LookML Developer II Webinar

Looker Training
2021

Google Cloud

Good morning, I’m Emma

Technical Writer

Looker Documentation

Santa Cruz, CA

Agenda

Looker Developer Bootcamp

01	Customizing Looker with Liquid
02	Templated Filters and Parameters
03	Caching & Datagroups
04	Derived Tables

Examples are based on hypothetical data.

Why customize with advanced features?

Out of the box, Looker comes equipped with a wide variety of advanced features. Knowing how to use them will help you:

- Create **one-click workflows** between Looker and other tools
- Design highly-reusable code **tailored to individual users needs**

Why customize with advanced features?

Out of the box, Looker comes equipped with a wide variety of advanced features. Knowing how to use them will help you:

- **Optimize** the query load sent to your database and query runtimes
- Develop rich, **complex analytics** that go beyond basic dimensions and measures

Customizing with Liquid

Creating a dynamic Looker experience

Liquid and Looker are a powerful combination. Together, they can:

- Create one-click workflows from Looker to other tools
- Guide a user to relevant, curated content
- Customize how data is displayed in Looker

What is Liquid?

Open-source, Ruby-based template language created by Shopify.
Used in conjunction with LookML to build more flexible, dynamic code.

Liquid code is denoted by braces {}

What is Liquid?

Liquid falls into three different categories:

- **Objects:** tell Liquid where to show content on a page

```
dimension: product_image {
  sql: ${product_id} ;;
  html: <img src="http://www.acme.com/product_images/{{ value }}.jpg" /> ;;
}
```

- **Tags:** Create the logic and control flow for templates
- **Filters:** Change the output of a Liquid object

Using Liquid in Looker

There are several places in LookML where Liquid can be used:

- the `action` parameter
- the `html` parameter
- the `label` parameter of a field
- the `link` parameter
- parameters that begin with `sql`
    - `sql`
    - `sql_on`
    - `sql_table_name`

Common use cases

Some of the most popular use cases include:

- Creating dynamic links or rendering dynamic images
- Setting up custom drills
- Adding custom conditional formatting
- Integrating templated filters and parameters

Liquid parameters: Referencing LookML objects

Variable	Definition	Example output
`value`	Field value returned by the database query	8521935
`rendered_value`	Field value with Looker's default formatting	$8,521,935.00
`filterable_value`	Field value formatted for use as a filter in a Looker URL	8521935
`link`	URL to Looker's default drill link	/explore/thelook/orders?fields=orders.order_amount&limit=500
`linked_value`	Field value with Looker's default formatting and linking	$8,521,935.00

Custom links

Custom links: Building workflows

Set up custom workflows between Looker content or between Looker and other internal or external resources.

- Link from an executive dashboard to a detail dashboard
- Link from a Look or dashboard to an Explore
- Link from a value in a Look or dashboard to a related page on the external web (i.e., a Salesforce page)

The `link` parameter

Most links are added to dimensions and measures using the [`link`](#) parameter

- **`label`** is the name this link will have in the drill menu
- **`url`** is the link URL and supports full Liquid (but not full HTML)

```
dimension: field_name {
  link: {
    label: "desired label name"
    url: "desired_url"
    icon_url: "url_of_an_image_file"
  }
}
```

- **`icon_url`** is the image URL to be used as an icon for this link

Linking within Looker

Linking within Looker

Links between related dashboards and Explores help users navigate with ease. Pre-populated filters allow you to guide their experience.

Linking to a detail dashboard

Business Overview

1m ago

Order Date: is in the last 90 days

User State: is any value

25,036
Orders

$46.49
Average Sale Price

$67.53
Average Spend per Customer

Revenue by Marketing Channel

$454,834.11
$326,661.49
$209,866.81
$156,383.54
$52,944.10
$240,138.01
$400,000.00
$200,000.00
$0.00
Total Sales
Organic
Search
Facebook
Email
Display
Traffic Source

Customers by Source and Gender

Female
6%
12%
17%
40%
27%

Male
5%
12%
17%
40%
26%

Display
Email
Facebook
Organic
Search

Users by State

Brand Sales

	Brand	Total Nu...	Average Orde...	Total Sales
1	Levi's	2,936	$57.95	$172,346.44
2	Columbia	1,214	$60.91	$74,792.95
3	Allegra K	1,204	$12.53	$15,188.59
4	Dockers	1,163	$41.56	$48,795.85

Linking to a detail dashboard

Business Overview

Order Date: is in the last 90 days

User State: is any value

25,036
Orders

$46.49
Average Sale Price

$67.53
Average Spend per Customer

Revenue by Marketing Channel

Total Sales: $400,000.00, $200,000.00, $0.00

Organic $454,834.11 · Search $326,661.49 · Facebook $209,866.81 · Email $156,383.54 · Display $52,944.10

$240,138.01

Traffic Source

Drill into Columbia

by Product Name

by SKU

by Product ID

Links

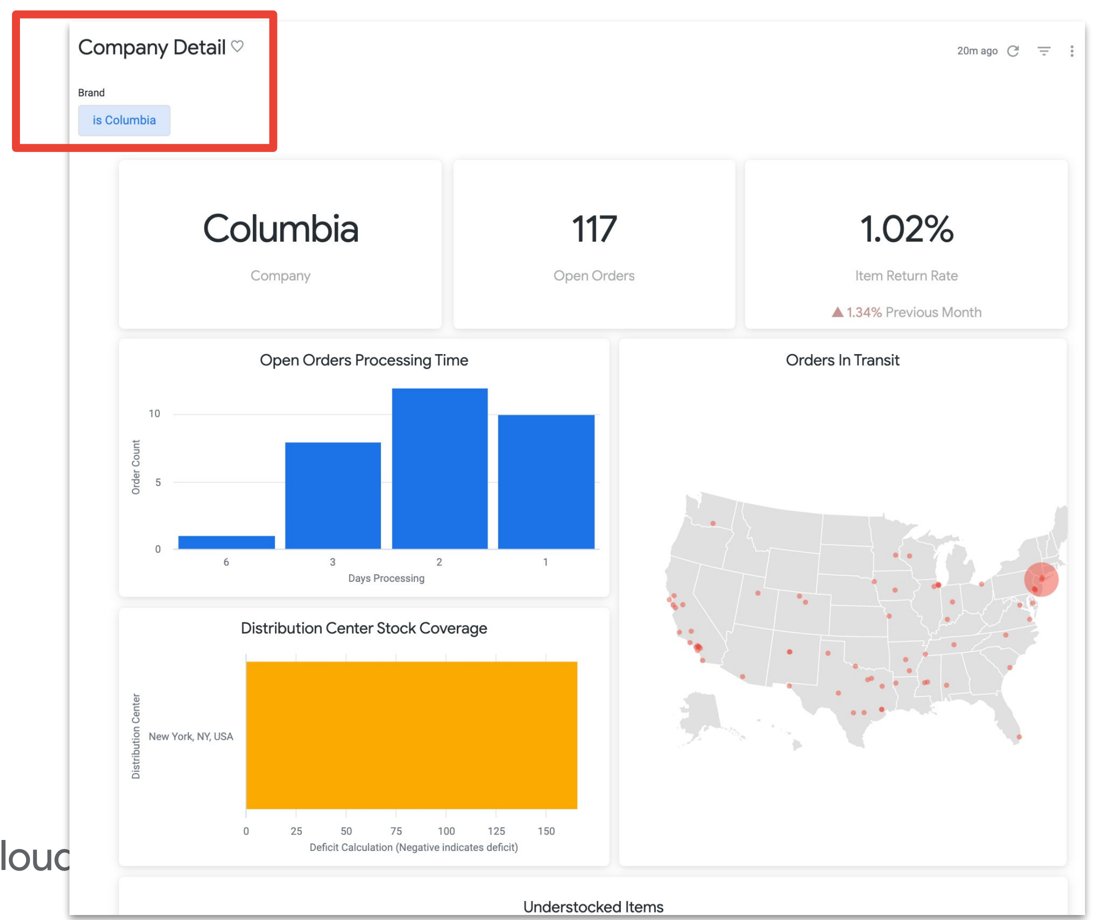
Columbia Analytics Dashboard

Google Search

Users by State

	Brand			Total Sales
1	Levi's			$172,346
2	Columbia	1,214	$60.91	$74,792

Linking to a detail dashboard



Google Cloud

Links to a dashboard

The **Brand** dimension will contain a link to a Looker dashboard that has been filtered for that brand.

```
dimension: brand {
  type: string
  sql: ${TABLE}.brand ;;
}
```

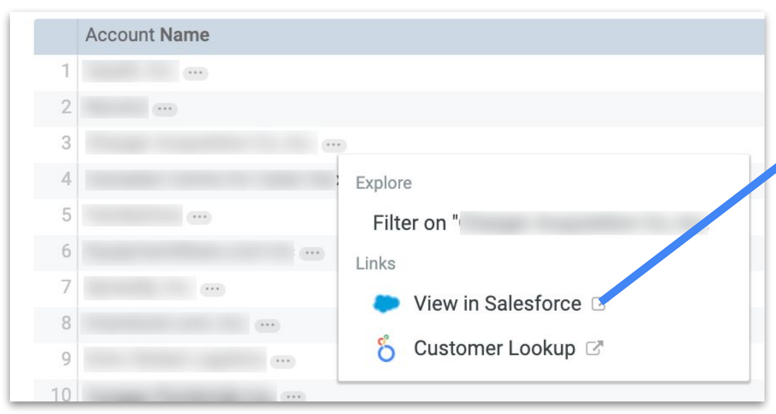
Links to a dashboard

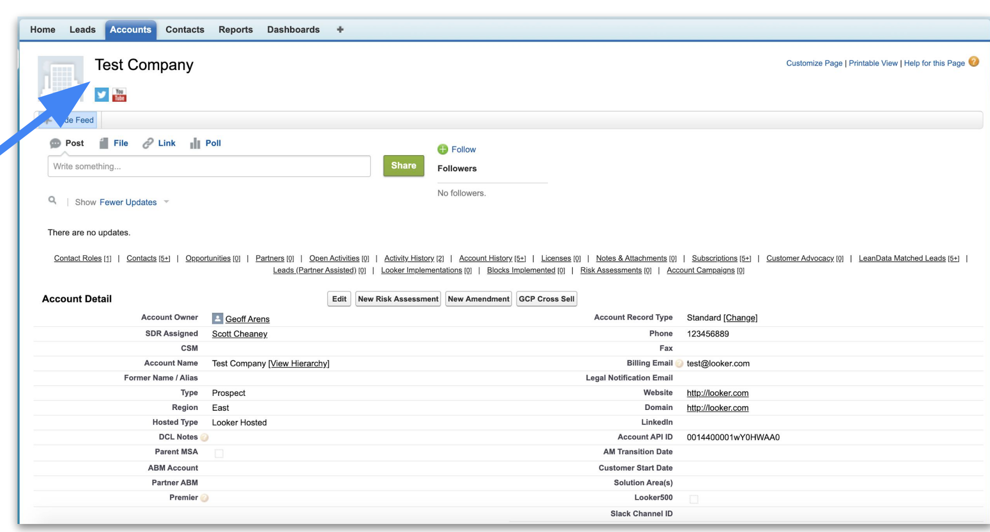
The **Brand** dimension will contain a link to a Looker dashboard that has been filtered for that brand.

```
dimension: brand {
  Type: string
  sql: ${TABLE}.brand ;;
    link: {
      label: "{{value}} Analytics Dashboard"
      url: "/dashboards/24?Brand={{ value | encode_uri }}"
      icon_url: "http://www.looker.com/favicon.ico"
    }
}
```

Linking outside of Looker

Linking to sources outside of Looker can help build workflows for users. Clicking into a dimension could create a JIRA ticket, run a Google search, or open the corresponding page in Salesforce.





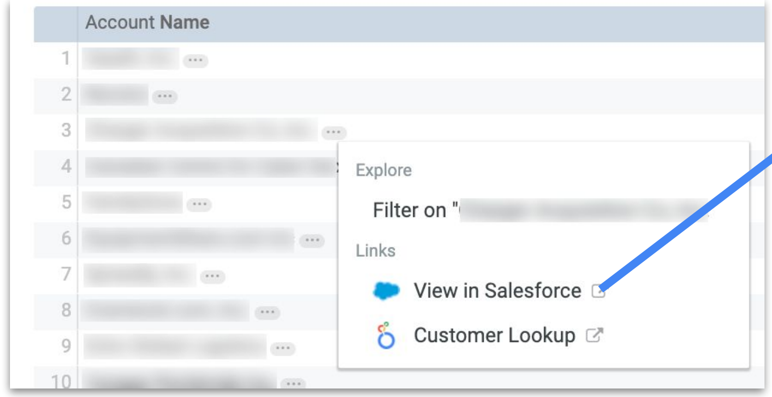
The `html` parameter

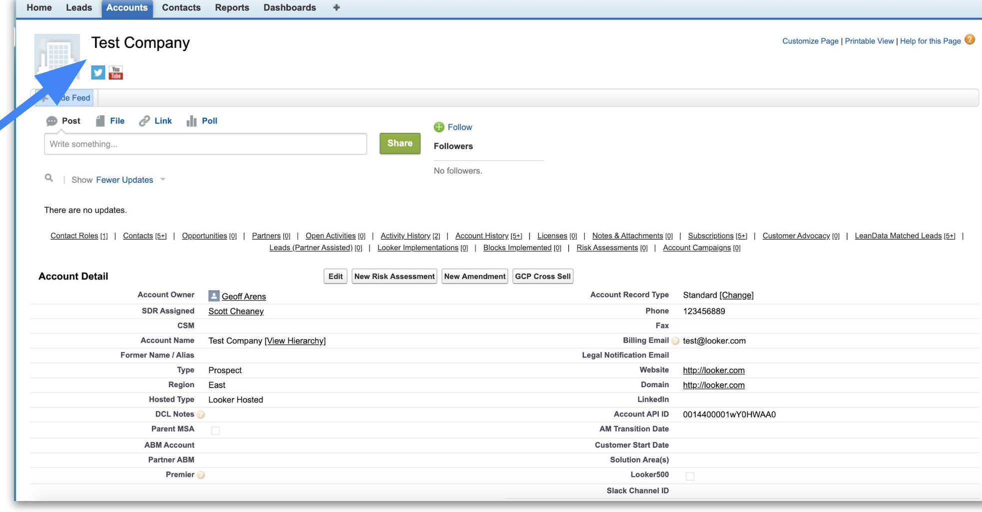
For even more customized drilling and linking, use the `html` parameter

- The dimension value will be shown in Looker and will also be a hyperlink
- Clicking the value will take a user to the specified link within the HTML
- Additional adjustments can be made to customize the user experience

Custom link to Salesforce

```
dimension: id {
  sql: ${TABLE}.opportunity_id ;;
  html: <a href="https://na9.salesforce.com/{{ value }}" target="_new">
    <img src="https://www.salesforce.com/favicon.ico" height=16 width=16></a>
;;
}
```





Custom drills

What is drilling?

Drills allow you to go from high-level metrics like sums and counts into the row-level data that goes into those calculations.

This has the benefit of showing general trends, while also allowing deeper dives to find outliers that may not surface at the high-level.

What is drilling?

A great place to start is to talk to your users and ask what questions come up when viewing reports.

- What is the next question they have after seeing a number?
- Do certain visualizations always lead them to ask iterative questions?
- What values do they think might have contributed to a number that they'd like to confirm?

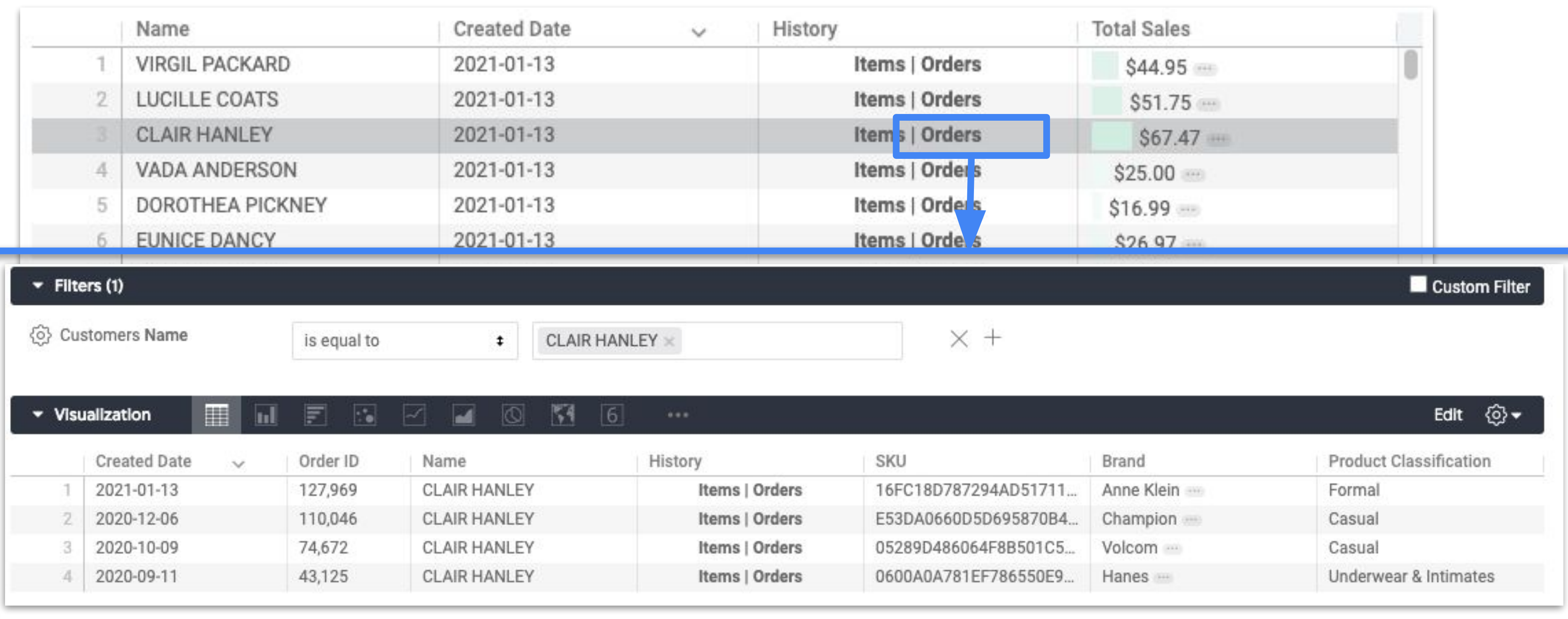
What is drilling?

Designing your drills with these questions in mind will improve your users experience and help them tell better stories from the data.

Custom drilling with HTML

	Name	Created Date	History	Total Sales
1	VIRGIL PACKARD	2021-01-13	**Items | Orders**	$44.95
2	LUCILLE COATS	2021-01-13	**Items | Orders**	$51.75
3	CLAIR HANLEY	2021-01-13	**Items | Orders**	$67.47
4	VADA ANDERSON	2021-01-13	**Items | Orders**	$25.00
5	DOROTHEA PICKNEY	2021-01-13	**Items | Orders**	$16.99
6	EUNICE DANCY	2021-01-13	**Items | Orders**	$26.97
7	CONNIE ABEYTA	2021-01-13	**Items | Orders**	$7.98
8	PATRICIA SOSA	2021-01-13	**Items | Orders**	$24.16

Custom drilling with HTML



Custom drilling with HTML

```
dimension: history {
  sql: ${TABLE}.user_name ;;
  html: <a
href="/explore/thelook/orders?fields=orders.detail*&f[users.id]
={{ id._value }}">Orders</a>
    | <a
href="/explore/thelook/order_items?fields=order_items.detail*&f
[users.id]={{ id._value }}">Items</a>;;
```

Conditional formatting

Conditional formatting

HTML can be used to apply custom formatting to any fields in Looker.

- Add custom colors to dimension labels or header backgrounds
- Include picture or icons as part of displayed values
- Add custom details via a drop down into the cell of a table
- Build a progress bar into the cell of a table that compares the cell value against a goal

Conditional formatting: Example

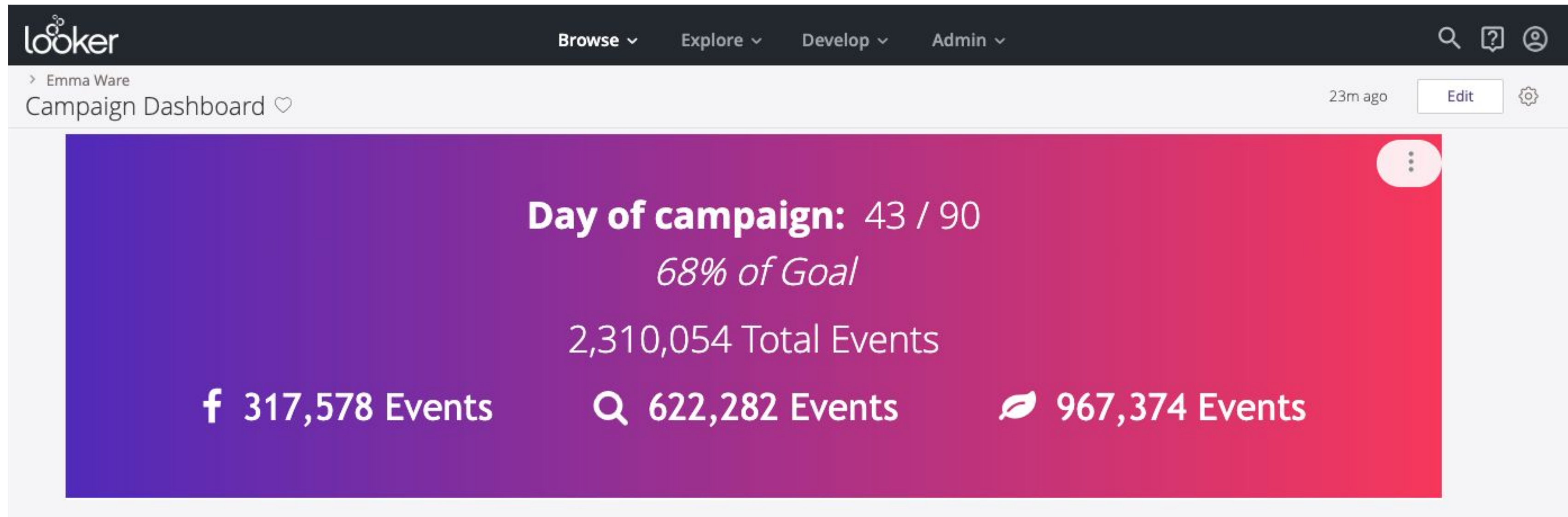
```
dimension: account_health {
  sql: ${TABLE}.account_health ;;
  html: {% if value == 'At Risk' %}
      <b><p style="color: black;
background-color: #dc7350; margin: 0;
border-radius: 5px; text-align:center">{{ value
}}</p></b>

{% elsif value == 'Safe' %}
      <b><p style="color: black;
background-color: #e9b404; margin: 0;
border-radius: 5px; text-align:center">{{ value
}}</p></b>

{% else %}
      <b><p style="color: black;
background-color: #49cec1; margin: 0;
border-radius: 5px; text-align:center">{{ value
}}</p></b>
{% endif %}
    ;;
}
```

Conditional formatting: Advanced example

```
measure: total_gross_margin {
   type: sum
   value_format_name: usd
   sql: ${gross_margin} ;;
html:
<div style="width:100%"> <details>
<summary style="outline:none">{{ total_gross_margin._linked_value }}
</summary> Sale Price: {{ total_sales_price._linked_value }}
<br/>
Inventory Costs: {{ inventory_items.total_cost._linked_value }}
</details>
     </div>;;
}
```

	Order Items **Created Year**	2016
	Order Items **Created Month Num**	Order Items **Total Gross Margin**
1	1	▼ $153,275.68 Sale Price: $290,816.84 Inventory Costs: $137,541.15
2	2	▶ $149,192.79
3	3	▶ $161,288.00
4	4	▶ $162,927.63
5	5	▶ $167,593.41
6	6	▶ $167,510.27

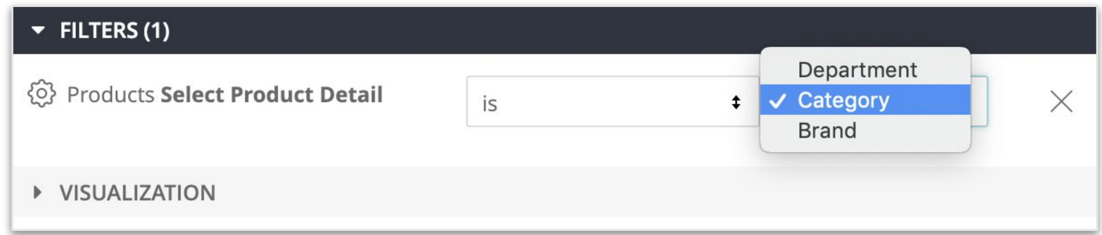
Conditional formatting: Advanced example



Parameters & Templated Filters

Parameters & Templated Filters

Increase interactivity in Explores, Looks, and Dashboards for users



Parameters & Templated Filters

WHAT: User-input values that can be added into a query dynamically

- Parameters: **specific, fixed values** that can be entered by users and then passed directly into a SQL query using liquid
- Templated Filters: user-entered values that are passed into SQL queries using **intelligently written conditional logic**

Parameters & Templated Filters

WHY: Provide **greater flexibility** in how user inputs can influence the SQL queries written

- Dynamic dimensions and measures to consolidate code
- Dynamic derived tables
- Conditionally displayed values

How Do We Do This in Looker?

Step 1	Step 2	Step 3	Step 4
Developer sets up back-end logic	User inputs value into front-end filter	Value is inserted into the SQL & new query is run	Explore is returned

Parameters

Step 1

Developer Sets Up Back-End Logic

Step 1

Parameter field that takes a single user-input value

- type: string / number / unquoted / date, etc.
- allowed_value or suggestions

Used in the syntax

`{% parameter parameter_name %}`

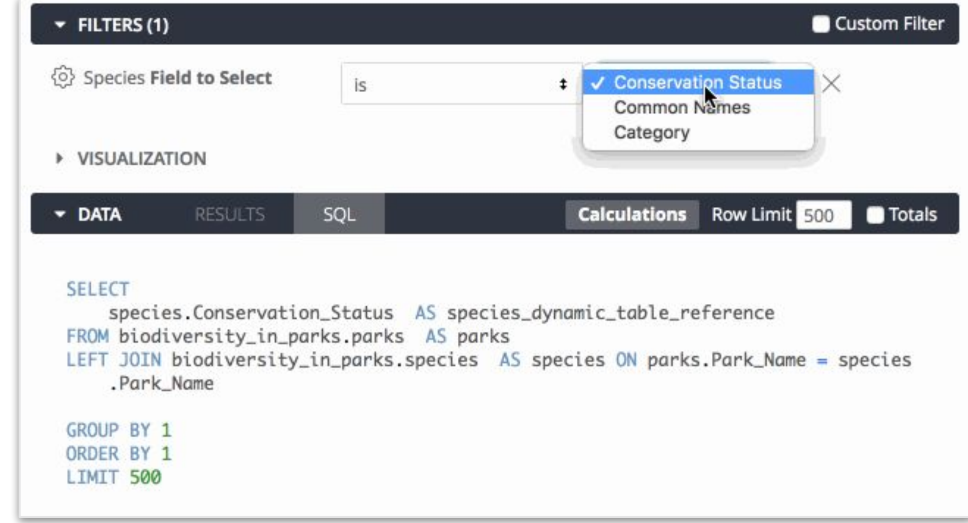
```
parameter: field_to_select {
  type: unquoted
  allowed_value: {
    value: "Category"
    label: "Category"
  }
  allowed_value: {
    value: "Conservation_status"
    label: "Conservation Status"
  }
  allowed_value: {
    value: "Common_names"
    label: "Common Names"
  }
}

dimension: dynamic_column_select {
  type: string
  sql: ${TABLE}.{% parameter field_to_select %} ;;
  label_from_parameter: field_to_select
}
```

Step 2

User Inputs Value into Front-End Filter

Parameters create filter-only field on the front end in an Explore

`allowed_values` appear as drop-down options

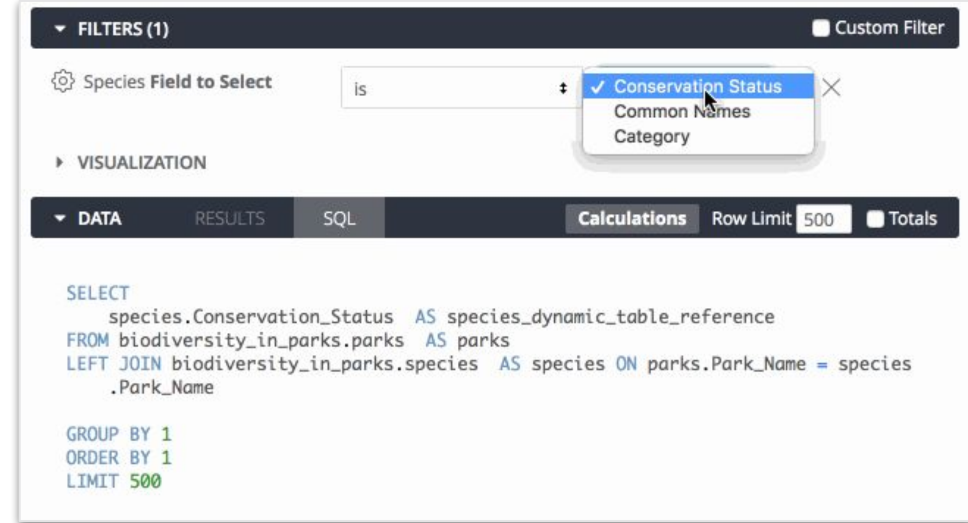


Step 3

Value Inserted into SQL & New Query Run

Step 3

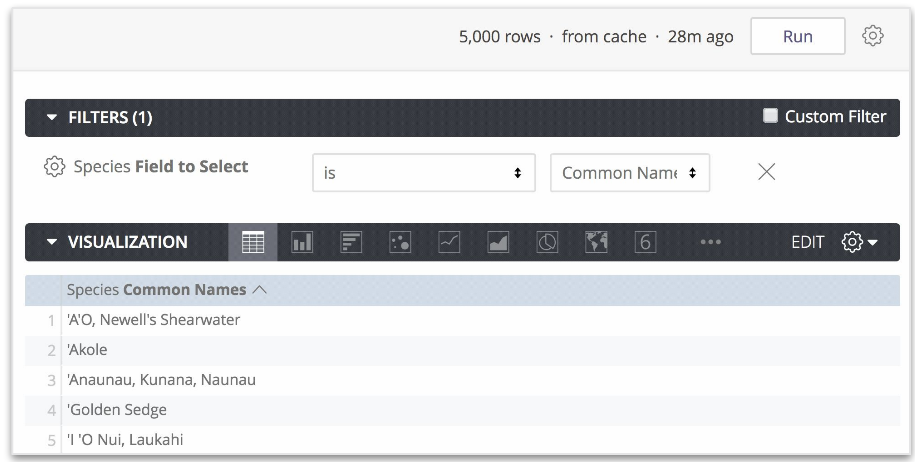
Parameter value is inserted into the `{% parameter parameter_name %}` portion of SQL



Step 4

Explore is Returned

Step 4



Parameters

Example

There's a hierarchy within the product view, and dashboard users need to view dashboard visualizations by any level of this hierarchy. The hierarchy includes the following fields:

- Department (highest level)
- Category
- Brand (lowest level)

Parameters

Set Up Input Logic via LookML

label: what the user will see in the filter options

value: the value that will be inserted into the SQL query

default_value: the value that will be inserted automatically if a user has not yet made a selection

```
parameter: select_product_detail
{
    type: unquoted
    default_value: "department"
    allowed_value: {
     value: "department"
     label: "Department"
    }
    allowed_value: {
     value: "category"
     label: "Category"
    }
    allowed_value: {
     value: "brand"
     label: "Brand"
    }
}
```

Parameters

Dynamic Dimension Creation

Input the parameter value directly into the SQL as the field name:

```
dimension: product_hierarchy {
  label_from_parameter:
   select_product_detail
  type: string
  sql:
${TABLE}.{% parameter select_product_detail %}
   ;;
```

FILTERS (1) Custom Filter

Products **Select Product Detail** is Category

VISUALIZATION

DATA RESULTS SQL Calculations Row Limit 500 Totals

```
SELECT
    products.category AS "products.product_hierarchy"
FROM public.order_items  AS order_items
LEFT JOIN public.inventory_items  AS inventory_items ON order_items.inventory_item_id = inventory_items.id
LEFT JOIN public.products  AS products ON inventory_items.product_id = products.id

GROUP BY 1
ORDER BY 1
LIMIT 500
```

Google Cloud

Parameters

Dynamic Dimension Creation

```
dimension: product_hierarchy {
    label_from_parameter:
             select_product_detail
    type: string
    sql:
    {% if select_product_detail._parameter_value == 'department' %}
    ${department}
    {% elsif select_product_detail._parameter_value == 'category' %}
    ${category}
    {% else %}
    ${brand}
    {% endif %} ;;
}
```

Templated Filters

Step 1

Developer Sets Up Back-End Logic

Step 1

Filter field that utilizes Looker's generated SQL filter logic with string, number, date, etc types

Value generates logic in SQL

Used in the syntax

```
{% condition filter_name %}
field_to_affect {% endcondition %}
```

```
filter: animal_conservation_status {
  type: string
  suggest_dimension: species.conservation_status
  suggest_explore: parks
  }
```

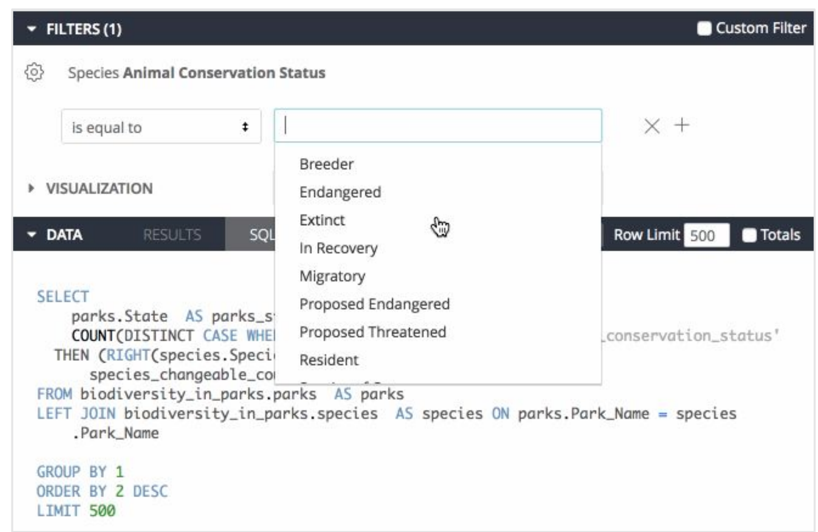
Step 2

User Inputs Value into Front-End Filter

Step 2

A filter-only field is created on the front end in an Explore

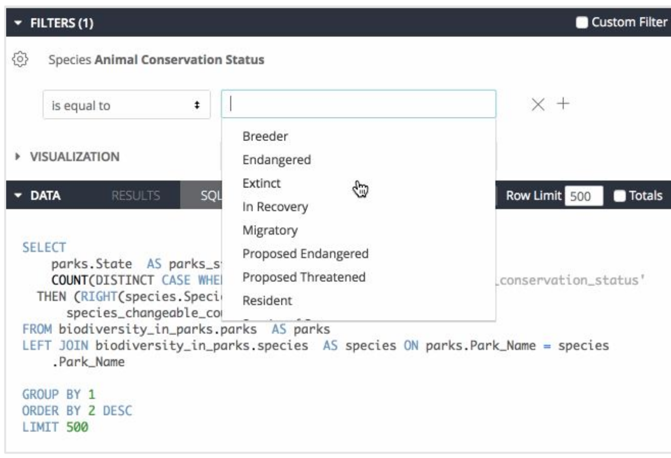
`suggest_dimension` values appear as drop-down options



Step 3

Value Inserted into SQL & New Query Run

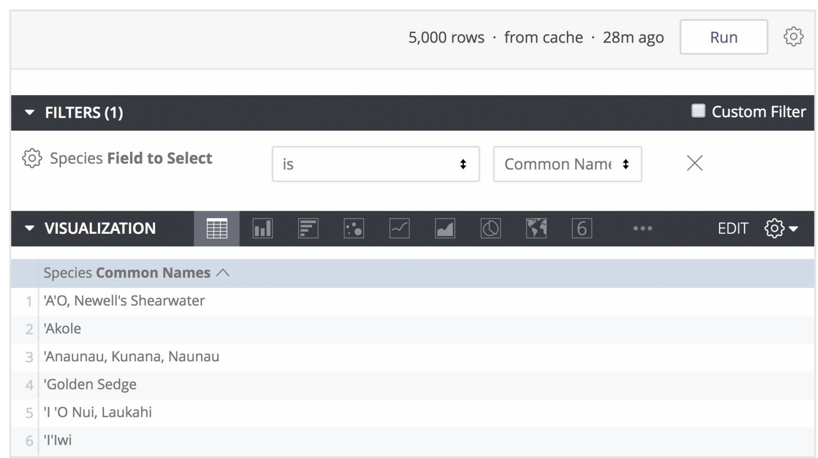
Step 3



Step 4

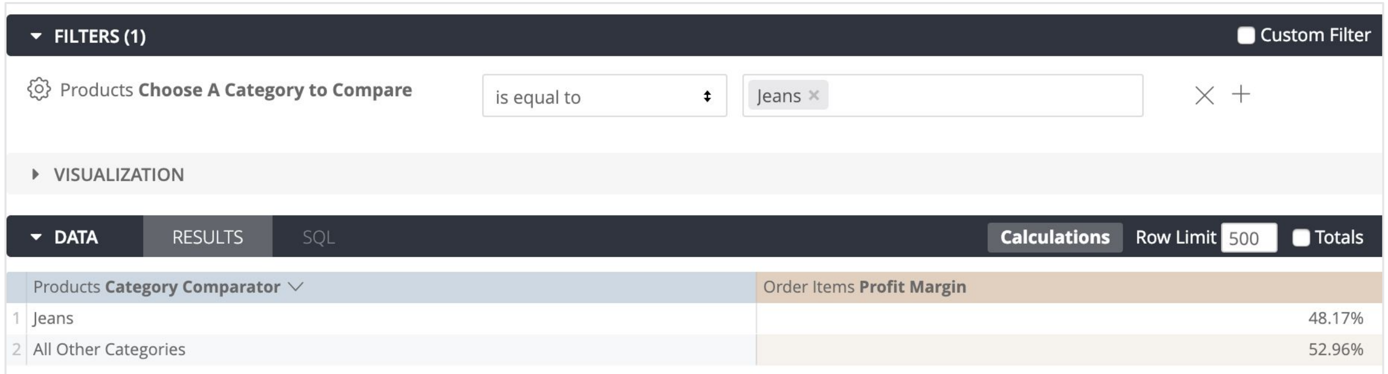
Explore is Returned

Step 4



Templated Filters

How does the profit margin of the Jeans category compare to the profit margin across all other categories?



Templated Filters

Example: LookML Input Logic

Suggest Explore: the Explore that will be queried in order to pull a list of suggested filter values

Suggest Dimension: the dimension that should be used within the suggest Explore for providing a list of suggested filter value

```
filter: choose_a_category_to_compare {
    type: string
    suggest_explore: inventory_items
    suggest_dimension: products.category
  }
```

Templated Filters

Example: Dynamic Dimension

```
dimension: category_comparator {
    type: string
    sql:
        CASE WHEN
            {% condition choose_a_category_to_compare %}
                ${category}
            {% endcondition %}
        THEN ${category}
            ELSE 'All Other Categories'
        END
        ;;
}
```

Templated Filters

FILTERS (1) Custom Filter

Products **Choose A Category to Compare** | is equal to | Jeans

VISUALIZATION

DATA RESULTS SQL | Calculations | Row Limit 500 | Totals

```
SELECT
    CASE WHEN (products.category = 'Jeans') THEN products.category
        ELSE 'All Other Categories'
        END
           AS "products.category_comparator",
    (COALESCE(SUM((order_items.sale_price - inventory_items.cost) ), 0))/NULLIF((COALESCE(SUM(order_items.sale_price ), 0)),
        0)  AS "order_items.profit_margin"
FROM public.order_items  AS order_items
LEFT JOIN public.inventory_items  AS inventory_items ON order_items.inventory_item_id = inventory_items.id
LEFT JOIN public.products  AS products ON inventory_items.product_id = products.id

GROUP BY 1
ORDER BY 1 DESC
LIMIT 500
```

When to Use Parameters vs. Templated Filters

Parameter Fields	Templated Filters
Insert user input directly (or using values you define as allowed values)	Insert values as Looker-generated logical statements

Caching &
Datagroups

Google Cloud

Why cache?

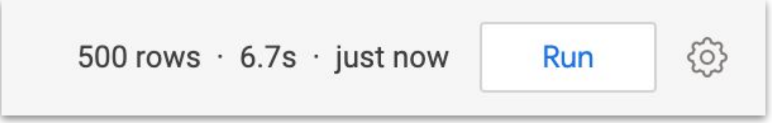
Using cached results of prior queries helps to reduce database load

If you ETL new data into your database every 12 hours, your caching policy in Looker should reflect this

How Caching Works in Looker

A query is run by a user and cached (cache results are stored in an encrypted file on the Looker instance)

How Caching Works in Looker



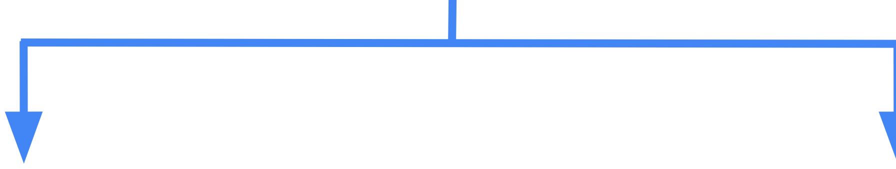
A new query

500 rows · from cache · just now Run

A cached query

How Caching Works in Looker

For any new queries, the cache is checked to see if the same query was previously run before running the query against the database



If the query is not found, Looker runs the query against the database and caches the new result

If the query is found and the results are still valid then Looker uses the cached results

If the query is found and the results are no longer valid, Looker runs the query against the database and caches the new result

Implementing Caching in Looker

These caching policies can then be applied to various Looker objects:

- At the **model** or **Explore** level: use persist_with parameter to specify which Explores use each policy for clearing the query cache
- In a **PDT** definition: use datagroup_trigger to specify which policy to use in rebuilding the PDT
- On **Looks** and **dashboards**: build schedules that trigger based on datagroups to cause content to run and send immediately after the cache has been invalidated, thus warming the cache with the latest results

Datagroups

WHAT: Named **caching policies** within Looker that can be applied to models, Explores, or Persistent Derived Tables

WHY: Integrate Looker more closely with ETL processes or guarantee a refreshed cache

- Define one or more datagroup parameters at the model level
- Different caching policies require separate datagroup definitions

Configuring Datagroups

Caching policy parameters:

- sql_trigger parameter
    - Should be SQL query that returns one row with one column
    - Typically will query a field that serves as a good indicator that the underlying data has been updated, such as a max(date) or will return a specific time of day
- max_cache_age to indicate the longest amount of time in which a query should be cached before being invalidated
- Only one of these parameters is required, but both are recommended

```
datagroup: daily_etl {
  max_cache_age: "24 hours"
  sql_trigger: SELECT max(id) FROM my_tablename ;;
}
```

Applying Datagroups to Query Results

A datagroup's caching policy can be applied to one, some or all Explores in a model.

- As a default for all Explores in a model: use the **persist_with** parameter at the model level and specify the name of the datagroup
- For a specific Explore: use the **persist_with** parameter in that Explore's definition and specify the name of the datagroup
- For a group of Explores: use the **persist_with** parameter in each Explore's definition and specify the name of the same datagroup

Applying Datagroups to Query Results

Datagroups can also be used to add persistence to derived tables, **which will be covered in the next section**

ecommerce_data.model

```
1  connection: "thelook_events"
2  persist_with: order_items
3
```

```
explore: order_items {
  persist_with: order_items
  join: users {
    type: left_outer
    sql_on: ${order_items.user_id} = ${users.id} ;;
    relationship: many_to_one
  }
}
```

Datagroup example

```
datagroup: orders_datagroup {
 sql_trigger: SELECT max(id) FROM my_tablename ;;
 max_cache_age: "24 hours"
 label: "ETL ID added"
 description: "Triggered when new ID is added to ETL log"
}
```

Can be added to a model, explore, etc.

Monitoring Datagroups

Datagroups

Database

Datagroups

Name	Label	Connection	Model	Type	Description	Actions
e_faa_folders_default_datagroup		faa	e_faa_flights			LookML
e_faa_merge_conflict_default_datagroup		faa	e_faa_merge_conflict			LookML
e_flights_datagroup Cache Reset At: 86d ago	ETL ID added	faa			Triggered when new ID is added to ETL log	LookML
orders_datagroup Trigger value: 2020-03-13 21:42:13 +0000 Trigger last checked: 4m ago Cache Reset At: 4m ago		thelook_events	thelook	sql_trigger		LookML
events_information Trigger value: 228037 Trigger last checked: 4m ago Cache Reset At: 23h ago		thelook_events	thelook_events	sql_trigger		LookML

Derived Tables

Going beyond the tables in your database

By default, Looker generates views from tables that already exist in your database.

LookML

order_items.view — Table in your database

```
1  view: order_items {
2    sql_table_name: public.order_items ;;
3
4    dimension: order_item_id {
5      primary_key: yes
6      type: number
7      sql: ${TABLE}.id ;;
8    }
9
```

Generated SQL

DATA RESULTS SQL

```
SELECT
    order_items.order_id  AS "order_items.order_id",
    COALESCE(SUM(order_items.sale_price ), 0) AS "order_item
FROM public.order_items  AS order_items

GROUP BY 1
ORDER BY 2 DESC
LIMIT 500
```

Table in your database

Going beyond the tables in your database

Sometimes in SQL, you need will want to create new tables, or sub-selects. In SQL terms, these can be known as …

- Temporary tables or materialized views
- CTEs
- Subqueries

Multi-step Aggregations

This will fail...

```
SELECT  MAX(SUM(sales ) )
FROM orders
GROUP BY department
```

Fix by aggregating and grouping in multiple steps with a subquery:

```
SELECT  MAX(subquery.total_sales )
FROM
     (SELECT SUM(sales) as total_sales
      FROM orders
      GROUP BY department) AS subquery
```

Example:

Which department has the highest total sales?

1. Find the total sales by department
2. Find the maximum of those totals

Derived Tables Can Help!

You may want more flexibility to **restructure data** and **define complex query logic** to do cool things like...

- **Pre-aggregate** fields to aggregate aggregates
- **Utilize window functions** to sessionalize event data
- **Roll up user data** to the month level to track periods of (in)activity and analyze retention
- **Union** different marketing channels' tables so that KPIs are standardized and aggregated across the channels

What Are Derived Tables?

Manually written query whose result set can be queried like a regular database table

Integrated into Looker as views

Can be joined into Explores just like standard views

They can be ephemeral or written into the database (PDT)

Two Types of Derived Tables

SQL Derived Table

- Easy to learn
- Easy to understand
- Uses complex joins, calculations and functions such as UNION

Native Derived Table

- Maximum code reusability
- Easier to maintain
- Easier to read and understand

SQL Derived Tables

Step 1 Build / Test Query in SQL Runner

You are in **Development Mode**. **Exit Development Mode**

Looker Browse Explore **Develop** Admin

SQL Runner

Run

Database | Model | History

Connection

events_ecommerce

Schema

public

Search this schema

Tables

distribution_centers
etl_jobs
events
foo
inventory_items
order_items
products
users

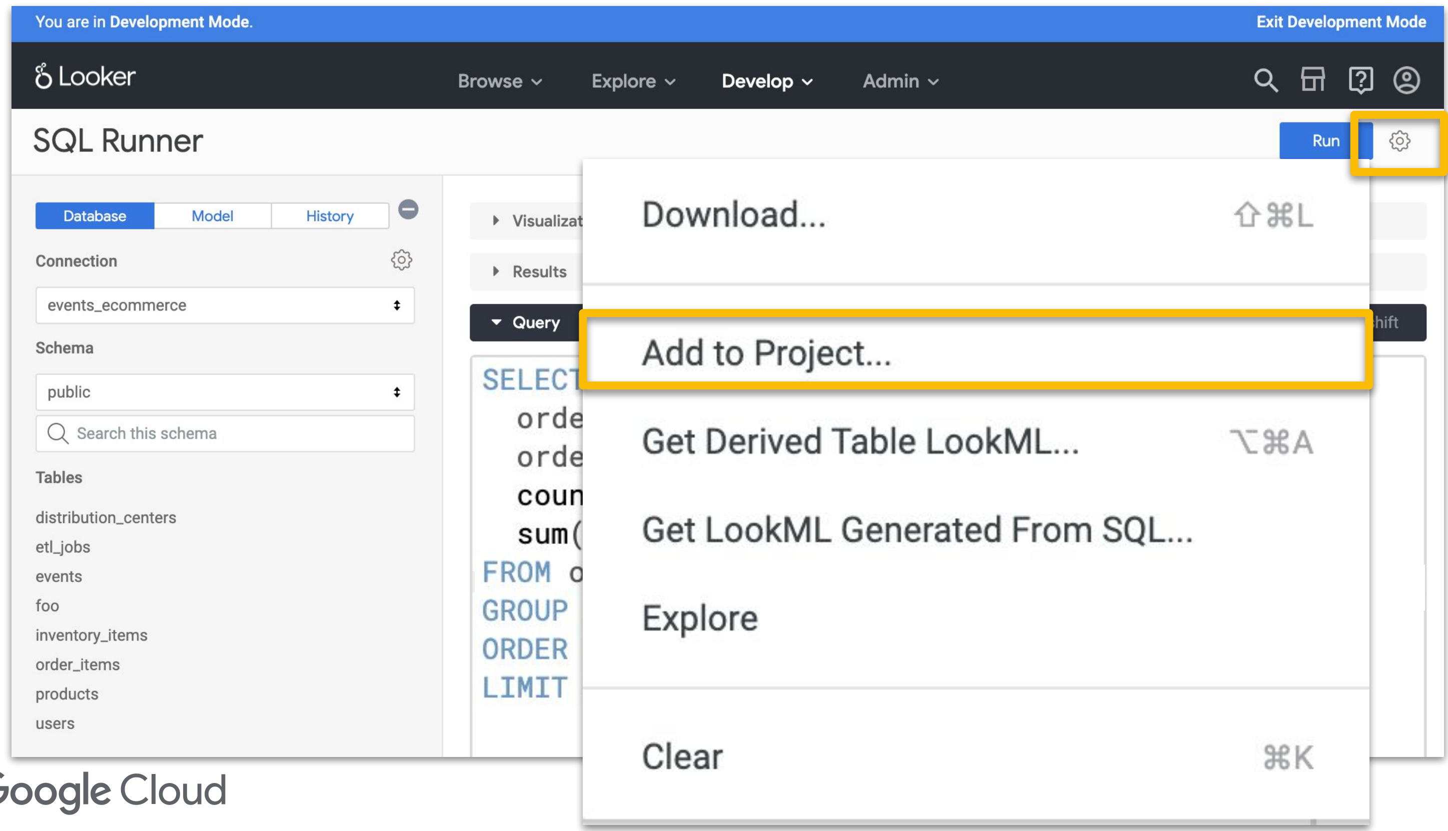
Visualization

Results

Query Amazon Redshift

```
SELECT
  order_items.order_id,
  order_items.user_id,
  count(*) as order_item_count,
  sum(order_items.sale_price) as order_total
FROM order_items
GROUP BY 1,2
ORDER BY 3 DESC
LIMIT 50
```

Step 2 Add Query into your LookML Project



Step 3 Name your New View

You are in Development Mode. Exit Development Mode

Looker Browse Explore Develop Admin

SQL Runner

Run

Add to Project

Project: ecommerce

View Name: sql_runner_query

Cancel | Add

inventory_items
order_items
products
users

Step 4

Review & Clean Up Your View

Looker creates a new view with the SQL Runner query and will write dimensions for every field as well as a count measure:

You are in Development Mode. Exit Development Mode

Looker | Browse | Explore | Develop | Admin

ecommerce | quick-fix | shared branch | Validate LookML

File Browser

- brand_overview.dashboard
- category_order_facts.view
- distribution_centers.view
- ecommerce_data.model
- etl_jobs.view
- events.view
- foo.view
- inventory_items.view
- lookml_best_practices.md
- map.topojson
- order_facts.view
- order_facts_sql_runner.view
- order_items.view
- products.view
- urls_for_pulling_from_git.md
- user_facts.view
- user_order_facts.view
- user_order_sequence_facts.view
- user_order_sequence_facts_cascaded.view
- user_retention.view
- users.view

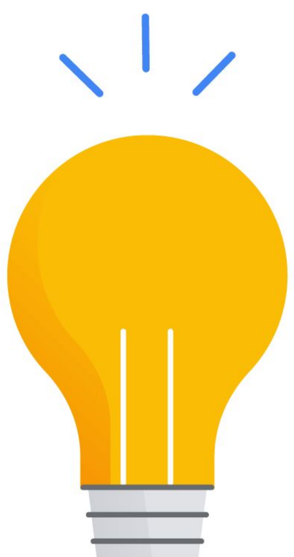
order_facts_sql_runner.view

```
view: order_facts_sql_runner {
  derived_table: {
    sql: SELECT
          order_items.order_id,
          order_items.user_id,
          count(*) as order_item_count,
          sum(order_items.sale_price) as order_total
        FROM order_items
        GROUP BY 1,2
        ORDER BY 3 DESC
        LIMIT 50
        ;;
  }

  measure: count {
    type: count
    drill_fields: [detail*]
  }

  dimension: order_id {
    type: number
    sql: ${TABLE}.order_id ;;
  }

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

  dimension: order_item_count {
    type: number
    sql: ${TABLE}.order_item_count ;;
  }

  dimension: order_total {
    type: number
    sql: ${TABLE}.order_total ;;
  }

  set: detail {
    fields: [order_id, user_id, order_item_count, order_total]
  }
```

Best Practice: 5 Steps for Success

After creating your shiny new derived table:

1. Move the view file to the appropriate **folder**
2. Erase any **LIMIT** in the SQL
3. Remove or hide the **count** measure
4. Establish the **primary key**
5. Write the desired **measures** and any additional dimensions



Native Derived Tables

Maximizing Code Reusability

The SQL for the User Facts table just built included the following definitions:

- COUNT(distinct order_items.order_id) as lifetime_order_count
- SUM(order_items.sale_price) as lifetime_revenue

```
measure: order_count {
  description: "A count of unique orders"
  type: count_distinct
  sql: ${order_id} ;;
}
```

```
measure: total_revenue {
  type: sum
  value_format_name: usd
  sql: ${sale_price} ;;
  drill_fields: [detail*]
}
```

BUT WAIT! These measures were already defined within the LookML in the `order_items` view

Native Derived Tables

How can we take advantage of dimensions and measures that have already been defined within the LookML?

Native Derived Tables are derived tables that perform the same function as a written SQL query, but are expressed natively in the LookML language

- Easier to read and understand when modeling data
- Enables code to be reused
- More maintainable since physical database references are minimized

Step 1 Build Your Query

Looker

Browse Explore Develop Admin

Explore

500 rows · 4.5s · just now Run

Orders and Revenue

Search

All Fields | In Use

- Custom Fields + Add
- Customers
- Inventory Items
- Order Items
  - DIMENSIONS
  - Created Date
  - Delivered Date
  - Delivery Time
  - Order ID
  - Order Status
  - Returned Date
  - Sale Price
  - Sale Price Tier
  - Shipped Date
  - User ID
  - MEASURES
  - Average Sales
  - Order Count
  - Order Item Count

Filters

Visualization

Data | Results | SQL | Row Limit 500 | Totals

Row limit reached. Results may be incomplete

	Order Items User ID	Order Items Total Sales	Order Items Order Item Count
1	2924	$2,424.90	14
2	9996	$2,125.58	14
3	16536	$2,101.15	12
4	278	$1,960.44	5
5	26618	$1,938.51	13
6	425	$1,930.65	24
7	56217	$1,858.74	13
8	10752	$1,818.98	6
9	32972	$1,806.00	2
10	56528	$1,804.10	14
11	16304	$1,758.49	17
12	5634	$1,754.94	13
13	11581	$1,747.93	17
14	7617	$1,694.68	16
15	3902	$1,694.28	15
16	12130	$1,664.87	16
17	2200	$1,643.85	21
18	6881	$1,641.92	13
19	362	$1,631.67	14

Step 2 Select **Get LookML**

Looker Browse Explore Develop Admin

Explore

500 rows · 4.5s · just now Run

Orders and Revenue

Search

All Fields In Use

- Custom Fields + Add
- Customers
- Inventory Items
- Order Items
  - DIMENSIONS
  - Created Date
  - Delivered Date
  - Delivery Time
  - Order ID
  - Order Status
  - Returned Date
  - Sale Price
  - Sale Price Tier
  - Shipped Date
  - User ID
  - MEASURES
  - Average Sales
  - Order Count
  - Order Item Count

Filters

Visualization

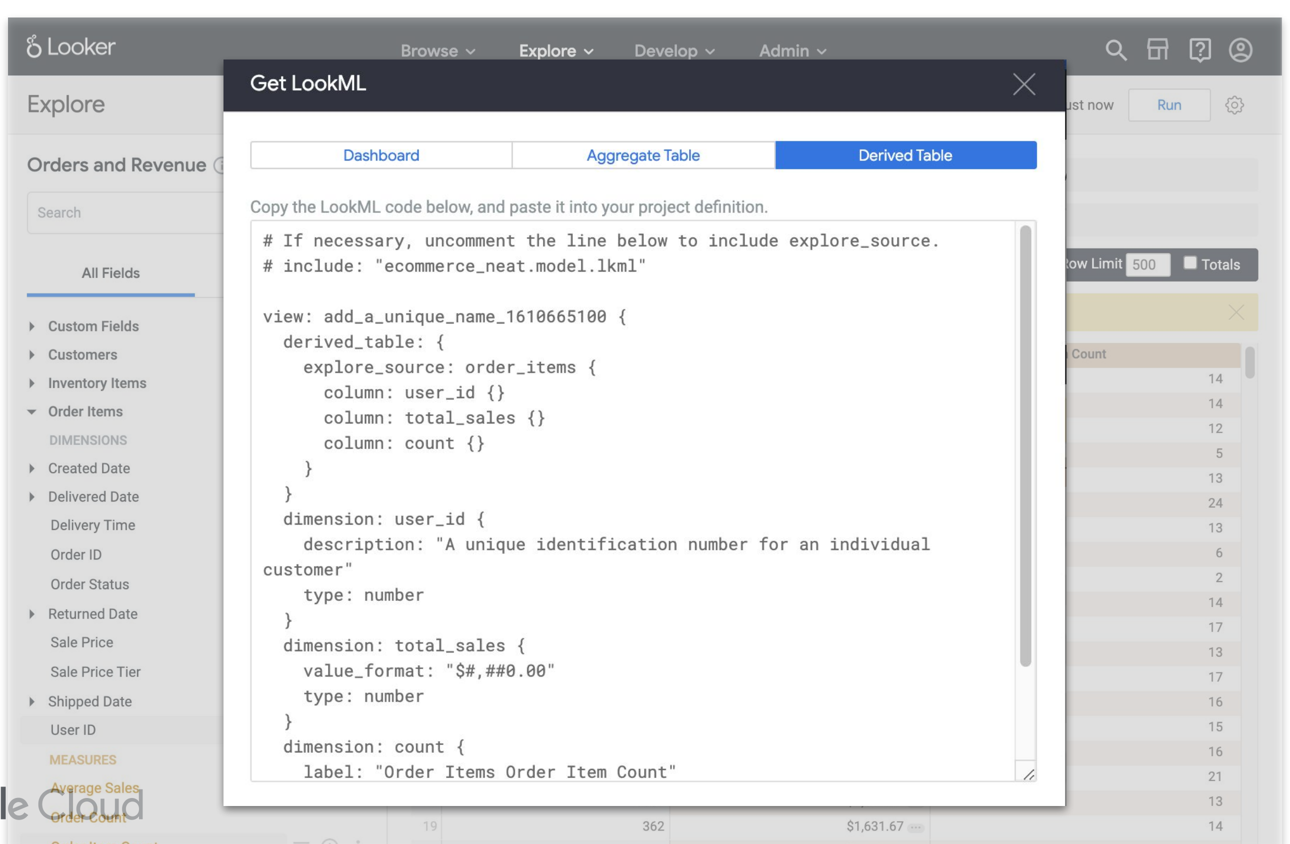
Data Results SQL

Row limit reached. Results may be incomplete

	Order Items User ID	Order Items To
1	2924	
2	9996	
3	16536	
4	278	
5	26618	
6	425	
7	56217	
8	10752	
9	32972	
10	56528	
11	16304	
12	5634	
13	11581	
14	7617	
15	3902	
16	12130	
17	2200	
18	6881	
19	362	$1,631.67

Menu item	Shortcut
Save as a Look...	⇧⌘S
Save to Dashboard...	⇧⌘A
Download...	⇧⌘L
Send...	⌥⇧S
Save & Schedule...	⌥⌘S
Share...	⌘U
Get LookML...	⌥⌘A
Merge Results...	
Remove Fields & Filters	⌘K
Clear Cache & Refresh	⇧⌘↩

Step 3 Copy the Derived Table LookML



Google Cloud

Native Derived Table Parameters

`explore_source:`

the Explore defined within Looker that contains the field and join definitions required for the desired query

`column:`

specifies an output column for the derived table

- often paired with a “field” parameter to link the new table column back to the appropriate underlying column
- can be named differently from the underlying field referenced

`filters:`

can be used for applying filters to the derived table using the same syntax as a filtered measure

Native Derived Table Parameters

`derived_column:`	`bind_filters:`	`expression_custom_filter:`
specify one or more columns that don't exist in the Explore specified by the explore_source parameter	used for applying templated filters to the native derived table	specify one or more custom filter expressions on an explore_source query

Persistent Derived Tables

Persistent Derived Tables

Add two (2) parameters to a derived table when persisting it:

1. Table refresh logic for table rebuilding
   a. `datagroup_trigger`: triggered by some change that takes place in the underlying data as defined within a datagroup
   b. `sql_trigger_value`: triggered by a change in the underlying data
   c. `persist_for`: a set time period
2. Indexes
   a. A single or multiple index for most databases
   b. Sort key(s) and a distribution key for Redshift
   c. Cluster key(s) and partition key(s) for BigQuery

Ephemeral vs. Persistent Derived Tables

Ephemeral derived tables will build at runtime as a temporary table (mysql) or via a SQL common table expression

Derived Table SQL

```
WITH user_order_facts AS (SELECT
        order_items.user_id as user_id
        , COUNT(DISTINCT order_items.order_id) as lifetime_orders
        , SUM(order_items.sale_price) AS lifetime_revenue
        , MIN(NULLIF(order_items.created_at,0)) as first_order
        , MAX(NULLIF(order_items.created_at,0)) as latest_order
        , COUNT(DISTINCT DATE_TRUNC('month', NULLIF(order_items.created_at,0))) as number_of_distinct_months_with_orders
        , SUM(order_items.sale_price) AS order_value
      FROM order_items
      GROUP BY user_id
      )
```

Explore SQL

```
SELECT
    user_order_facts.lifetime_orders  AS "user_order_facts.lifetime_orders",
    COUNT(DISTINCT order_items.order_id ) AS "order_items.order_count"
FROM public.order_items  AS order_items
LEFT JOIN user_order_facts ON user_order_facts.user_id = order_items.user_id

GROUP BY 1
ORDER BY 2 DESC
LIMIT 500
```

Ephemeral vs. Persistent Derived Tables

Persistent derived tables will be stored as physical tables within the database once built. Looker will then simply query those physical tables as needed. Looker will build separate PDTs in development and production modes

```
-- use existing user_order_facts in teach_scratch.LR$KDYI2NQM4DW046XHR9XQH_user_order_facts
SELECT
    user_order_facts.lifetime_orders  AS "user_order_facts.lifetime_orders",
    COUNT(DISTINCT order_items.order_id ) AS "order_items.order_count"
FROM public.order_items  AS order_items
LEFT JOIN teach_scratch.LR$KDYI2NQM4DW046XHR9XQH_user_order_facts AS user_order_facts ON user_order_facts.user_id = order_items
    .user_id

GROUP BY 1
ORDER BY 2 DESC
LIMIT 500
```

Summary

Derived Tables

WHAT: Tables defined within Looker that do not exist in the database

- Two types of derived tables
    - Ephemeral: built at query time
    - Persisted: stored in the database
- Two ways to write derived tables
    - SQL
    - Native (using LookML)
- Defined within the LookML
- Referenced in the LookML just like any other table

Summary

Derived Tables

WHY: Expand the sophistication of analyses

- Aggregate data to a different level of granularity (***ex:*** *aggregate fact data*)
- Speed up performance (***ex:*** *precompute joins*)
- Write custom SQL for advanced use cases (***ex:*** *utilize window functions*)

Questions?

Q&A from the audience:

Q: When using links, to create them available to just a particular context, I've been creating a new dimension with a unique name, including in a report, and then hiding the field. Is there a better way to manage dimensions that vary only by the link?

A: It sounds like this could be a good use case for some liquid conditional statements! Not knowing your particular use case, let's assume we have user attributes for different departments (A, B, C, etc), and each department needs its own unique link for the dimension. To accomplish this, we would use the following pattern for as many departments as needed:

```
dimension: multiple_links {
     type: string
     sql: ${TABLE}.field_name
     html:
          {% if _user_attributes['department'] == ‘Department A’ %}
               <a href=”www.link-for-department-a”> Link text </a>
          {% elsif _user_attributes['department'] == ‘Department B’ %}
               <a href=”www.link-for-department-b”> Link text </a>
          … etc
          {% endif %};;
          }
```

Google Cloud

Q&A from the audience:

Q: Will New UI Dashboards support drilling to dashboards?
A: Drilling improvements are slated for the new dashboard experience, but no details are available at the moment. If drilling to dashboards is something you'd like to see, I highly recommend creating a feature request in our Community (community.looker.com).

Q: Can you pass more than one filter to a dashboard using liquid?
A: Yes! You can pass multiple filters in using liquid. As long as the dashboard you are drilling to has filters created on it, use the following pattern:

```
dimension: dimension_name {
      type: string
      sql: ${TABLE}.field_name ;;
      html:
      <a href="/dashboards/dashboardnumber?Filter1={{ value }}&Filter2={{ field2_name._value}}">
            {{ value }}
      </a> ;;
}
```

Q&A from the audience:

Q: What is the difference between encode_uri vs encode_url?
A: Both tags will encode strings so they function correctly in a link. `encode_url` is the tag created for base liquid by Shopify while `encode_uri` is unique to Looker.

Q: Does the filter that you use in the URL for liquid have to be saved as part of the target dashboard? What if dashboard 24 doesn't have a "Brand" filter at the top?
A: The target Dashboard needs to have a filter created on it that matches the name of the filter in the link.

Google Cloud

Q&A from the audience:

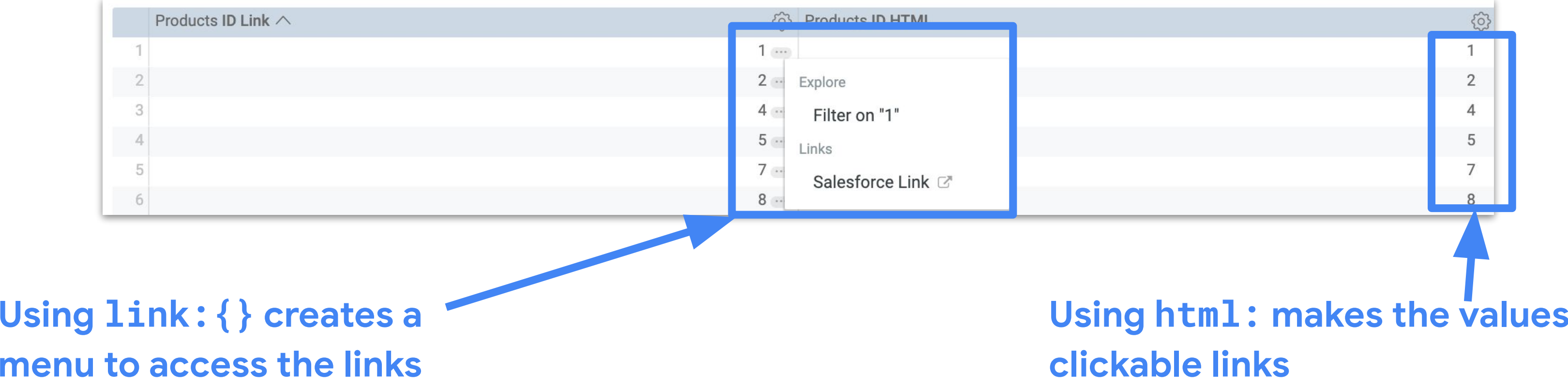
Q: So a LookML with this link URL Code is coded in the LookML. What if I have a dashboard where I don't want a link on the Dimension Brand Do I have to refer to a different LookML Code?

A: Currently there is not a way to have the Brand dimension display or hide a link at the Dashboard level, but you can control if a link is shown based on which Explore the Brand dimension is a part of. Let's say we have two Explores, Yeslinks and Nolinks. We could use the following patterns to shown links on the Brand dimension in the Yeslinks Explore, and hide links in the Nolinks Explore.

```
dimension: brand {
      type: string
      sql: ${TABLE}.string
      html:
            {% if _explore._name == 'Yeslinks' %}
                  <a href="/dashboards/24?Brand={{ value | encode_uri }}"> Link text </a>
            {% else %}
                  {{ value }}
            {% endif %};;
            }
```

Q&A from the audience:

Q: For something like the salesforce link example, is it best practice to use html or link (if not doing much customization to the link)?
A: It mainly depends on how you want your link to appear to the user.



Q&A from the audience:

Q: For which user permissions is drilling available?
A: A user will need the see_drill_overlay permission to see drills.

Q: How do you make the custom drill with html open in a new tab?
A: Adding target="_blank" to a link will cause it to open in a new tab or browser window, depending on the user's browser settings.

```
<a href="/dashboards/24?Brand={{ value | encode_uri }}" target="_blank"> Link text </a>
```

Q: This custom drilling will show as a pop up? like the normal drill?
A: The example with user item and order history opens in a new window. It is also possible to customize drill menus with liquid, please check out our Help Center Article "More Powerful Data Drilling".

Q&A from the audience:

Q: If you're referencing a different field in a custom drill (like ID instead of name), does the relationship between the field clicked and the field referenced need to be 1:1?
A: Yes

Q: Can this conditional formatting be applied also on the second column in your example?
A: I believe this is referring to the example of custom colored bars using html, in which case the answer would be yes. Formatting using liquid and html can be applied to both dimensions and measures.

Q: What's the best resource for learning more about liquid syntax and potential applications of it?
A: I highly recommend reading our Liquid Variable Reference Looker documentation for syntax help and examples of using liquid in Looker. Additionally, searching our Help Center (help.looker.com) and Community (community.looker.com) for Liquid will produce a treasure trove of examples and applications.

Q: Can I use a parameter to set the colors?
A: Yes! We have an example of using a parameter to change colors in our Documentation.

Q&A from the audience:

Q: Does looker accept html5?
A: For security, Looker only accepts certain sanitized HTML. You can find the allowed HTML in our Documentation.

Q: What's _linked_value?
A: `linked_value` is the value of the field with Looker's default formatting and default linking.

Q: How the Total Gross Margin example looks like in the graph?
A: In the example showing the Total Gross Margin with a drop-down detail of the values that compose it, the results must be visualized as a table visualization for the HTML customization to work. Other visualization types will not correctly display the drop-down.

Q&A from the audience:

Q: On that last example - if you pull into multiple numbers into a single tile, how does that affect dashboard load times?
A: Because we are rendering more data in a single visualization, it take additional resources for the browser to load than a standard single value visualization. However, dashboard performance impact should be negligible.

Q: What is the LookML code (liquid) of the last viz?
A: You can find the LookML code for the colorful single value visualization in this Community post.

Q&A from the audience:

Q: Question about the advanced single viz html formatting: the code for icons references a html "fa" class (eg. fa fa-facebook; fa fa-leaf). does this mean that looker has imported the Font Awesome CSS library in the background? thanks

A: Yes, Looker uses Font Awesome v4.1.0.

Q&A from the audience:

Q: Can I build a parameter/suggestion field? Parameters have value & label, which would be tremendous for users to pick a friendly label and pass an ID in the value.
A: This sounds like a templated filter may be better suited to this use case. While parameters require users to choose from a predetermined value, templated filters allow users to input a much wider range of values and filter options.

Q: Can a parameter be used to filter the options available for another filter? Like we can do with dashboard filters
A: Not currently, but this sounds like an excellent idea for a feature request!

Q: Is it possible to link two different liquid filters from two different views (both joined to the same explore) to the same Dashboard filter?
A: Yes, provided the filter on the dashboard matches the filter name in the URL being used in the liquid.

Q&A from the audience:

Q: Does the level of detail also work for the dimension_group date? For example if I want to go from day to hour, would I be able to do that?
A: Yes, you can reference different parts of a dimension group to change the granularity of time. This Help Center article has an example of using parameters to change the granularity of time.

Q: is there a way to select multiple options from a parameter or input a value?
A: A parameter takes a single input, for multiple inputs you will want to use a templated filter.

Q: Does this mean that Brand becomes the default value if the user doesn't click to select something else?
A: Correct, because we set a default value of “Brand” this is the level of granularity that will be used if a user does not select anything.

Q&A from the audience:

Q: How do I insert a parameter value in the sql: block of a derived table?
A: Parameters and liquid can be referenced in the SQL block of a derived table. To see all of the places the parameter liquid variable can be used, please see our documentation.

Q: How can I create a filter in LookMLs to filter information by "User Logged". Eg: I want to display on a dashboard, only sales done by the User's Team (Assuming there is a Team Table with Manager's name)
A: This is an excellent use case for User Attributes! We have a Help Center article that walks through this very use case.

Q: Are these templated filters usable as page filter too, or only within a look?
A: Templated filters can be used on Explore, Looks, and Dashboards.

Q&A from the audience:

Q: why a templated filter is better than using a dimension as a filter?
A: Standard Looker filters created from dimensions affect the WHERE clause of the generated SQL, and therefore apply to all fields in the query. Templated filters can be applied directly to fields in the SELECT clause of the generated SQL, opening up a world of advanced filtering and comparison capabilities.

Q: All the example for templated filters shows a drop down style? Will it be possible to do check box, button bars slider style filters?
A: The example shown were on an Explore, which only uses a drop down style. Templated filter can also be used on a Dashboard where more filtering options are available.

Q: Do other filters apply to the templated filters? let's say comparing jeans vs other categories in a certain geographical area
A: Yes! Standard Looker filters created from dimensions affect the WHERE clause of the generated SQL, and therefore apply to all fields in the query

Q&A from the audience:

Q: Can you use templated filters and parameters on dashboards?
A: Yes!

Q: Is the query only cached for the initial 500 results?
A: The query is cached for all of the returned results. If you have 5000 results then 5000 rows will be cached.

Q: Should we expect questions like these for lookML developer certification?
A: Topics such as derived tables and caching are included in the Exam Guide for the LookML Developer certification exam, and therefore are fair game. For a list of topics to study please check out our Exam Guides.

Q: How do we set cache for dashboard or Looks
A: To set a cache for a dashboard or Look, you want to create a caching policy for the model or Explore that provides the data on the dashboard.

Q&A from the audience:

Q: If an Explore contains three tables with different ETL timings, how do you refresh the cache whenever any of the three tables updates?
A: You will want to create a datagroup trigger that reflects the table that updates the most frequently, and then apply that datagroup to any Explores you want updated.

Q: Why do we need trigger and cache age both
A: Although only one of these two options is required, we recommend using both when creating a datagroup. For example, the `max_cache_age` parameter ensures that if the cache for a datagroup isn't cleared by the `sql_trigger`, then the cache entries will expire by a certain time.

Q: Can we have data groups based on etl triggers like control M and other scheduling tools that will send inputs or cache cubes based on file drop?
A: The `sql_trigger` for a datagroup must be derived from a value that can be calculated in the database you have connected to Looker.

Q&A from the audience:

Q: Can you explain what is symmetric aggregation?
A: Lloyd Tabb has a wonderful explanation of symmetric aggregation in our Help Center.

Q: How do i best join dimension/measures from derived table back to another view? it seems like when i try to do this, the query runs for along time. Any tips?
A: You may want to double-check that you have the correct join relationship between your derived table and your view. Additionally, all views should have unique primary keys defined. Derived tables should have indexes and sort keys defined. For additional optimization tips, check out this Help Center article.

Q&A from the audience:

Q: From your experience what is the best way to build a histogram like visualization as it doesn't seem to be native in Looker. I know step 1 is derived table, but how would I get the "bucketing"?
A: Depending on what you are bucketing, the first step may indeed be to create a derived table so that you can calculate your aggregate and then reference it as a dimension in Looker. Once you have done that, the next step will be to create a tier dimension to bucket those values into the ranges you desire. This Help Center article can walk you through these steps. Also, if you are using custom visualizations in the Looker Marketplace, you may be interested to know that a histogram visualization was recently added!

Q: Are PDT created as a new view within the model?
A: If you are following the steps to create a derived table from SQL Runner or and Explore then yes, the derived table will be added as a new view within the model.

Thank you