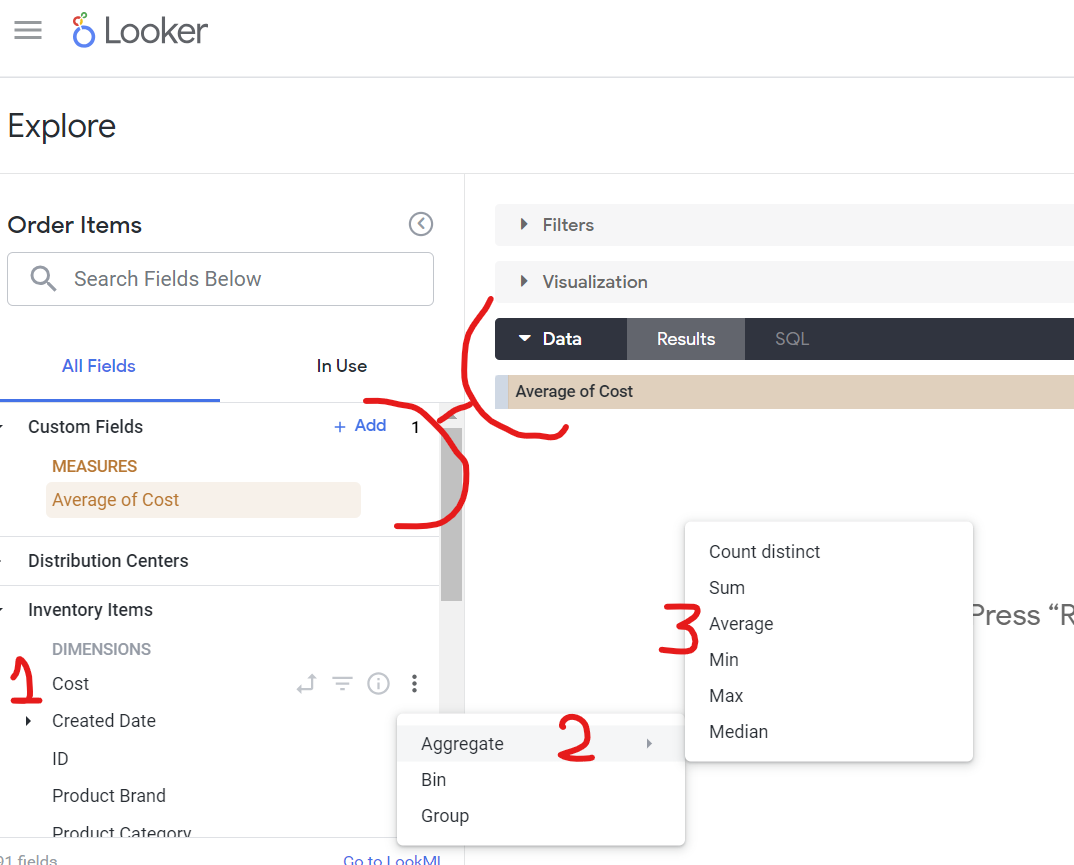
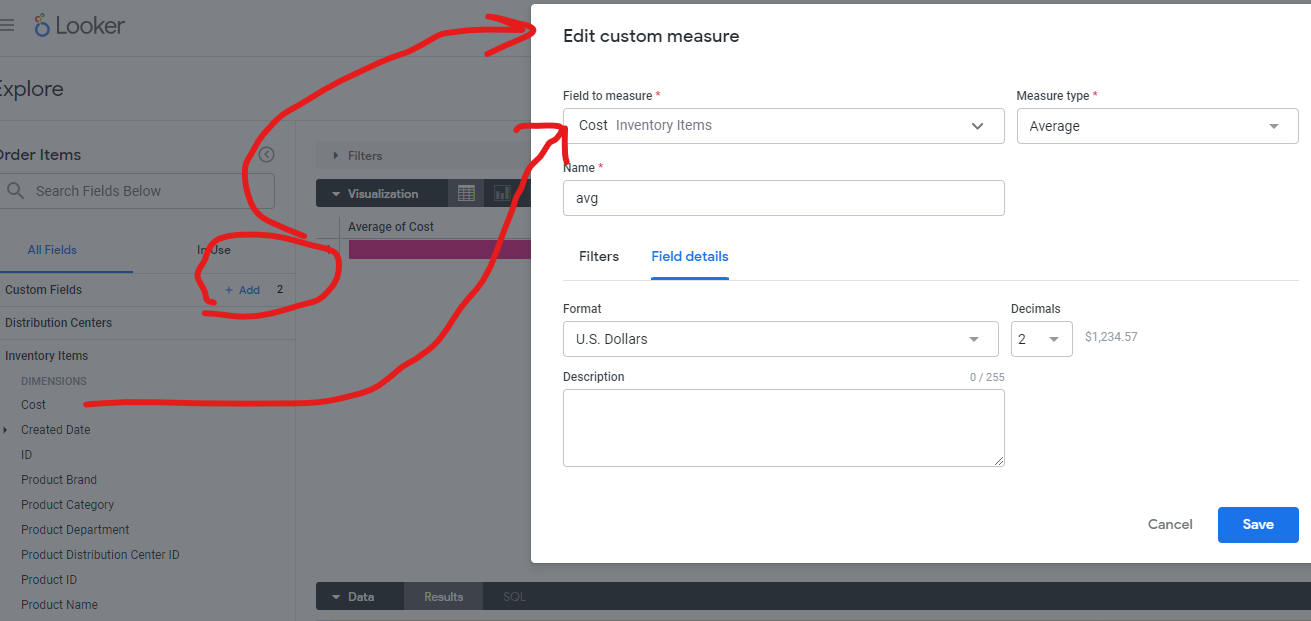
Looker Labs

Using Custom Fields in Looker Explores.

**Custom Fields –** can be created within the explores without the need for any developing skills.

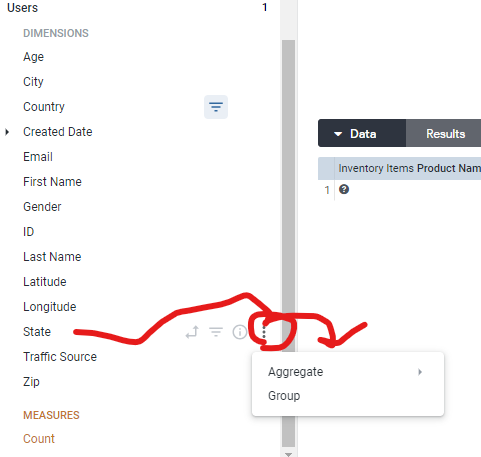
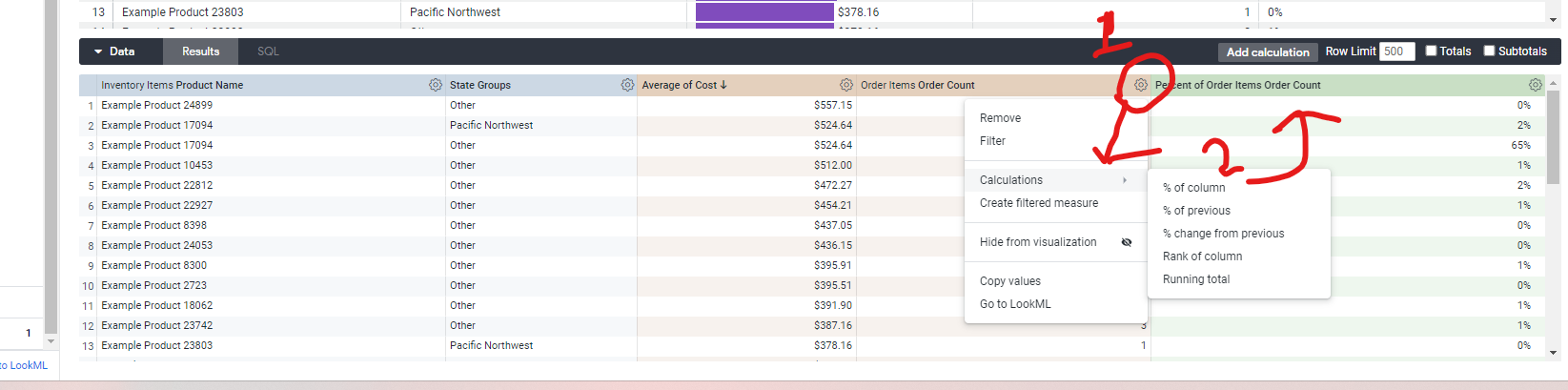
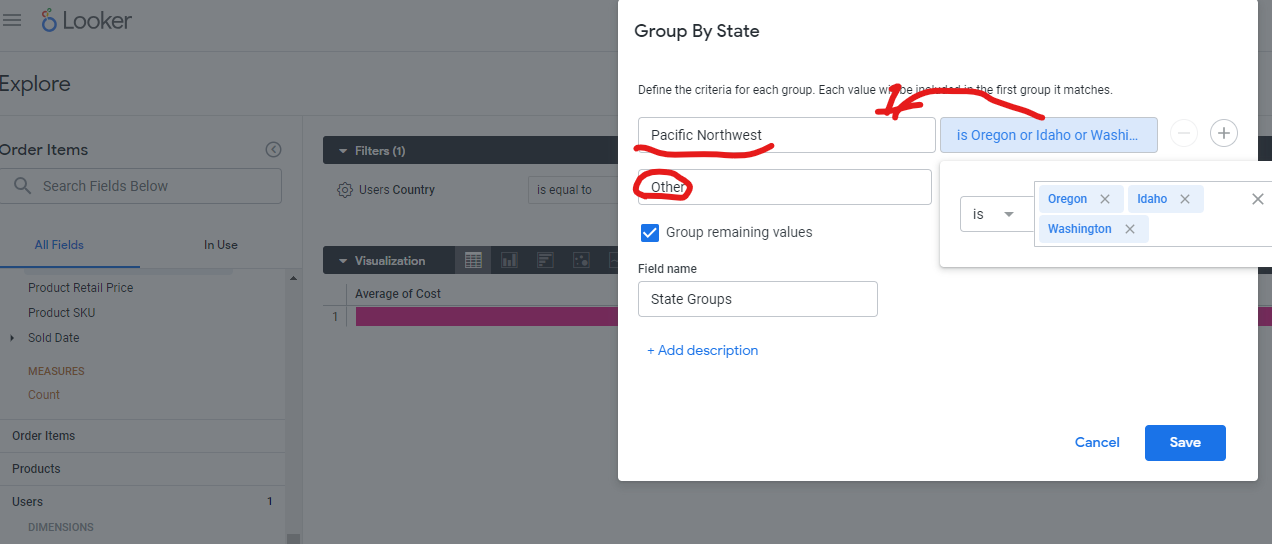
Option 1: Creating a custom field from existing field, such as creating ‘average cost’ based on the ‘Average’ dimension.

Option 2: Create custom fields by pressing ‘Add’ in custom fields section and choosing the type of field you want to create – Measure – Dimension – or Table Calculation.



**Create a custom grouping**

The **Group** **custom** **field** **type** lets you create ad hoc custom groups for dimensions and custom dimensions without needing to use logical functions in Looker expressions or needing to develop CASE WHEN logic in sql parameters or type: case fields.

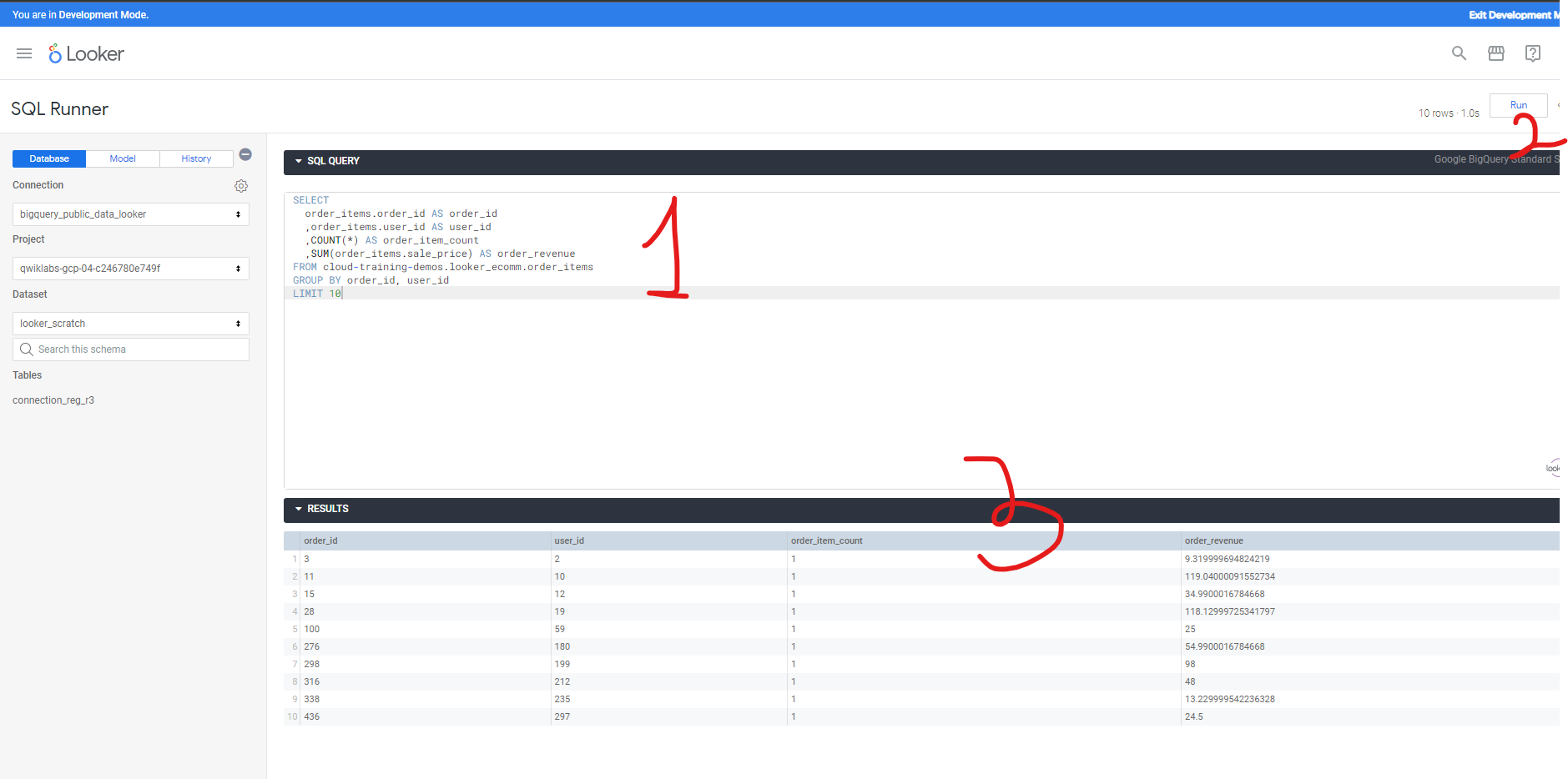
**Turning values into % relative to total:**

**Derived tables**

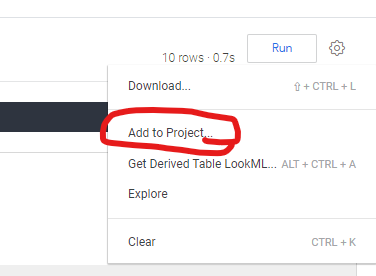
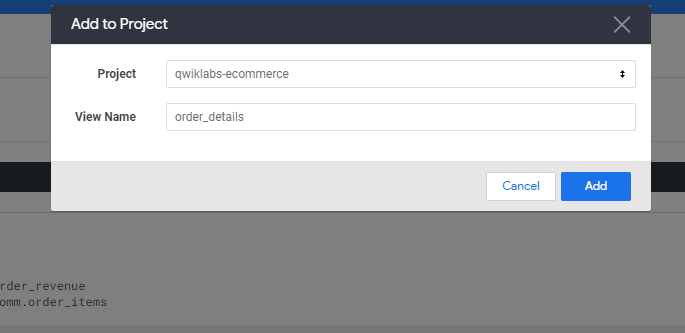
In **Looker, LookML** developers can use derived tables to create new tables that are not already defined in the underlying database. For example, as a **LookML** developer, you can create derived tables to summarize details from existing tables, such as order details for each order in an e-commerce dataset.

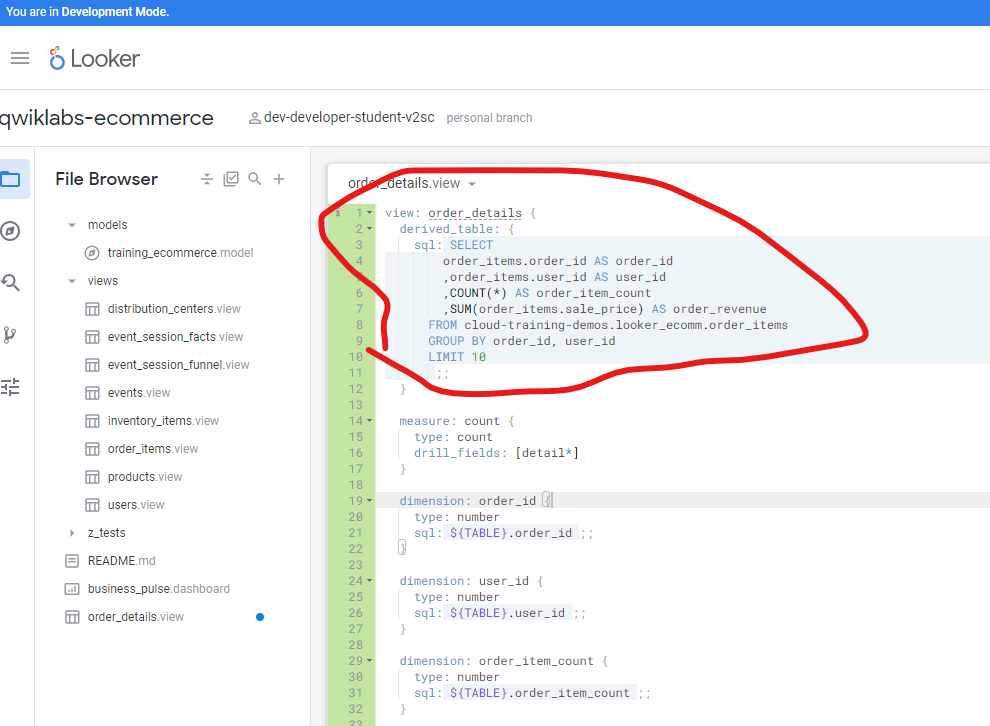
**SQL derived table (virtual CTE) –** created using sql queries

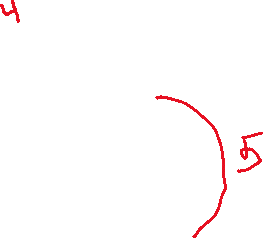
1. Develop > SQL runner
2. In SQL runner write your SQL code:

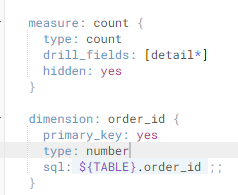


1. Add to existing project by clicking gear next to RUN



1. You will be redirected to **Looker IDE** where your newly created derived will be added to the view you’ve chosen in step 3.
2.  Looker auto-generates a dimension for each column in the SELECT clause of the SQL query as well as a new count measure. SEE above step 5

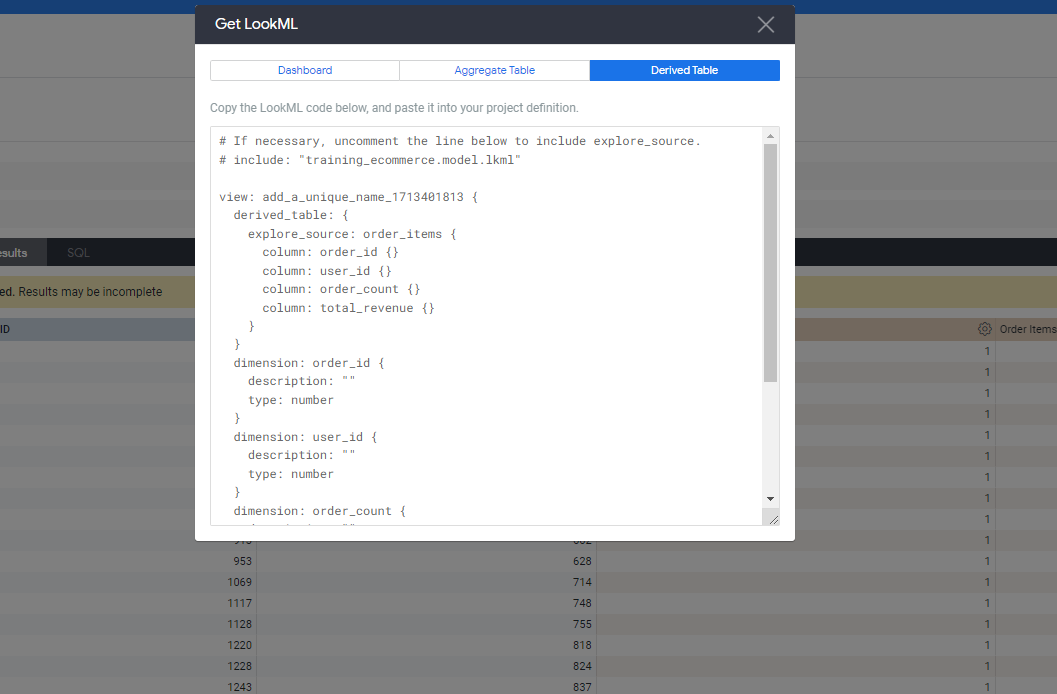


1. Add a primary key to this new generated view by adding ‘primary\_key: yes’ to appropriate dimension

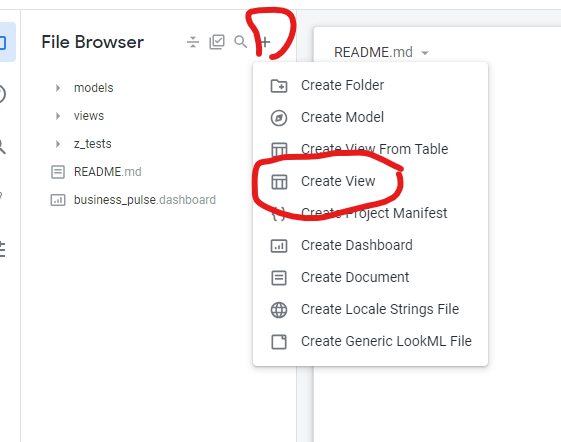
**Native derived table (virtual CTE) –** created in Explore queries and then copied to LookML files

* They allow you to inherit already existing dimensions, measures, and even Explores and join logic.
* Makes code more maintainable
* Rather than creating a new SQL derived table, you can use existing dimensions and measures to easily define a new NDT.

1. Go to explore and select appropriate explore (ex: explore > E-commerce > order items
2. From the available views, choose appropriate dimensions and measures (that you want) and run to see results
3. Click Settings (settings gear icon) next to Run (top right of page), and select Get LookML
   1. Copy Derived Table LookML code

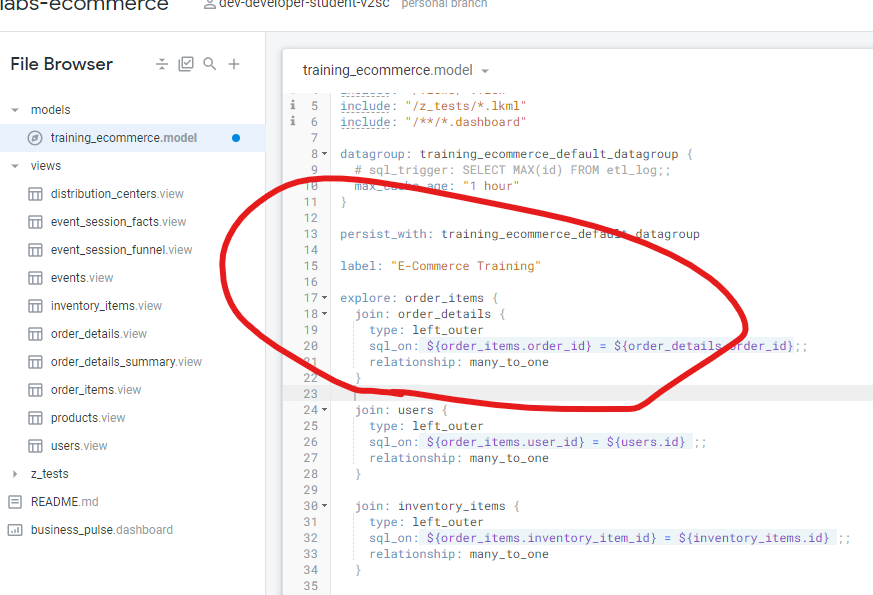


1. On the Looker navigation menu, click the Develop tab and then select the qwiklabs-ecommerce LookML project.
2. Next to File Browser, click Add file or folder (Add file or folder icon) and select Create View. Then copy paste the LookML code for the derived table.



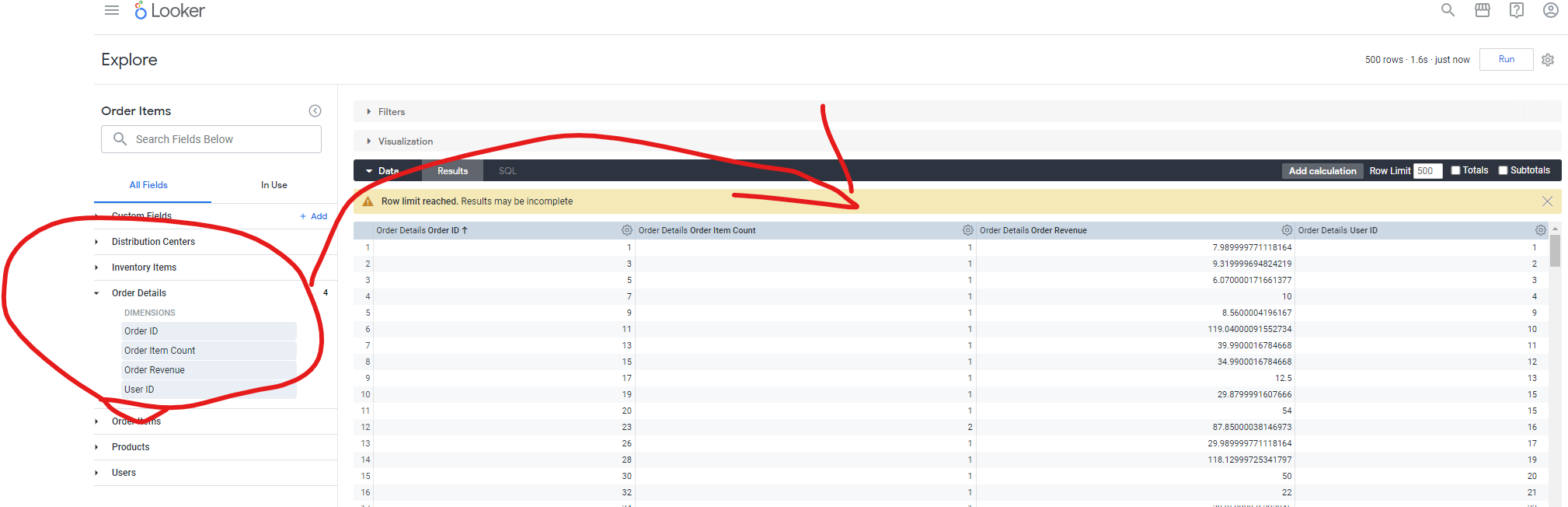
**Task 3. Join a new view to an Explore**

1. In the model file, in your explore, add the join condition for the newly created view:

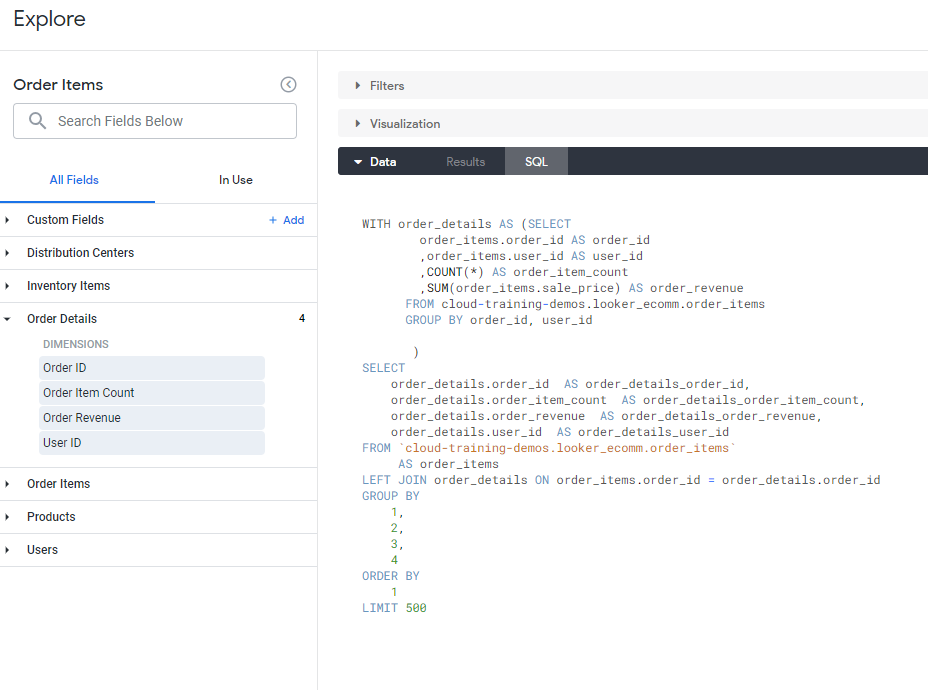


The sql\_on parameter identifies the join field as order\_id. The relationship parameter identifies that there are potentially many instances of an order\_id in order\_items but only one instance of each order\_id in order\_details, which is organized as one summary row for each order.

1. This view will now be accessible in the explores:



1. Click on the SQL tab to see the SQL query generated by Looker.



The CTE is a derived table that we’ve crated.

**Persisting A Derived tables:**

**Persistent derived tables, or** PDTs, are written to and stored in the connected database.

* Ready to go ad hoc
* Take up storage space

To persist a derived table, you must use **one or two of these parameters** in the definition:

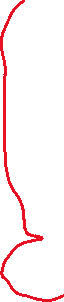
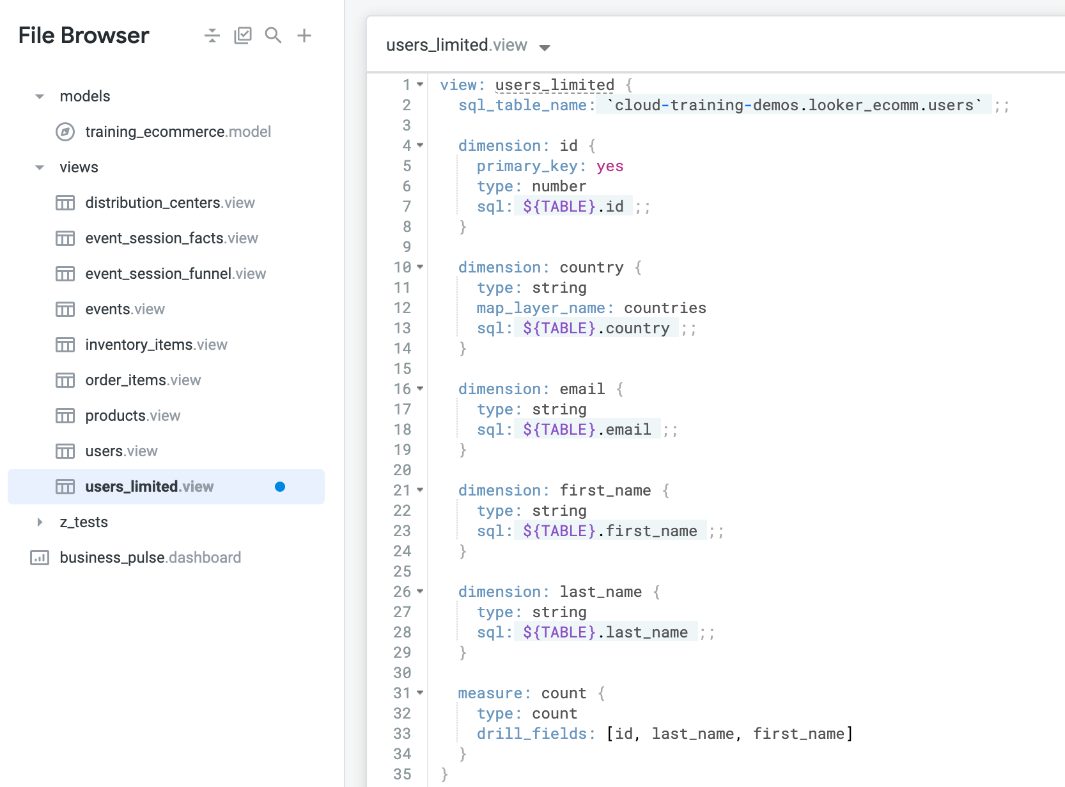
* datagroup\_trigger uses a datagroup, or caching policy configured in the model. If datagroups are defined in the model, then this is the best practice for persisting derived tables.
* sql\_trigger\_value uses a pre-written SELECT statement that returns one value. Looker sends that SELECT statement to the database repeatedly, and when it discovers the result has changed, it takes this as a cue to rebuild the PDT.
* persist\_for instructs the PDT to stay up for a set duration, such as “1 hour” or “4 hours”.

1. you persist the native derived table using the parameter **datagroup\_trigger,** which rebuilds a persistent derived table based on a pre-defined datagroup (cache policy) in the model file.
2. Go to the derived table view and add appropriate parameter:



# **Task 1. Create a view**

1. Go to develop, select developer mode, there click on ‘Create View’
2. Drag your newly created view into the existing Views folder.
3. Specify the SQL table name you want to connect your view to.
4. Add dimensions and a measure:



**Drill\_fields:** The drill\_fields parameter controls what happens when a user clicks the value of a table cell while they're exploring data. When a user does this, they "drill" into the data, allowing them to see the individual records that make up that cell, limit their query to the cell's value, or slice the data in a related way.

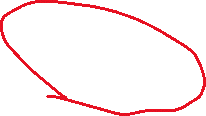
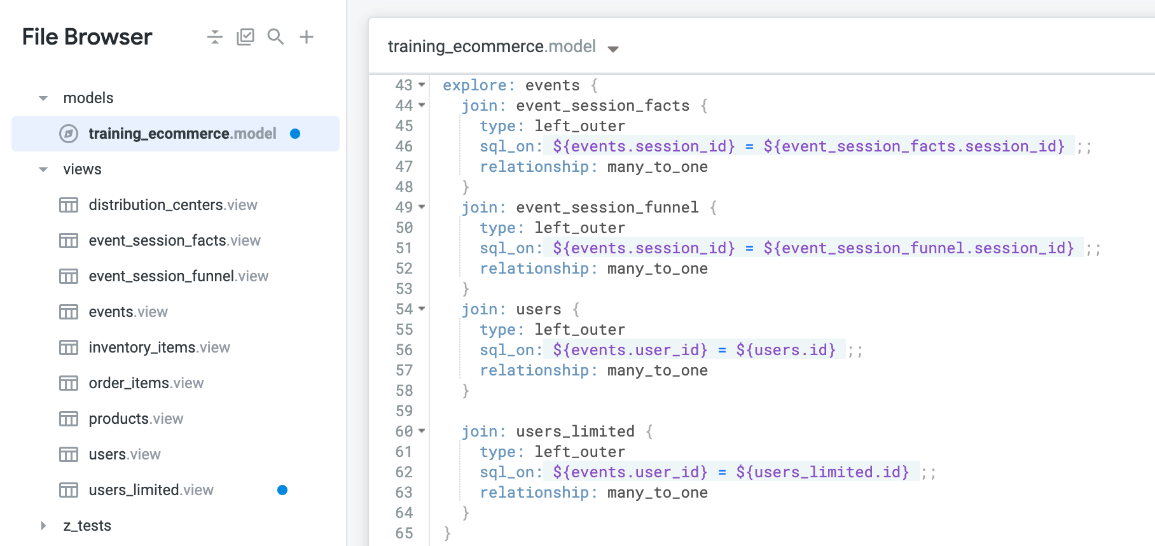
**${TABLE}** is a Looker variable that represents the name of the SQL table associated with the view. It's a placeholder that Looker will replace with the actual table name when executing the SQL query.

* **;;** indicates the end of the SQL expression.

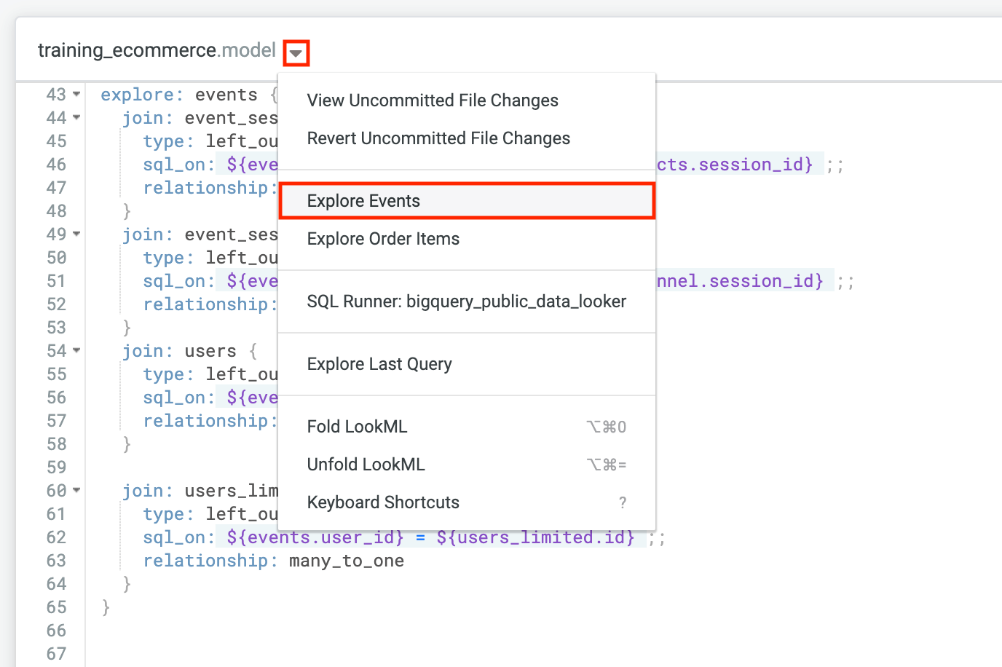
**map\_layer\_name: countries**: The map\_layer parameter lets you define a custom map layer that can then be used to plot regional data, such as counties or zip codes, in Looker and create map charts.

* + Built-in: countries, us\_states, us\_counties\_fips etc.

1. Save changes, validate, commit and deploy to production
2. Join your created view to an existing explore based on primary keys



1. You can check it, by navigating to explores the following way:

**Looker Types of Joins:**

**MYSQL types of Joins:**

**BigQuery types of Joins:**