Latest date:

Measure: Latest\_update\_date{

Type:date

Sql:max(${created\_raw})

}

Yes or no

measure: yes\_no\_total\_revenue {

type: yesno

sql: ${total\_revenue}>100000;;

}

Aggregate measure can perform measures only And filters can work

Distinct is allows only unique values

It wont allows duplicate values

Aggregate measure working on top of the dimension

It only allows to perform dimensions only

Non-aggregated measures🡺string,date,number,yesno

Filters not support

Non-aggregated measures🡺 can call and perform measures and dimensions both

Post-sql only can call and perform measures only

measure: percent\_of\_previous {

type: percent\_of\_previous

sql: ${order\_count} ;;

value\_format\_name:decimal\_0

### Model : Multiple models can exist for the same database connection in a single LookML project. Each model can expose different data to different users. For example, sales agents need different data than company executives, and so you would probably develop two models to offer views of the database appropriate for each user.

* Multiple [view](https://cloud.google.com/looker/docs/lookml-terms-and-concepts#view) files, one file for every table in the database.One [model](https://cloud.google.com/looker/docs/lookml-terms-and-concepts#model) file. The model file declares an [Explore](https://cloud.google.com/looker/docs/lookml-terms-and-concepts#explore) for every view.
* Each Explore declaration includes [join](https://cloud.google.com/looker/docs/lookml-terms-and-concepts#join) logic to join any view that Looker can determine is related to the Explore.

### View

A view declaration defines a list of [fields](https://cloud.google.com/looker/docs/lookml-terms-and-concepts#field) (dimensions or measures) and their linkage to an underlying table or derived table. In LookML a view typically references an underlying database table, but it can also represent a [derived table](https://cloud.google.com/looker/docs/lookml-terms-and-concepts#derived-table).

A view may join to other views. The relationship between views is typically defined as part of an [Explore](https://cloud.google.com/looker/docs/lookml-terms-and-concepts#explore) declaration in a model file.

By default, view names appear at

### Explore

 the [connection](https://cloud.google.com/looker/docs/reference/param-model-connection) parameter is used to specify the [database connection](https://cloud.google.com/looker/docs/lookml-terms-and-concepts#database_connection) for the model, and the [include](https://cloud.google.com/looker/docs/reference) parameter is used to specify the files that will be available for the model to reference.

The explore declaration in the example above also specifies join relationships between views. For details on join declarations, visit the section on [joins](https://cloud.google.com/looker/docs/lookml-terms-and-concepts#join) on this page. Visit the [Join parameters](https://cloud.google.com/looker/docs/reference/param-join) documentation page for more details about the LookML parameters that can be used with the join parameter.

### Dimension and measure fields

[Views](https://cloud.google.com/looker/docs/lookml-terms-and-concepts#view) contain fields, mostly dimensions and measures, which are the fundamental building blocks for Looker queries.

In Looker, a dimension is a groupable field and can be used to filter query results. It can be:

* An attribute, which has a direct association to a column in an underlying table
* A fact or numerical value
* A derived value, computed based on the values of other fields in a single row

In Looker, dimensions always appear in the GROUP BY clause of the SQL that Looker generates.

In Looker, fields are listed on the **Explore** page in the [field picker](https://cloud.google.com/looker/docs/changing-explore-menu-and-field-picker#field_picker_display_overview) on the left side of the page. You can expand a view in the field picker to show the list of fields that are available to query from that view.

To import LookML files from a different project, use the project manifest file to specify a name for your current project and the location of any external projects, which could be stored locally or remotely. For example:

# This project

project\_name: "my\_project"

# The project to import

local\_dependency: {

project: "my\_other\_project"

}

remote\_dependency: ga\_360\_block {

url: "https://github.com/llooker/google\_ga360"

ref: "4be130a28f3776c2bf67a9acc637e65c11231bcc"

}

After defining the external projects in the project manifest file, you can use the [include](https://cloud.google.com/looker/docs/reference/param-model-include) parameter in your [model file](https://cloud.google.com/looker/docs/lookml-terms-and-concepts#model) to add files from those external project to your current project. For example:

include: "//my\_other\_project/imported\_view.view"

include: "//ga\_360\_block/\*.view"

### Sets

In Looker, a **set** is a list that defines a group of fields that are used together. Typically sets are used to specify which fields to display after a user [drills down](https://cloud.google.com/looker/docs/lookml-terms-and-concepts#drill) into data. Drill sets are specified on a field-by-field basis, so you get complete control over what data is displayed when a user clicks a value in a table or dashboard. Sets can also be used as a security feature to define groups of fields visible to specific users

set: order\_items\_stats\_set {

fields: [

### Drill down

Drill behavior is different for dimensions and measures:

* When drilling on a dimension, the new query filters on the drilled value. For example, if you click on a specific date in a query of customer orders by date, the new query will show orders only on the specific date.
* When drilling on a measure, the new query will show the dataset that contributed to the measure. For example, when drilling on a count, the new query will show the rows to calculate that count. When drilling on max, min, and average measures, drilling still shows *all* the rows that contributed to that measure. This means that drilling on a max measure, for example, shows *all* the rows that were used to calculate the max value, not just a single row for the max value.

## Creating persistent derived tables (PDTs)

To make a derived table into a persistent derived table (PDT), you define a [persistence strategy](https://cloud.google.com/looker/docs/derived-tables#persistence_strategies) for the table. To optimize performance, you should also add an [optimization strategy](https://cloud.google.com/looker/docs/derived-tables#optimization_strategies).

### Persistence strategies

The persistence of a derived table can be managed by Looker or, for [dialects that support materialized views](https://cloud.google.com/looker/docs/reference/param-view-materialized-view#dialect_support), by your database using [materialized views](https://cloud.google.com/looker/docs/reference/param-view-materialized-view).

To make a derived table persistent, add one of the following parameters to the derived\_table definition:

* Looker-managed persistence parameters:
  + [datagroup\_trigger](https://cloud.google.com/looker/docs/derived-tables#datagroup_trigger)
  + [sql\_trigger\_value](https://cloud.google.com/looker/docs/derived-tables#sql_trigger_value)
  + [interval\_trigger](https://cloud.google.com/looker/docs/derived-tables#interval_trigger)
  + [persist\_for](https://cloud.google.com/looker/docs/derived-tables#persist_for)

#### **datagroup\_trigger**

[Datagroups](https://cloud.google.com/looker/docs/caching-and-datagroups) are the most flexible method of creating persistence. If you have defined a [datagroup](https://cloud.google.com/looker/docs/reference/param-model-datagroup) with [sql\_trigger](https://cloud.google.com/looker/docs/reference/param-model-datagroup" \l "sql_trigger) or [interval\_trigger](https://cloud.google.com/looker/docs/reference/param-model-datagroup" \l "interval_trigger), you can use the [datagroup\_trigger](https://cloud.google.com/looker/docs/reference/param-view-datagroup-trigger) parameter to initiate the rebuilding of your PDTs.

Looker maintains the PDT in the database until its datagroup is triggered. When the datagroup is triggered, Looker rebuilds the PDT to replace the previous version. This means that, in most cases, your users will not have to wait for the PDT to be built. If a user requests data from the PDT while it is being built and the query results aren't in the cache, Looker will return data from the existing PDT until the new PDT is built. See [Caching queries](https://cloud.google.com/looker/docs/caching-and-datagroups#using_a_datagroups_caching_policy) for an overview of datagroups.

#### **sql\_trigger\_value**

The [sql\_trigger\_value](https://cloud.google.com/looker/docs/reference/param-view-sql-trigger-value) parameter triggers the regeneration of a PDT based on a SQL statement that you provide. If the result of the SQL statement is different from the previous value, the PDT is regenerated. Otherwise, the existing PDT is maintained in the database. This means that, in most cases, your users will not have to wait for the PDT to be built. If a user requests data from the PDT while it is being built, and the query results aren't in the cache, Looker will return data from the existing PDT until the new PDT is built.

#### **interval\_trigger**

The [interval\_trigger](https://cloud.google.com/looker/docs/reference/param-view-interval-trigger) parameter triggers the regeneration of a PDT based on a time interval that you provide, such as "24 hours" or "60 minutes". Similar to the sql\_trigger parameter, this means that usually the PDT will be pre-built when your users query it. If a user requests data from the PDT while it is being built, and the query results aren't in the cache, Looker will return data from the existing PDT until the new PDT is built.

#### **persist\_for**

Yet another option is to use the [persist\_for](https://cloud.google.com/looker/docs/reference/param-view-persist-for-for-derived-table) parameter to set the length of time the derived table should be stored before it is marked as expired, so that it is no longer used for queries and will be dropped from the database.

**Optimization strategies**

Because PDTs are stored in your database, you should optimize your PDTs using the following strategies, as supported by your dialect:

* [cluster\_keys](https://cloud.google.com/looker/docs/reference/param-view-cluster-keys)
* [distribution](https://cloud.google.com/looker/docs/reference/param-view-distribution)
* [distribution\_style](https://cloud.google.com/looker/docs/reference/param-view-distribution-style)
* [indexes](https://cloud.google.com/looker/docs/reference/param-view-indexes)
* [partition\_keys](https://cloud.google.com/looker/docs/reference/param-view-partition-keys)
* [sortkeys](https://cloud.google.com/looker/docs/reference/param-view-sortkeys)

ed whenever its dependent PDTs are rebuilt. So, in this case, the persist\_for PDT will be rebuilt on the schedule of its dependent PDTs. This means that persist\_for PDTs can be affected by the persistence strategy of their dependents.

## Manually rebuilding persistent tables for a query

Users can select the **Rebuild Derived Tables & Run** option from an Explore's menu to override the persistence settings and rebuild all the PDTs and [aggregate tables](https://cloud.google.com/looker/docs/aggregate_awareness#adding_aggregate_tables_to_your_project) required for the current query in the Explore:

This option is visible only to users with [develop](https://cloud.google.com/looker/docs/admin-panel-users-roles#develop) permission, and only after the Explore query has loaded.

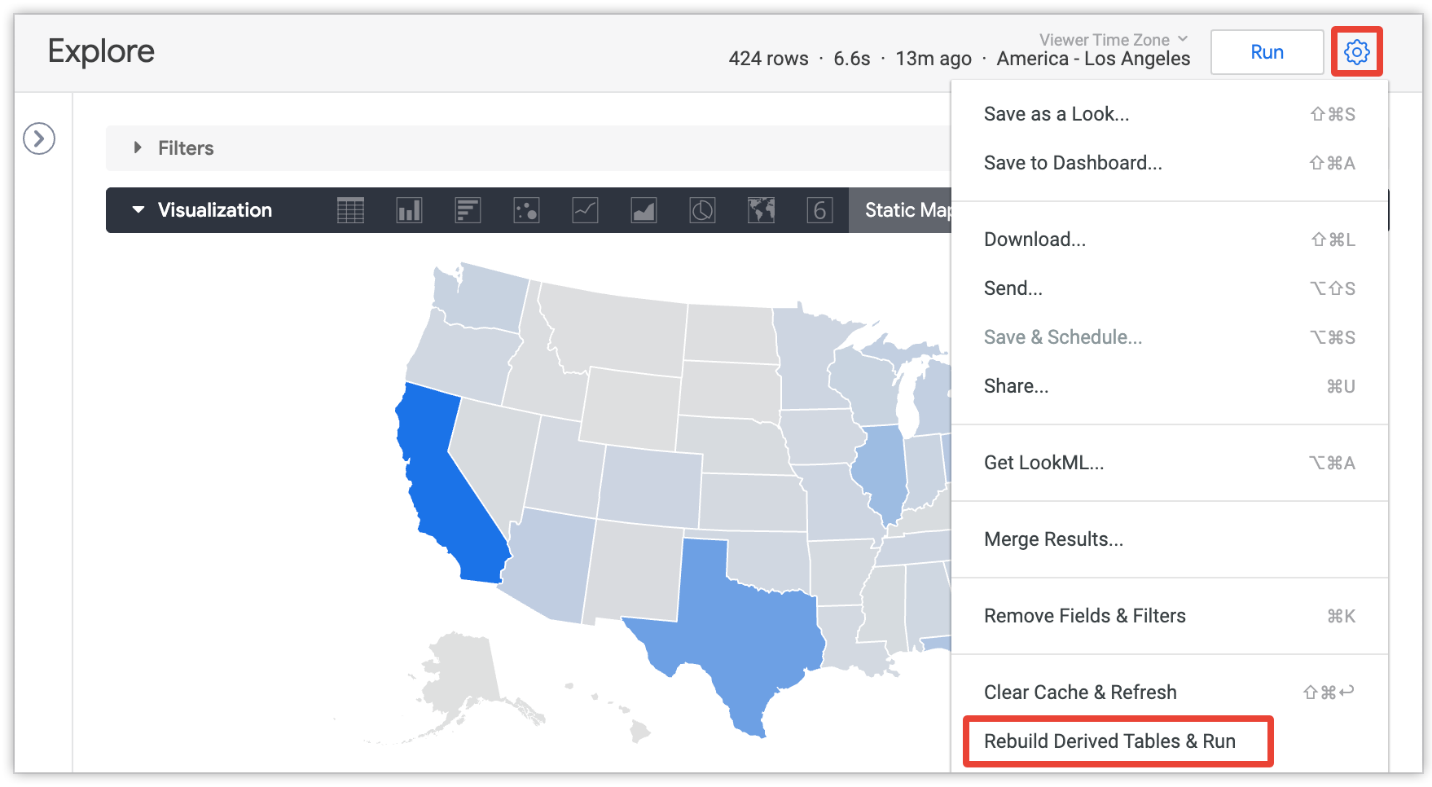
The **Rebuild Derived Tables & Run** option rebuilds all the persistent tables (all the PDTs and [aggregate tables](https://cloud.google.com/looker/docs/aggregate_awareness#adding_aggregate_tables_to_your_project)) that are required to answer the query, regardless of their persistence strategy. This includes any aggregate tables and PDTs in the current query, and it also includes any aggregate tables and PDTs that are referenced by the aggregate tables and PDTs in the current query.

In the case of [incremental PDTs](https://cloud.google.com/looker/docs/incremental-pdts), the **Rebuild Derived Tables & Run** option triggers the build of a new increment. With incremental PDTs, an increment includes the time period specified in the [increment\_key](https://cloud.google.com/looker/docs/reference/param-view-increment-key) parameter, and also the number of previous time periods specified in the [increment\_offset](https://cloud.google.com/looker/docs/reference/param-view-increment-offset) parameter, if any. See the [Incremental PDTs](https://cloud.google.com/looker/docs/incremental-pdts#example_scenarios) documentation page for some example scenarios that show how incremental PDTs build, depending on their configuration.

ed whenever its dependent PDTs are rebuilt. So, in this case, the persist\_for PDT will be rebuilt on the schedule of its dependent PDTs. This means that persist\_for PDTs can be affected by the persistence strategy of their dependents.

## Manually rebuilding persistent tables for a query

Users can select the **Rebuild Derived Tables & Run** option from an Explore's menu to override the persistence settings and rebuild all the PDTs and [aggregate tables](https://cloud.google.com/looker/docs/aggregate_awareness#adding_aggregate_tables_to_your_project) required for the current query in the Explore:



This option is visible only to users with [develop](https://cloud.google.com/looker/docs/admin-panel-users-roles#develop) permission, and only after the Explore query has loaded.

The **Rebuild Derived Tables & Run** option rebuilds all the persistent tables (all the PDTs and [aggregate tables](https://cloud.google.com/looker/docs/aggregate_awareness#adding_aggregate_tables_to_your_project)) that are required to answer the query, regardless of their persistence strategy. This includes any aggregate tables and PDTs in the current query, and it also includes any aggregate tables and PDTs that are referenced by the aggregate tables and PDTs in the current query.

In the case of [incremental PDTs](https://cloud.google.com/looker/docs/incremental-pdts), the **Rebuild Derived Tables & Run** option triggers the build of a new increment. With incremental PDTs, an increment includes the time period specified in the [increment\_key](https://cloud.google.com/looker/docs/reference/param-view-increment-key) parameter, and also the number of previous time periods specified in the [increment\_offset](https://cloud.google.com/looker/docs/reference/param-view-increment-offset) parameter, if any. See the [Incremental PDTs](https://cloud.google.com/looker/docs/incremental-pdts#example_scenarios) documentation page for some example scenarios that show how incremental PDTs build, depending on their configuration.

## Case sensitivity

LookML is case-sensitive, so be sure to match the case when referring to LookML elements. Looker will alert you if you have referred to an element that doesn't exist.