Google Cloud

Define dimensions and measure with LookML

Previously, you explored data modeling languages that can be used to create and represent semantic data models. You also learned more about LookML, the modeling language developers use when interacting with the Looker IDE. In this reading, you'll learn more about LookML, and dimensions and measures.

How LookML functions

LookML is a programming language. Different programming languages function in different ways. For example, Python is an imperative programming language, which means that programmers have to define the sequence of steps they want the system to perform to achieve the desired result. Basically, programmers have to tell the system exactly what to do in imperative programming. SQL is a declarative language, meaning that programmers need to tell the system what they want to happen. With SQL, the computer will take the necessary steps to execute that command. LookML is a SQL-based modeling language, so it's also a declarative language.

Essentially, declarative languages run on logic to apply your commands to objects within the system. Part of working with declarative languages is establishing the objects you want to query, so that the system can effectively locate the correct data. For LookML, this involves defining dimensions and measures.

Dimensions and measures in LookML

In LookML, Views represent tables that are either directly linked to the database tables, or derived from those tables. These Views contain fields called dimensions that are used to build queries in the Looker environment. Dimensions are fields that can be grouped together to filter results, including:

- Attributes associated with columns in the underlying tables
- Facts and numerical values
- Derived values from calculations

Measures are fields that use mathematical functions like **COUNT**, **SUM**, **AVG**, **MIN**, or **MAX** to aggregate dimensions. Remember, dimensions allow you to describe data, and measures allow you to combine and use dimensions in calculations.

Building dimensions and measures

Often, the dimensions contained in your Views will have a direct correlation to columns in your underlying tables.You can also build custom dimensions to help you gather data in a useful way. To build a custom dimension, you need to define it using keywords, parameters, and filters. **Parameters** are variables you can apply to interact with the data. Here's a simple dimension with just three parameters:

```
Unset
dimension: category {
      label: "Category"
      sql: TRIM(${TABLE}.category) ;;
      drill_fields: [item_name]
}
```

In this example you can see some of the elements of building dimensions:

- Keyword definition**:** This code uses the dimension keyword to start building the dimension and to name it "category." The open curly bracket on this line indicates the beginning of the parameters for this dimension.
- label**:** This parameter is how the dimension will appear to end users in the Explore page. In this case, the dimension has been labeled Category with a capital "C."
- sql**:** This parameter indicates how the value is retrieved from the database. The **SELECT** and **FROM** clauses are implied here. The ${TABLE} is a reference to the table name variable defined at the top of the View file.
- drill_fields: This is an optional parameter that defines the behavior of this value when it's clicked on by users in the Explore results. In this case, it's indicating that when a user clicks on the value for this category in a report using the Category dimension, the report will drill down to the item_name dimension values. The closed curly bracket is where the parameter definitions end.

Defining measures is very similar. For example, here is a simple measure:

```
Unset
measure: average_days_to_process {
```

```
      label: "Average Days to Process"
      type: average
      value_format_name: decimal_2
      sql: ${days_to_process} ;;
}
```

You might notice some similarities between the measure definition and the dimension definition:

- Keyword definition: the measure definition also uses a keyword to indicate that a measure is being defined. In this case, it's `average_days_to_process`. The curly bracket here marks where the parameters for this measure begin.
- `label`: This parameter means that the measure will appear as "Average Days to Process" for end users because of the label in this definition.
- `type`**:** The measure being defined here also indicates the type of measure it will be: the average.
- `value_format_name`**:** Indicates it should be `decimal_2` (a number with exactly 2 decimal places). Like the dimension measure, the parameters are closed off with the other curly bracket.
- `sql`**:** Like the previous dimension, this measure will be retrieved from the database using SQL. In this case, there is a dimension named, `days_to_process`, and this measure takes the average of values of that dimension.

Key takeaways

LookML is a declarative language that requires users to define objects within the system so that it can perform queries as defined. Part of this process involves establishing dimensions and measures which will translate to query components for users in Explores.