
#JOINdata
@lookerdata

# Hands-on Lab: Extends: The End of Endless Rewriting

looker
Kevin McCarthy
Professional Services

#JOINdata
@lookerdata

# looker.com/hol

Select the **Extends: The End of Endless Rewriting** lab in the drop-down

# Kevin McCarthy

Principal Consultant, Professional Services



looker

# Extends

Why to use them and how they work

# Why use extensions

## Create building blocks for model development

Modularize code to create building blocks and features

- Write DRY (Don’t Repeat Yourself) code
- Make changes more easily
- Ensure consistency across code
- Manage different field sets or use cases for different users more easily

# What are extensions?

## Apply object-oriented coding principles to LookML

Extend a LookML object = combine its contents with another LookML object

- Views
- Explores
- LookML dashboards

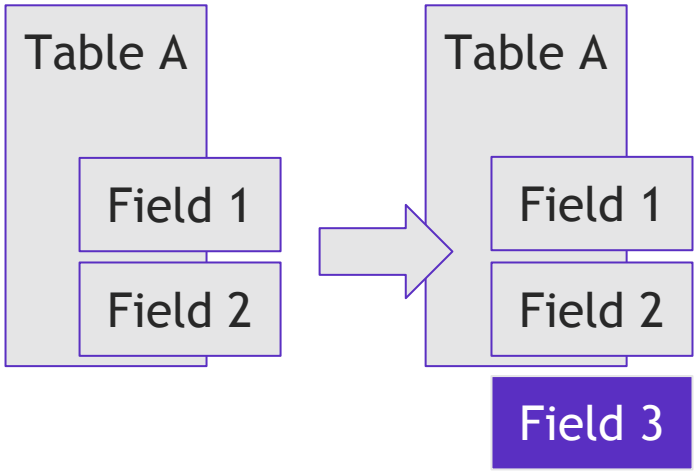


looker

# How extensions work

## Adding fields to a view

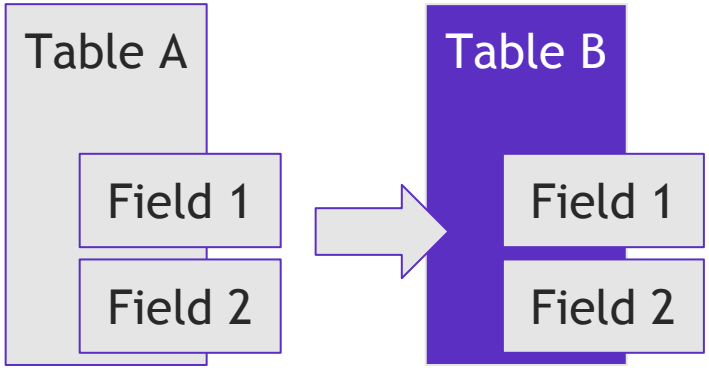
A view can be extended to include additional fields




**Original** **Extended**

## Changing the table of a view

A view can be extended to change the table it's pointing to by overriding an object's parameters




**Original** **Extended**

# The next level: creating templates

## Do more with extends

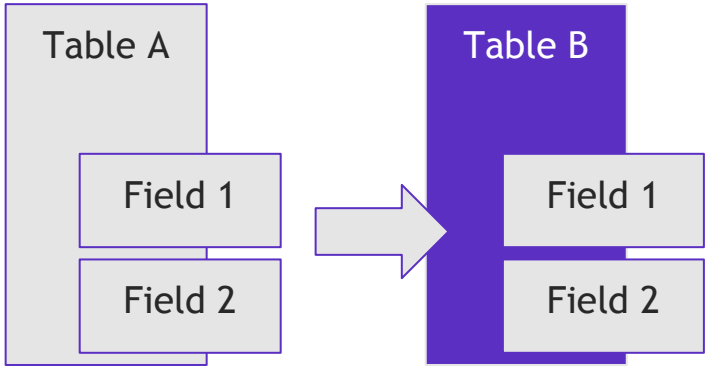


looker

# Remember… How extensions work

## Changing the table of a view

A view can be extended to change the table it's pointing to by overriding an object's parameters




**Original** **Extended**

# Generalized: all about that base

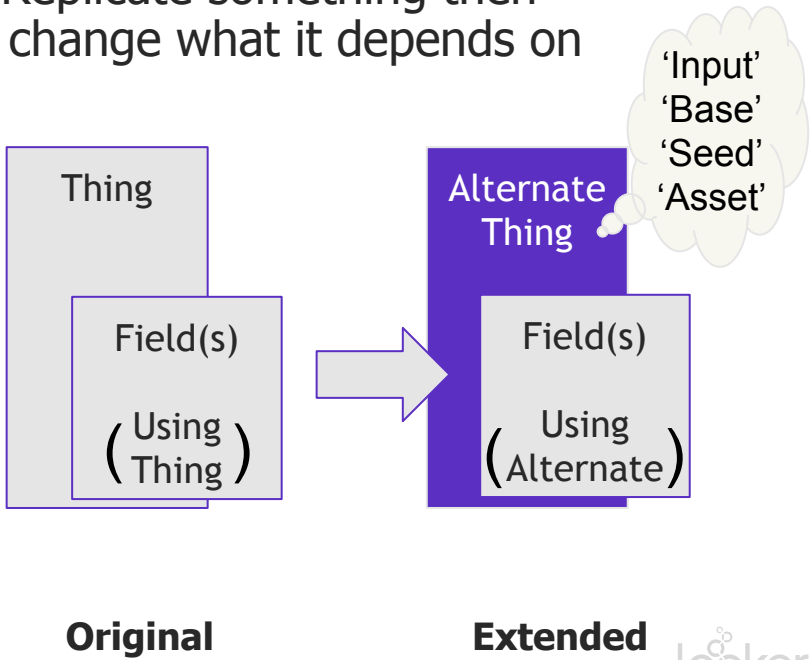
## Adding Fields to a View

A View can be extended to include additional Fields




## Changing the base

Replicate something then change what it depends on






looker

# Examples

## Applying features using extends

# Pseudo code example

## Step 1: Create a feature template

```
view: add_feature_x {

#Create the base asset/placeholder(s)
 1A) dimension:
input_placeholder{sql:override_me!;;}

 1B) Apply some complex code that builds
something based on input_placeholder
(e.g a reference in it's sql: parameter)

#Create / Expose Feature Rich Output
Fields
 1C) dimension:with_feature_applied {...}
}
```

## Step 2: Apply the feature template

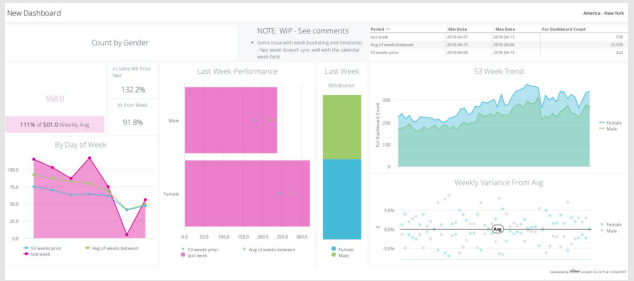
```
view: some_other_view {

#Override the base asset/placeholder(s)
2A) dimension:
input_placeholder{sql:${some_ref};;}

2B) extends: [add_feature_x]



2C)#Feature-rich fields are automatically
#inherited and updated for this context
}

2D) Profit!
```



looker

# Example feature types

| | Description | Examples |
|---|---|---|
| Functions & Templated Fields | Create a field that has some special features built on | • Apply min/max bounds to a field<br>• Safe Divide (handle divide by zero)<br>• Business days between two dates<br>• An 'Emojis Pyramid' format, etc<br>• Profiling Helper (shows min, max, count of nulls, etc, of a dim)<br>• Add a tooltip based on another field<br>• Rank of `${foo}` within `${bar}` by `${measure}` |
| Utilities & Standards | Other repeated patterns or commonly used logic | • `Count(primary key not null)` instead of `count(*)`<br>• Many Explores inherit a standard set of joins from a given view<br>• Leverage complex combinations of user attribute values<br>• Date Helpers based on Liquid's Now |
| Dashboards | Generate many versions from a template | • Last Week vs historical by `${dimension}` and `${measure}` |



# Exercise

## Applying features using extends

# Practice feature extends: Custom Tiers

## The Business Scenario:

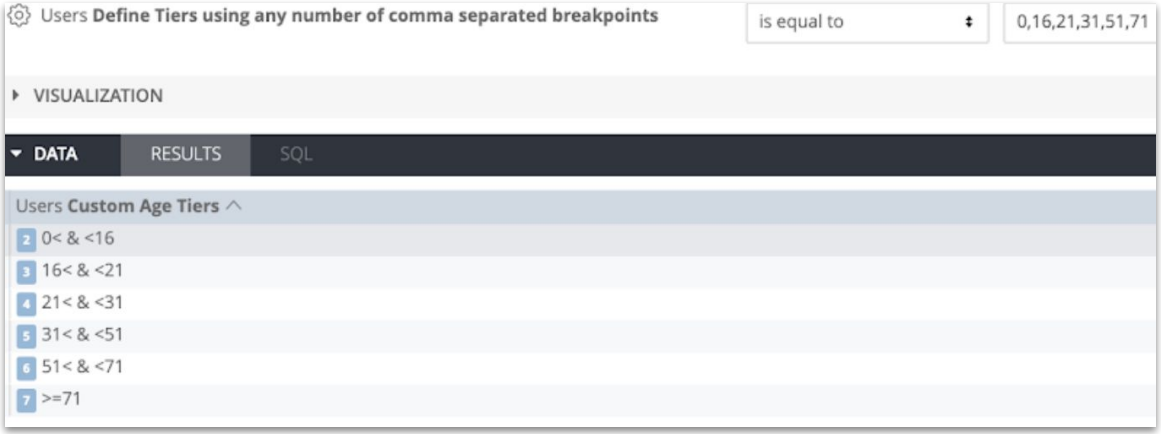
- Your data has a large number of numeric dimensions with many distinct numeric values.
- The range of these values can vary dramatically depending what filters are used, etc.

## The Ask:

- For each such dimension, add a Custom Tiers feature so users can define their tiers on the fly.

## The Task:

- Implement with extends to ensure consistency and easy future enhancements.



# Practice feature extends: Custom Tiers

## The Task:

- Enable custom tiering feature on two numeric dimensions.
- Use extends, to ensure consistency and easy future enhancements.

## The Exercise:

- We have already developed the feature.
- Implement two 'versions' of the Custom Tiers feature: **User Age Tiers** & **Sale Price Tiers**.

## The Steps (Attendees follow steps from here: https://looker.com/hol):

1. For User Age, go to the users view in the extends_lab project.
2. Within the users view file, add the following parameter:
3. Add the extends parameter to the users view object:
4. Add a dimension called field_to_tier. Set/Override the sql parameter label:

```
include: "/features/*"
extends: [custom_tiers]
dimension: field_to_tier {
 label: "Age Tiers"
 sql: ${age} ;;
}
```

1. Check your work
2. Repeat the steps to add the 'Sale Price Tiers' in the Order Items view.

# Practice feature extends: Custom Tiers

## The existing Custom Tiers feature

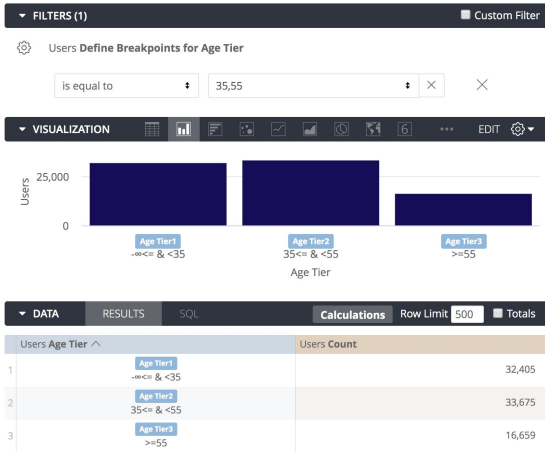
custom_tiers.view

```
1  ############
2  # Feature Description:
3  ## User specifies any number of arbitrary cutoffs in a parameter
4  ## takes field_to_tier's SQL field and groups data into tiers accordingly
5  #
6  # How to enable the feature:
7  ## Add the following to a view and adjust label and sql parameters as desired
8  # extends: [custom_tiers]
9  # dimension: field_to_tier {
10 #   label: "Label for Resulting Tiered Field"
11 #   sql: ${field_to_be_tiered} ;;
12 # }
13
14 view: custom_tiers {
15
16   extension: required
17
18   parameter: compare_cutoffs__arbitrary{
19     label: "Define Breakpoints for {{ field_to_tier._label | replace: view_name_labe
20     description: "Define Tiers using any number of comma separated breakpoints"
21     suggestions: ["0,10,100","-50,50,150,300"]
22     default_value: "0,10,100"
23     type:string
24   }
25
26   #field used for sorting and for Group Number Icon (in output field's HTML)
27   dimension: tier_number {
28     hidden: yes
```

## Apply to something in your model

users.view

```
1  include: "/features/*"
2  view: users {
3    extends: [custom_tiers]
4    dimension: field_to_tier {
5      sql: ${age} ;;
6      label: "Age Tiers"
7    }
8
```






looker

# Questions?


looker


Thank you
Rate this session in
the JOIN mobile app
#JOINdata
@lookerdata