes in your LookML projects, you need to enable Development Mode, which is frequently referred to as “dev mode.”

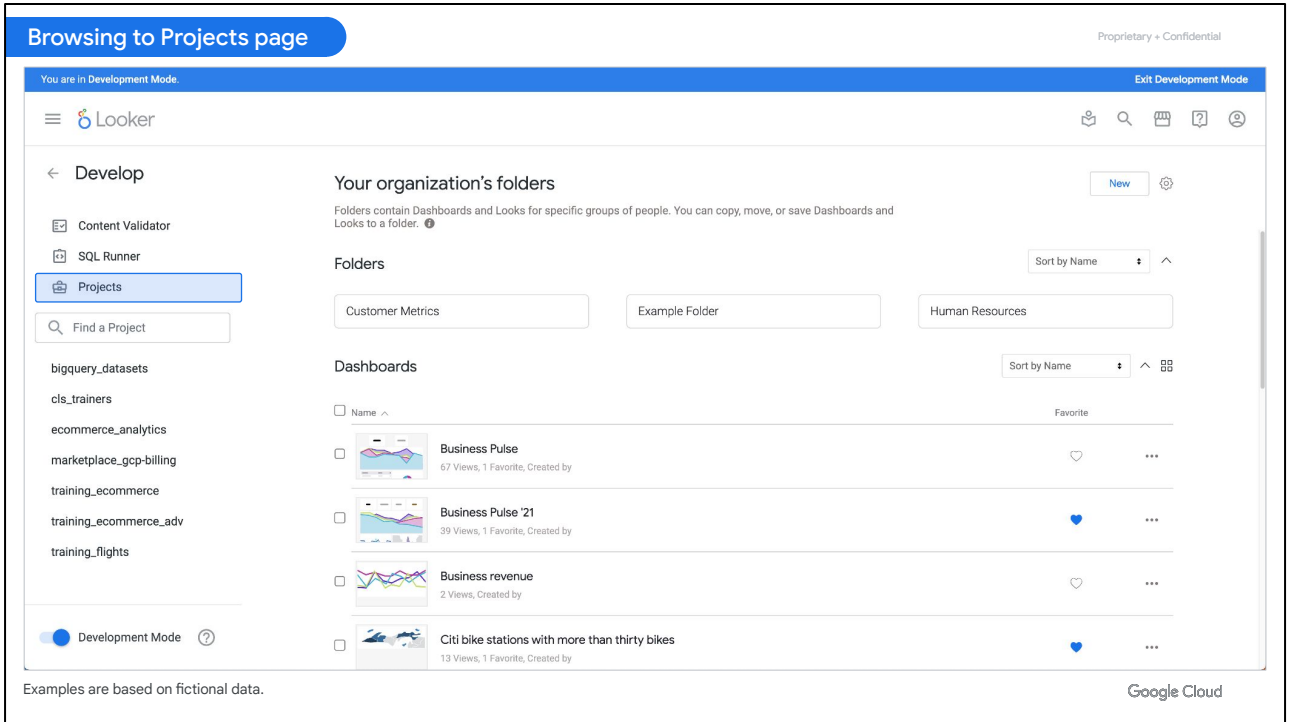
When you are working in development mode, Looker will display a blue banner at top of the Looker User Interface (or UI) with a message that you are in development mode.



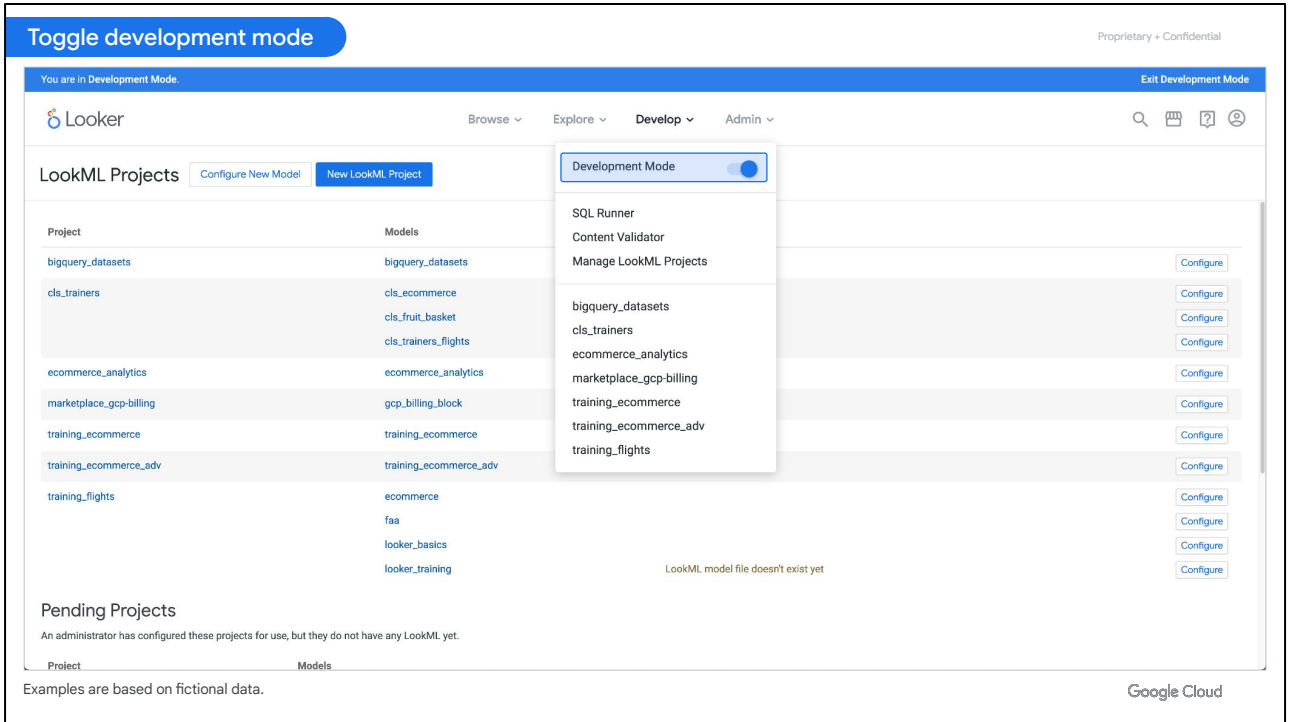
As a LookML developer, you have several options to enter development mode.

One option is to use the toggle button at the bottom left-side of the Looker UI to enter **Development Mode**. You can also use the same toggle button to exit **Development Mode** and return to production mode.

There is also a keyboard shortcut: Control+Shift+D.



Alternatively, at the top left-side of the Looker User Interface, you can click on **Develop** menu to see the options, and then select **Projects**.



Then, once you are on the **Projects** page, you can turn on Development Mode by clicking on the **Develop** option from the top menu bar and using the toggle button for **Development Mode**.

Browsing projects Proprietary + Confidential

You are in Development Mode Exit Development Mode

Looker Browse Explore Develop Admin 🔍 📄 ? 🗑️

LookML Projects Configure New Model New LookML Project

Development Mode

SQL Runner
Content Validator
Manage LookML Projects Configure

Project	Models	
bigquery_datasets	bigquery_datasets	Configure
cls_trainers	cls_ecommerce	Configure
	cls_fruit_basket	Configure
	cls_trainers_flights	Configure
ecommerce_analytics	ecommerce_analytics	Configure
marketplace_gcp-billing	gcp_billing_block	Configure
training_ecommerce	training_ecommerce	Configure
training_ecommerce_adv	training_ecommerce_adv	Configure
training_flights	ecommerce	Configure
	faa	Configure
	looker_basics	Configure
	looker_training	Configure

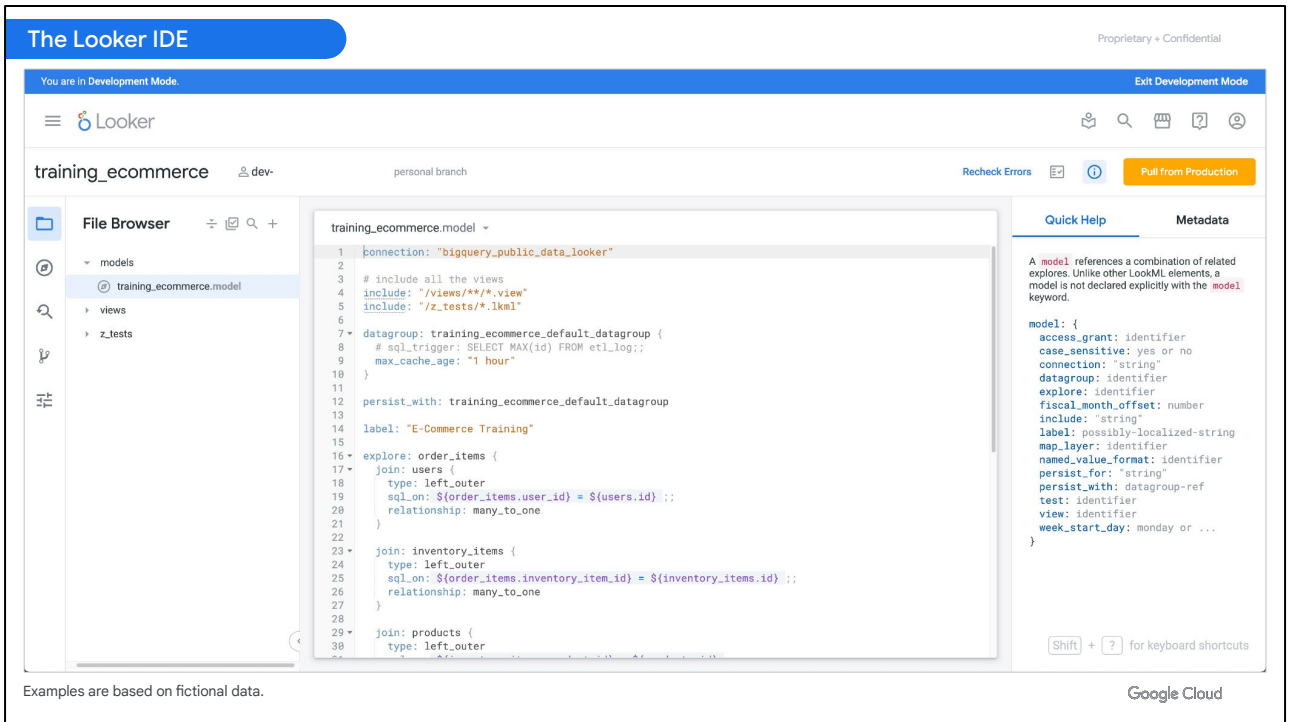
LookML model file doesn't exist yet

Pending Projects
An administrator has configured these projects for use, but they do not have any LookML yet.

Project	Models
---------	--------

Examples are based on fictional data. Google Cloud

From the **Projects** page, you can also click on a specific project name such as **training_ecommerce** to open it within the Looker Integrated development environment (IDE).



An IDE is an industry term for an application in which software engineers or developers can write and test their code.

Looker has its own built-in IDE in which developers can write LookML.

The Looker IDE displays six navigation options on the left-side panel. We'll walk through each option to understand the functionality that it provides for LookML developers.

The screenshot displays the Looker IDE interface in Development Mode. The top navigation bar includes the Looker logo, a search icon, and an 'Exit Development Mode' button. Below this, the current project is identified as 'training_ecommerce' on the 'personal branch'. A 'Reccheck Errors' button and a 'Pull from Production' button are also visible.

The main interface is divided into three sections:

- File Browser (Left):** A tree view showing the project's file structure. It includes folders for 'models' (containing 'training_ecommerce.model') and 'views' (containing various view files like 'distribution_centers.view', 'event_session_facts.view', etc.).
- Model Editor (Center):** A code editor displaying the 'training_ecommerce.model' file. The code defines a datagroup and an explore named 'order_items'.

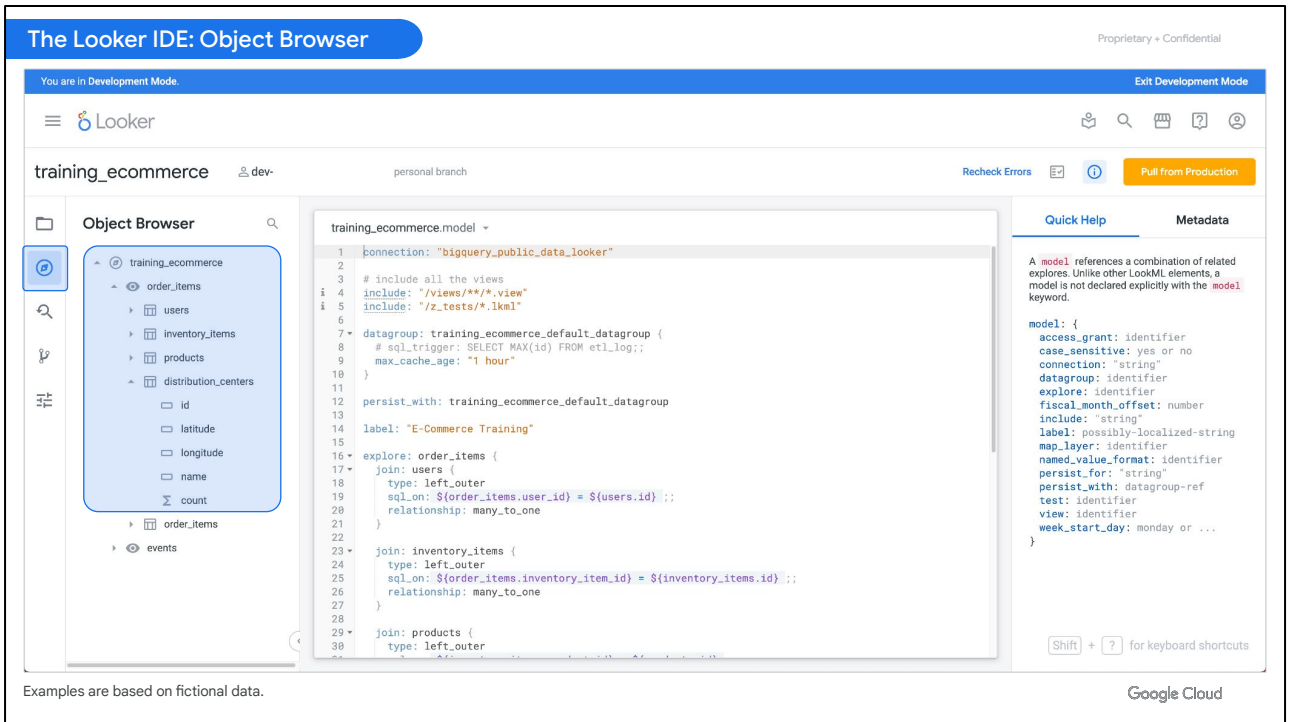
```
1 connection: "bigquery_public_data_looker"
2
3 # include all the views
4 include: "/views/**/*.view"
5 include: "/z_tests/*.lkm1"
6
7 datagroup: training_ecommerce_default_datagroup {
8   # sql_trigger: SELECT MAX(id) FROM etl_log;;
9   max_cache_age: "1 hour"
10 }
11
12 persist_with: training_ecommerce_default_datagroup
13
14 label: "E-Commerce Training"
15
16 explore: order_items {
17   join: users {
18     type: left_outer
19     sql_on: ${order_items.user_id} = ${users.id} ;;
20     relationship: many_to_one
21   }
22
23   join: inventory_items {
24     type: left_outer
25     sql_on: ${order_items.inventory_item_id} = ${inventory_items.id} ;;
26     relationship: many_to_one
27   }
28
29   join: products {
30     type: left_outer
```
- Quick Help / Metadata (Right):** A panel providing documentation for the 'model' keyword. It explains that a 'model' references a combination of related explores and that a model is not declared explicitly with the 'model' keyword. It also lists various metadata fields like 'access_grant', 'case_sensitive', 'connection', 'datagroup', 'explore', 'fiscal_month_offset', 'include', 'label', 'map_layer', 'named_value_format', 'persist_for', 'persist_with', 'test', 'view', and 'week_start_day'.

At the bottom left, a note states: "Examples are based on fictional data." At the bottom right, the Google Cloud logo is present.

The first option, represented by the folder icon, is the file browser. This displays all the LookML project files in the folder hierarchy.

Depending on the complexity of the project, you could see dozens or even hundreds of files, within many folders and levels of subfolders.

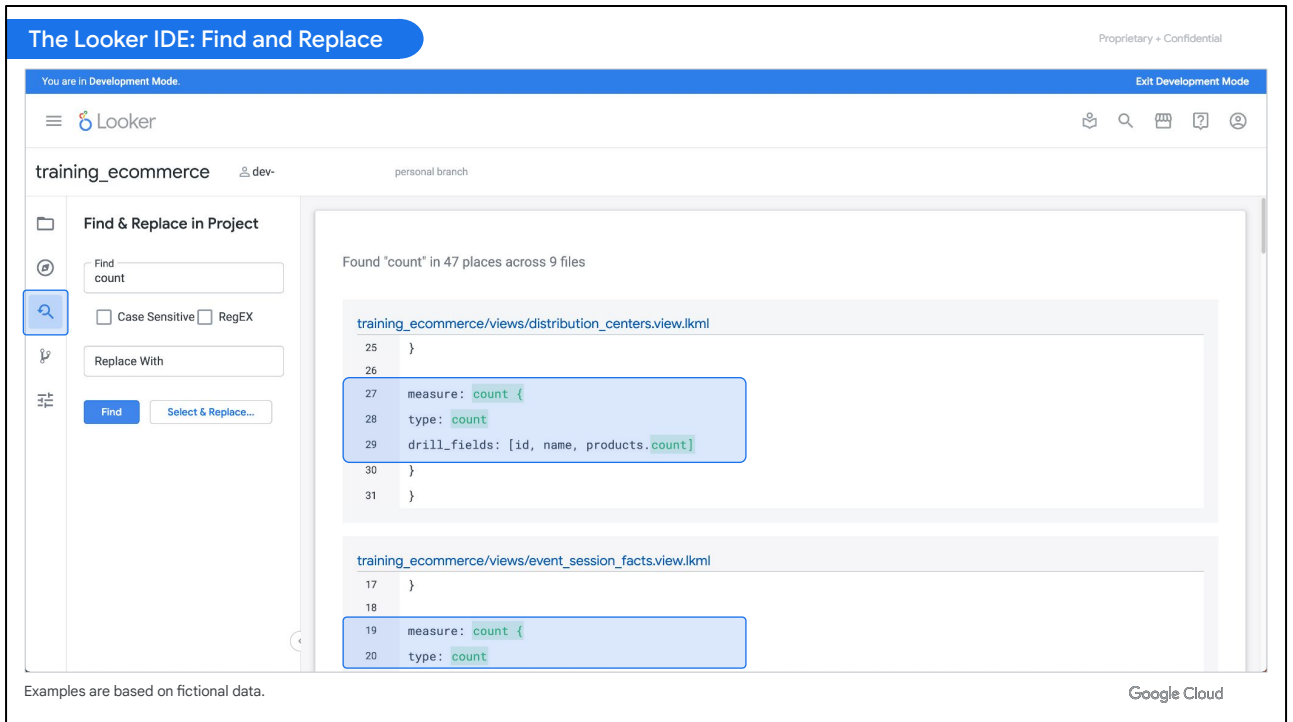
At your company, with your own development team, you can decide on any kind of folder structure that would be the most intuitive for your workflows.



The second option, represented by the compass icon, is the object browser. The object browser organizes project files by LookML object type in the project.

Each model can be expanded to show the Explores within. Furthermore, each Explore can be expanded to reveal the views that are joined together, and each view can be expanded to display the dimensions and measures.

This is really useful when you need to find a specific field or view being used within an Explore, especially if you have many similarly named objects in the model. You may also need to “visualize,” in a sense, in how many Explores the same view is being used, or which models have the same or similar Explores.



Next is the find-and-replace option. You can search for a specific word like “count” and see where and how many times this term appears throughout all the files in your entire project.

You can also batch-replace all instances of a text string with something else.

The screenshot displays the Looker IDE interface. At the top, a blue banner reads "You are in Development Mode" and "Exit Development Mode". Below this, the Looker logo and navigation icons are visible. The main workspace shows a file named "training_ecommerce.model" with a code editor containing LookML. The left sidebar is titled "Git Actions" and lists several options: "Current Branch" (set to "dev-"), "Commit History", "Commit", "Push Changes to Remote", "Pull from...", "Deploy to Production", "Test Git Connection", and "View Project on GitHub". The "Deploy to Production" option is highlighted with a blue box. The right sidebar shows "Quick Help" and "Metadata" sections. The "Metadata" section contains a definition of a "model" and a "model:" block. At the bottom left, a note states "Examples are based on fictional data." and the Google Cloud logo is at the bottom right.

```
1 connection: "bigquery_public_data_looker"
2
3 # include all the views
4 include: "/views/**/*view"
5 include: "/z_tests/*lkml"
6
7 datagroup: training_ecommerce_default_datagroup {
8   # sql_trigger: SELECT MAX(id) FROM etl_log;;
9   max_cache_age: "1 hour"
10 }
11
12 persist_with: training_ecommerce_default_datagroup
13
14 label: "E-Commerce Training"
15
16 explore: order_items {
17   join: users {
18     type: left_outer
19     sql_on: ${order_items.user_id} = ${users.id};;
20     relationship: many_to_one
21   }
22
23   join: inventory_items {
24     type: left_outer
25     sql_on: ${order_items.inventory_item_id} = ${inventory_items.id};;
26     relationship: many_to_one
27   }
28
29   join: products {
30     type: left_outer
```

model: {
 access_grant: identifier
 case_sensitive: yes or no
 connection: "string"
 datagroup: identifier
 explore: identifier
 fiscal_month_offset: number
 include: "string"
 label: possibly-localized-string
 map_layer: identifier
 named_value_format: identifier
 persist_for: "string"
 persist_with: datagroup-ref
 test: identifier
 view: identifier
 week_start_day: monday or ...
}

Shift + ? for keyboard shortcuts

The fourth option in the IDE shows the available Git actions.

These options allow you to switch between Git branches, view past commits by yourself and fellow developers, view the project on GitHub, or whichever Git provider you use at your company, and more.

The screenshot shows the Looker IDE interface. A 'Commit History' pop-up window is open, displaying a table of commit records. The table has columns for Description, Status, Tags, and SHA. The background shows the IDE's sidebar with 'Git Actions' and 'Commit History' options.

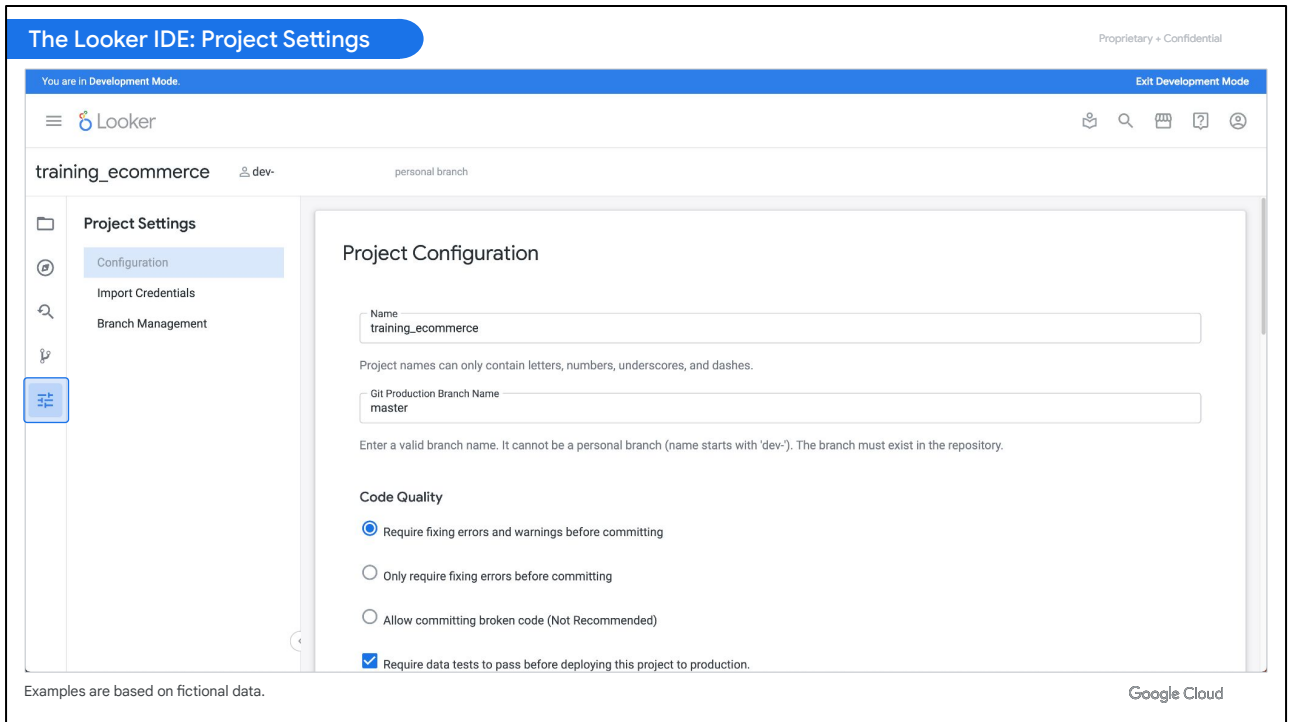
Description	Status	Tags	SHA
removed inventory_items Explore authored 2 years ago			e51a4c9
update DB table reference authored 2 years ago			12b21bf
fixed DB references, added tests authored 2 years ago			7278e42
baseline project for Qwiklabs labs authored 2 years ago			d8329cf
model updates authored 2 years ago			f913637
Merge branch 'master' of git@github.com:ll... authored 2 years ago			69468c8
initial creation authored 2 years ago			8fb66e9
Initial commit authored 2 years ago			c7d5909
Initial commit authored 2 years ago			79faad5

Examples are based on fictional data.

Google Cloud

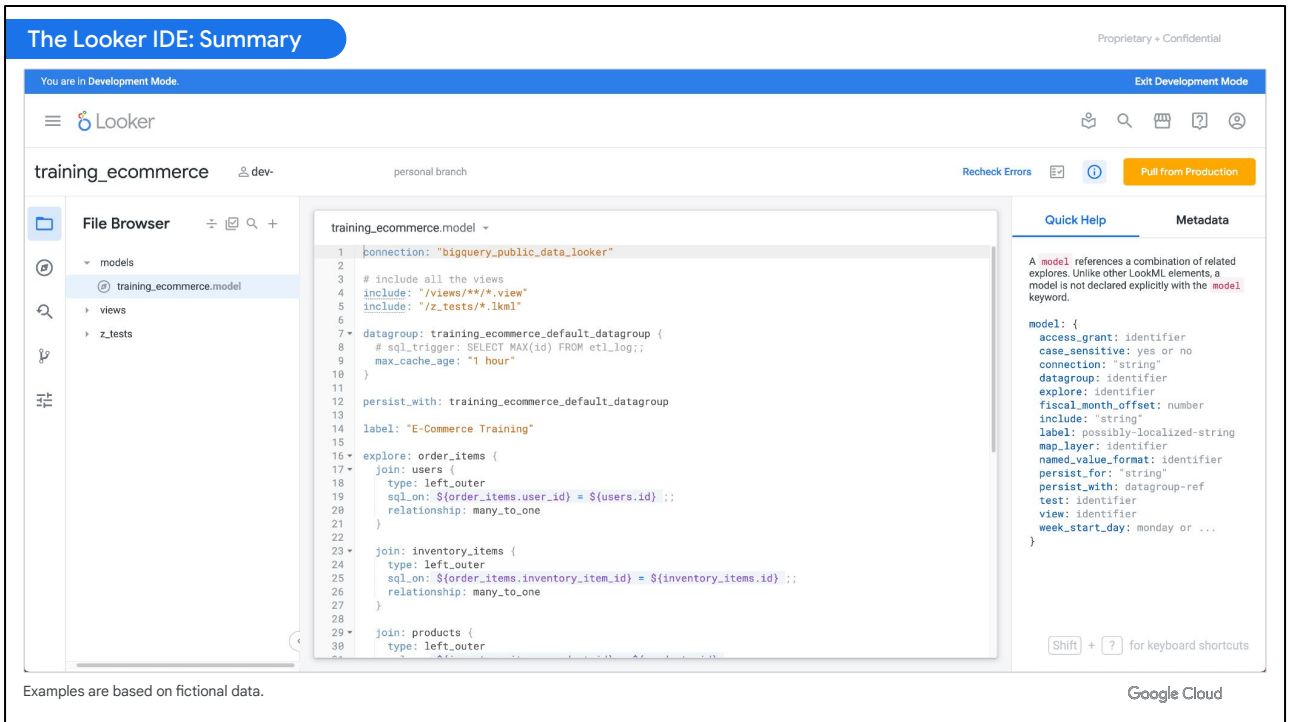
Inside of Git Actions is the Commit History pop-up.

This page lets you see the recent changes that have been deployed to the production environment. Additional details about the changes, such as the user and date associated with the change, are also available on this page.



The last option displays the project settings. As a LookML developer, you can see the configurations, but only a Looker admin can change them.

For example, admins can specify what is or isn't required to commit code, enable pull requests if your team prefers to do code reviews before deploying to production, change the Git connection, and more.

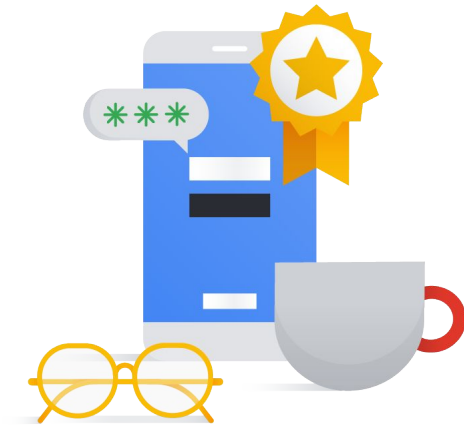


In summary, as a LookML developer, you can make and test changes in your organization's Looker instance in development mode, without impacting the production environment.

When working in your organization's Looker instance, be mindful of whether you are currently in production mode or development mode. The Looker IDE and Explore menus will offer a few different features and options in production vs. development mode. Depending on your permissions and local code differences, you may even see different lists of projects, project files, and Explores.

With this understanding of the differences between production and development mode, we hope that you have fun exploring the LookML projects in your organization's Looker instance!

Let's pause
for a quiz



Let's pause for a quiz.

Summary

Introduction to the Looker Platform

Understanding your users' experience

LookML project hierarchy

The Looker development environment

In this module, we started with an overview of the Looker platform and architecture to understand how Looker can play a key role in your organization's data workflows.

Then, we completed a walkthrough of the Looker user interface to help you understand how your business users leverage Looker to explore and analyze data. Next, we reviewed the LookML project hierarchy and discussed the role that each key component plays in the overall LookML structure.

Last, we explored the Looker development environment and reviewed its key features for your workflow as a LookML developer.

In the next module, we will expand on these concepts to begin writing LookML code to model dimensions and measures, the key data components used by business users to analyze and visualize data.

See you there!

