# LookML Generator Documentation

***Release 0.1.9***

**Joe Schmid**

**Apr 04, 2023**

# Contents

Contents:

CHAPTER 1

# LookML Generator

Programmatically generate LookML

- Free software: Apache Software License 2.0
- Documentation: https://lookml-gen.readthedocs.io.

## 1.1 Features

- Generate LookML views programmatically
- Include dimensions, dimension groups, filters, and measures in your views
- Support Persistent Derived Tables (PDTs)
- Write output to files or StringIO buffers

## 1.2 Quick Start

Install it:

```
pip install lookml-gen
```

Use it:

```
from lookmlgen.view import View
from lookmlgen.field import Dimension, DimensionGroup, Measure
from lookmlgen.base_generator import GeneratorFormatOptions

view_name = 'my_view'
v = View(view_name, sql_table_name='my_table')
v.add_field(Dimension('id', type='number', primary_key=True))
v.add_field(DimensionGroup('created'))
v.add_field(Dimension('name'))
v.add_field(Dimension('quantity', type='number'))
v.add_field(Measure('total_quantity', sql='${TABLE}.quantity', type='sum'))

with open('%s.view.lkml' % view_name, 'w') as f:
    v.generate_lookml(f, GeneratorFormatOptions(view_fields_alphabetical=False))
```

See it:

```
# STOP! This file was generated by an automated process.
# Any edits you make will be lost the next time it is
# re-generated.
view: my_view {
  sql_table_name: my_table ;;

  dimension: id {
    type: number
    primary_key: yes
    sql: ${TABLE}.id ;;
  }

  dimension_group: created {
    type: time
    timeframes: ["time", "date", "week", "month"]
    datatype: datetime
    sql: ${TABLE}.created ;;
  }

  dimension: name {
    sql: ${TABLE}.name ;;
  }

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

  measure: total_quantity {
    type: sum
    sql: ${TABLE}.quantity ;;
  }
}
```

## 1.3 TODOs

Full LookML support is far from complete right now. At the moment only very basic aspects of Views and Fields are supported and there is no support for Explores yet. However, it does cover the most common functionality, including

Persistent Derived Tables. The code can easily be extended and we'd love to get pull requests to fill out additional functionality.

## 1.4 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

# Installation

## 2.1 Stable release

To install LookML Generator, run this command in your terminal:

```
$ pip install lookml-gen
```

This is the preferred method to install LookML Generator, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

## 2.2 From sources

The sources for LookML Generator can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/symphonyrm/lookml-gen
```

Or download the tarball:

```
$ curl  -OL https://github.com/symphonyrm/lookml-gen/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

# CHAPTER 3

# Usage

To use LookML Generator in a project:

```
from lookmlgen.view import View
from lookmlgen.field import Dimension, DimensionGroup, Measure
from lookmlgen.base_generator import GeneratorFormatOptions

view_name = 'my_view'
v = View(view_name, sql_table_name='my_table')
v.add_field(Dimension('id', type='number', primary_key=True))
v.add_field(DimensionGroup('created'))
v.add_field(Dimension('name'))
v.add_field(Dimension('quantity', type='number'))
v.add_field(Measure('total_quantity', sql='${TABLE}.quantity', type='sum'))

with open('%s.view.lkml' % view_name, 'w') as f:
    v.generate_lookml(f, GeneratorFormatOptions(view_fields_alphabetical=False))
```

The contents of the file 'my_view.view.lkml' will be:

```
# STOP! This file was generated by an automated process.
# Any edits you make will be lost the next time it is
# re-generated.
view: my_view {
  sql_table_name: my_table ;;

  dimension: id {
    type: number
    primary_key: yes
    sql: ${TABLE}.id ;;
  }

  dimension_group: created {
    type: time
    timeframes: ["time", "date", "week", "month"]
```

(continues on next page)

(continued from previous page)

```
    datatype: datetime
    sql: ${TABLE}.created ;;
  }

  dimension: name {
    sql: ${TABLE}.name ;;
  }

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

  measure: total_quantity {
    type: sum
    sql: ${TABLE}.quantity ;;
  }
}
```

CHAPTER 4

# lookmlgen package

## 4.1 Submodules

## 4.2 lookmlgen.base_generator module

File name: base_generator.py Author: joeschmid Date created: 4/8/17

**class** lookmlgen.base_generator.**BaseGenerator**(*file=None*, *format_options=<lookmlgen.base_generator.GeneratorFormatOptions object>*)

Bases: `object`

Abstract base class for any subclass that generates LookML

**Parameters**

- **file** (*File handle or StringIO object*) – File handle of a file open for writing or a StringIO object
- **format_options** (*GeneratorFormatOptions*) – Formatting options to use during generation

**generate_lookml**(*file=None*, *format_options=None*)

Implement this method in subclasses to generate LookML

**Parameters**

- **file** (*File handle or StringIO object*) – File handle of a file open for writing or a StringIO object
- **format_options** (*GeneratorFormatOptions*) – Formatting options to use during generation

**class** lookmlgen.base_generator.**GeneratorFormatOptions**(*indent_spaces=2*, *newline_between_items=True*, *omit_default_field_type=True*, *view_fields_alphabetical=True*, *warning_header_comment='# STOP! This file was generated by an automated process.n# Any edits you make will be lost the next time it isn# re-generated.n'*, *omit_time_frames_if_not_set=False*)

Bases: object

Specify formatting options to be used during LookML generation

**Parameters**

- **indent_spaces** (*int*) – Number of spaces to indent
- **newline_between_items** (*bool*) – Add a newline between items
- **omit_default_field_type** (*bool*) – Leave out 'type: string'
- **warning_header_comment** (*string*) – Text to use as a comment as the top of the file warning the user that the file will get overwritten
- **omit_time_frames_if_not_set** (*bool*) – If no time frame is specified for a dimension_group field, omit the time_frames parameter. If timeframes is not included every timeframe option will be added to the dimension group.

## 4.3 lookmlgen.cli module

## 4.4 lookmlgen.field module

File name: field.py Author: joeschmid Date created: 4/9/17

**class** lookmlgen.field.**Dimension**(*name*, *primary_key=None*, ***kwargs*)

Bases: *lookmlgen.field.Field*

Generates LookML for a dimension field in a *View*

**Parameters**

- **name** (*string*) – Name of the dimension
- **primary_key** (*bool*) – Flag to designate the field as a primary key

**class** lookmlgen.field.**DimensionGroup**(*name*, *timeframes=None*, *datatype='datetime'*, ***kwargs*)

Bases: *lookmlgen.field.Field*

**Generates LookML for a dimension_group field in a** *View*

**Parameters**

- **name** (*string*) – Name of the dimension group
- **timeframes** (*list of strings*) – Timeframes for the group

- **datatype** (*string*) – Datatype for the group, defaults to 'datetime'

**class** lookmlgen.field.**Field**(*field_type*, *name*, *type='string'*, *label=None*, *sql=None*, *hidden=None*, *file=None*, *group_label=None*, *description=None*, ***kwargs*)

Bases: *lookmlgen.base_generator.BaseGenerator*

Base class used to generate fields within a *View*

**Parameters**

- **field_type** (a class variable from *FieldType*) – Name of the view
- **name** (*string*) – Name of the field
- **type** (*string*) – Type of the field contents, e.g. string, number, etc.
- **label** (*string*) – Label to use when displaying the field
- **sql** (*string*) – SQL snippet for the field
- **hidden** (*bool*) – Flag to designate the field as hidden
- **file** (*File handle or StringIO object*) – File handle of a file open for writing or a StringIO object
- **group_label** (*string*) – Group label to use for grouping the field
- **description** (*string*) – Field description that is show if a user hovers over the help link in the field picker

**generate_lookml**(*file=None*, *format_options=None*)

Writes LookML for a field to a file or StringIO buffer.

**Parameters**

- **file** (*File handle or StringIO object*) – File handle of a file open for writing or a StringIO object
- **format_options** (*GeneratorFormatOptions*) – Formatting options to use during generation

**class** lookmlgen.field.**FieldType**

Bases: object

Enum-style class used to specify known Field types

**DIMENSION = 1**

**DIMENSION_GROUP = 2**

**FILTER = 3**

**MEASURE = 4**

**classmethod type_name**(*type_id*)

**class** lookmlgen.field.**Filter**(*name*, ***kwargs*)

Bases: *lookmlgen.field.Field*

**Generates LookML for a filter field in a** *View*

**Parameters** **name** (*string*) – Name of the filter

**class** lookmlgen.field.**Measure**(*name*, ***kwargs*)

Bases: *lookmlgen.field.Field*

**Generates LookML for a measure field in a** *View*

**Parameters** **name** (*string*) – Name of the measure

## 4.5 lookmlgen.util module

File name: util.py Author: joeschmid Date created: 4/16/17

lookmlgen.util.**indent**(*s*, *num_spaces*)

## 4.6 lookmlgen.view module

File name: view.py Author: joeschmid Date created: 4/8/17

**class** lookmlgen.view.**DerivedTable**(*sql*, *sql_trigger_value=None*, *indexes=None*, *file=None*)
Bases: *lookmlgen.base_generator.BaseGenerator*

Generates the LookML View parameters to support derived tables, including persistent derived tables (PDTs).

**Parameters**

- **sql** (*string*) – SQL statement to execute
- **sql_trigger_value** (*string*) – SQL to determine when to trigger build
- **indexes** (*list of strings*) – List of coluxn names to use as indexes
- **file** (*File handle or StringIO object*) – File handle of a file open for writing or a StringIO object

**generate_lookml**(*file=None*, *format_options=None*)
Writes LookML for a derived table to a file or StringIO buffer.

**Parameters**

- **file** (*File handle or StringIO object*) – File handle of a file open for writing or a StringIO object
- **format_options** (*GeneratorFormatOptions*) – Formatting options to use during generation

**class** lookmlgen.view.**View**(*name*, *label=None*, *sql_table_name=None*, *file=None*)
Bases: *lookmlgen.base_generator.BaseGenerator*

Generates a LookML View

Initialize a View object with your parameters, add Fields such as *Dimension*, *Measure*, *DimensionGroup*, and *Filter*, and then generate LookML for the view using *generate_lookml()*

**Parameters**

- **name** (*string*) – Name of the view
- **label** (*string*) – Label to use for the view (may contain spaces)
- **sql_table_name** (*list of strings*) – Name of the SQL table to use in the view
- **file** (*File handle or StringIO object*) – File handle of a file open for writing or a StringIO object

**add_field**(*field*)
Adds a *Field* object to a *View*

**generate_lookml**(*file=None*, *format_options=None*)
Writes LookML for the view to a file or StringIO buffer.

**Parameters**

- **file** (*File handle or StringIO object*) – File handle of a file open for writing or a StringIO object
- **format_options** (*GeneratorFormatOptions*) – Formatting options to use during generation

**set_derived_table**(*derived_table*)
Adds a *DerivedTable* object to a *View*

## 4.7 Module contents

CHAPTER 5

# Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

## 5.1 Types of Contributions

### 5.1.1 Report Bugs

Report bugs at https://github.com/symphonyrm/lookml-gen/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### 5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

### 5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

### 5.1.4 Write Documentation

LookML Generator could always use more documentation, whether as part of the official LookML Generator docs, in docstrings, or even on the web in blog posts, articles, and such.

### 5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/symphonyrm/lookml-gen/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

## 5.2 Get Started!

Ready to contribute? Here's how to set up *lookml-gen* for local development.

1. Fork the *lookml-gen* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/lookml-gen.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv lookml-gen
$ cd lookml-gen/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 lookml-gen tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

## 5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/symphonyrm/lookml-gen/pull_requests and make sure that the tests pass for all supported Python versions.

## 5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_view
```

CHAPTER 6

# Credits

## 6.1 Development Lead

- Joe Schmid <jschmid@symphonyrm.com>

## 6.2 Contributors

- Lara Stoll <lstoll@symphonyrm.com>
- Cassandra Teale <cteale@symphonyrm.com>

CHAPTER 7

# History

## 7.1 0.1.0 (2017-04-17)

- First release on PyPI.

## 7.2 0.1.1 (2017-04-17)

- Switch to lookmlgen for module name

## 7.3 0.1.2 (2017-04-18)

- Move primary_key from Field to Dimension
- Remove stub for command line use
- Add docstrings

## 7.4 0.1.3 (2017-04-20)

- Default sql parameter of fields to ${TABLE}.field_name

## 7.5 0.1.4 (2017-04-20)

- Support sql_table_name
- Add formatting option for alphabetical view fields or not

## 7.6 0.1.5 (2017-04-24)

- Rename add_derived_table method to set_derived_table in View

## 7.7 0.1.7 (2017-06-20)

- Added formatting option to omit timeframe generating params if they're not set
- Update pytest to 3.1.2, cryptography to 1.9, sphinx to 1.6.2

## 7.8 0.1.8 (2017-06-23)

- Add a description parameter to fields

## 7.9 0.1.9 (2017-06-23)

- Bug fix for extra newlines between field type sections

CHAPTER 8

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## l

# Index