SmartyGrants
Software, data science & grantmaking intelligence

An enterprise of:
ourcommunity.com.au
Where not-for-profits go for help

SmartyGrants Analytics Technical Documentation

Version 1.5
SmartyGrants Version 8.26.25
8 July 2025

SmartyGrants
552 Victoria Street
Wurundjeri Country
North Melbourne 3051
Victoria Australia

PO Box 354
North Melbourne 3051
Victoria Australia
ABN 24 094 608 705

+61 3 9320 6888
service@smartygrants.com.au
www.smartygrants.com

Denis Moriarty AM
Group Managing Director
Carol Schwartz AO
Chair

Table of Contents

1 Using This Guide

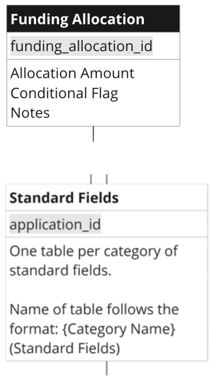
This document contains an in-depth breakdown of the data model used in SmartyGrants Analytics.

- Users seeking a short guide to combining fields in a widget can skip to the section "Designing a Widget."
- Other sections can be referred to for the "why" behind the guidelines contained in "Designing a Widget."

2 Definitions

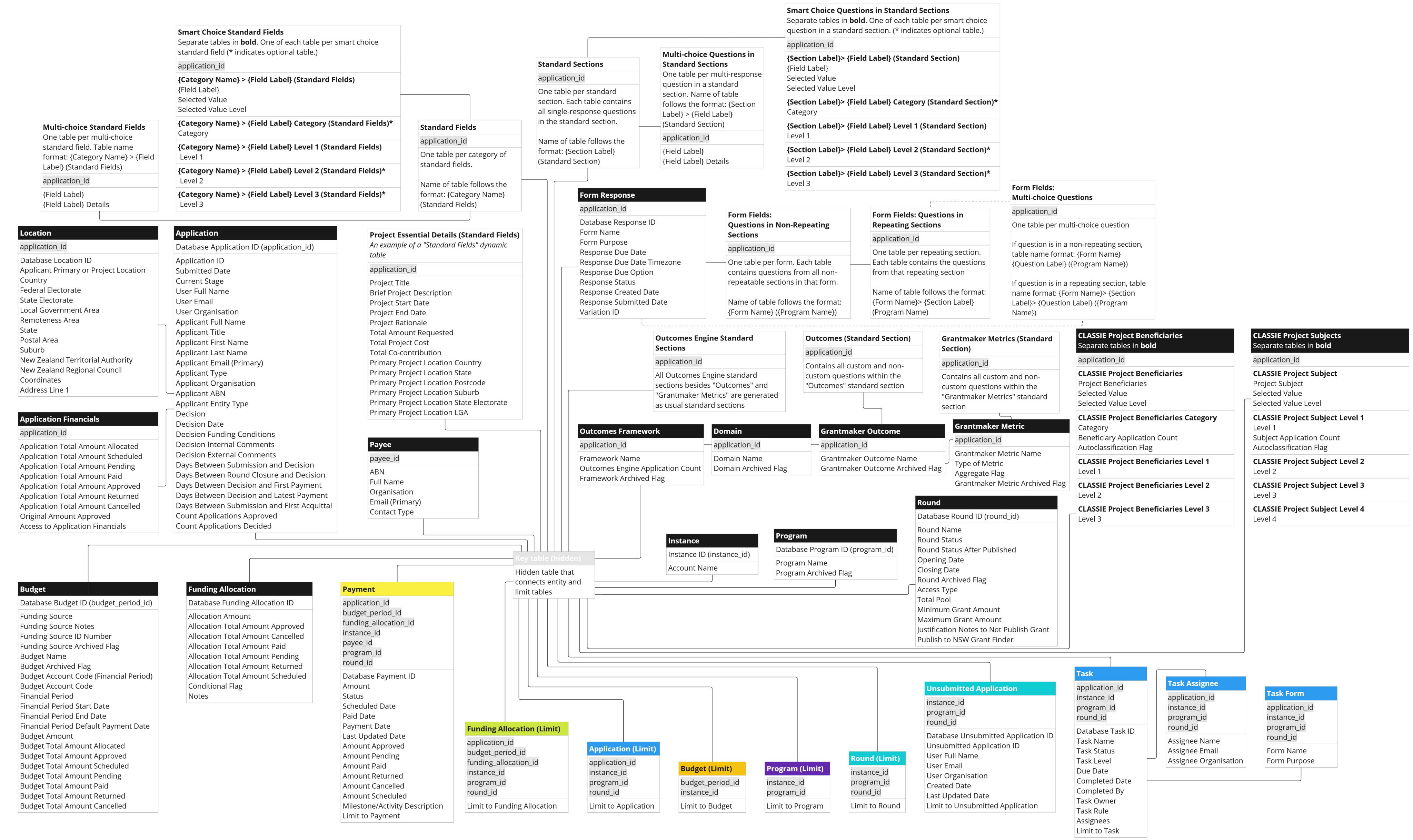
	Definition	**Examples**
Entity	An object or concept in SmartyGrants	An application
Key	A unique ID which identifies an entity.	Every application has a unique application_id
Key Combination	An entity's key combined with the keys of other entities that it is "nested" under or belongs to. The supplied data model diagram provides the key combination of all entities.	An application belongs to a round, which belongs to a program, which belongs to an instance. The key combination for an application is Application ID, Round ID, Program ID, Instance ID
Permutation	A "real" case of a key combination, made up of IDs assigned in the SmartyGrants database	A permutation of an application key combination could be {APPID01, ROUNDID01, PROGRAMID01, INSTANCEID01}
Widget query	The code that SmartyGrants Analytics runs in the background to build a widget	-

3 Interpreting the Data Model


Funding Allocation
funding_allocation_id
Allocation Amount
Conditional Flag
Notes
Standard Fields
application_id
One table per category of standard fields.
Name of table follows the format: {Category Name} (Standard Fields)

Black tables	Entity tables, the top half of boxes contain a *key*
Coloured tables	Limit Tables, the top half of boxes contain a *key combination*
White tables	Dynamic tables. There may be multiple of each table represented in the data model (e.g. several "Standard Fields" tables if an instance has several standard field categories). {} indicates a dynamic table name. For example, {Category Name} Standard Fields is the naming convention for standard field tables. If an instance contains a category called "Volunteer Information", the corresponding table would be Volunteer Information (Standard Fields)
Greyed out fields	Fields hidden from the user

4 Data Model Diagram



Smart Choice Standard Fields
Separate tables in **bold**. One of each table per smart choice standard field (* indicates optional table.)
application_id
{Category Name} > {Field Label} (Standard Fields)
{Field Label}
Selected Value
Selected Value Level
{Category Name} > {Field Label} Category (Standard Fields)*
Category
{Category Name} > {Field Label} Level 1 (Standard Fields)
Level 1
{Category Name} > {Field Label} Level 2 (Standard Fields)*
Level 2
{Category Name} > {Field Label} Level 3 (Standard Fields)*
Level 3

Multi-choice Standard Fields
One table per multi-choice standard field. Table name format: {Category Name} > {Field Label} (Standard Fields)
application_id
{Field Label}
{Field Label} Details

Standard Fields
application_id
One table per category of standard fields.

Name of table follows the format: {Category Name} (Standard Fields)

Standard Sections
application_id
One table per standard section. Each table contains all single-response questions in the standard section.

Name of table follows the format: {Section Label} (Standard Section)

Multi-choice Questions in Standard Sections
One table per multi-response question in a standard section. Name of table follows the format: {Section Label} > {Field Label} (Standard Section)
application_id
{Field Label}
{Field Label} Details

Smart Choice Questions in Standard Sections
Separate tables in **bold**. One of each table per smart choice question in a standard section. (* indicates optional table.)
application_id
{Section Label}> {Field Label} (Standard Section)
{Field Label}
Selected Value
Selected Value Level
{Section Label}> {Field Label} Category (Standard Section)*
Category
{Section Label}> {Field Label} Level 1 (Standard Section)
Level 1
{Section Label}> {Field Label} Level 2 (Standard Section)*
Level 2
{Section Label}> {Field Label} Level 3 (Standard Section)*
Level 3

Location
application_id
Database Location ID
Applicant Primary or Project Location
Country
Federal Electorate
State Electorate
Local Government Area
Remoteness Area
State
Postal Area
Suburb
New Zealand Territorial Authority
New Zealand Regional Council
Coordinates
Address Line 1

Application
Database Application ID (application_id)
Application ID
Submitted Date
Current Stage
User Full Name
User Email
User Organisation
Applicant Full Name
Applicant Title
Applicant First Name
Applicant Last Name
Applicant Email (Primary)
Applicant Type
Applicant Organisation
Applicant ABN
Applicant Entity Type
Decision
Decision Date
Decision Funding Conditions
Decision Internal Comments
Decision External Comments
Days Between Submission and Decision
Days Between Round Closure and Decision
Days Between Decision and First Payment
Days Between Decision and Latest Payment
Days Between Submission and First Acquittal
Count Applications Approved
Count Applications Decided

Project Essential Details (Standard Fields)
An example of a "Standard Fields" dynamic table
application_id
Project Title
Brief Project Description
Project Start Date
Project End Date
Project Rationale
Total Amount Requested
Total Project Cost
Total Co-contribution
Primary Project Location Country
Primary Project Location State
Primary Project Location Postcode
Primary Project Location Suburb
Primary Project Location State Electorate
Primary Project Location LGA

Form Response
application_id
Database Response ID
Form Name
Form Purpose
Response Due Date
Response Due Date Timezone
Response Due Option
Response Status
Response Created Date
Response Submitted Date
Variation ID

Form Fields: Questions in Non-Repeating Sections
application_id
One table per form. Each table contains questions from all non-repeatable sections in that form.

Name of table follows the format: {Form Name} ({Program Name})

Form Fields: Questions in Repeating Sections
application_id
One table per repeating section. Each table contains the questions from that repeating section

Name of table follows the format: {Form Name}> {Section Label} (Program Name)

Form Fields: Multi-choice Questions
application_id
One table per multi-choice question

If question is in a non-repeating section, table name format: {Form Name}> {Question Label} ({Program Name})

If question is in a repeating section, table name format: {Form Name}> {Section Label}> {Question Label} ({Program Name})

Application Financials
application_id
Application Total Amount Allocated
Application Total Amount Scheduled
Application Total Amount Pending
Application Total Amount Paid
Application Total Amount Approved
Application Total Amount Returned
Application Total Amount Cancelled
Original Amount Approved
Access to Application Financials

Payee
payee_id
ABN
Full Name
Organisation
Email (Primary)
Contact Type

Outcomes Engine Standard Sections
application_id
All Outcomes Engine standard sections besides "Outcomes" and "Grantmaker Metrics" are generated as usual standard sections

Outcomes (Standard Section)
application_id
Contains all custom and non-custom questions within the "Outcomes" standard section

Grantmaker Metrics (Standard Section)
application_id
Contains all custom and non-custom questions within the "Grantmaker Metrics" standard section

CLASSIE Project Beneficiaries
Separate tables in **bold**
application_id
CLASSIE Project Beneficiaries
Project Beneficiaries
Selected Value
Selected Value Level
CLASSIE Project Beneficiaries Category
Category
Beneficiary Application Count
Autoclassification Flag
CLASSIE Project Beneficiaries Level 1
Level 1
CLASSIE Project Beneficiaries Level 2
Level 2
CLASSIE Project Beneficiaries Level 3
Level 3

CLASSIE Project Subjects
Separate tables in **bold**
application_id
CLASSIE Project Subject
Project Subject
Selected Value
Selected Value Level
CLASSIE Project Subject Level 1
Level 1
Subject Application Count
Autoclassification Flag
CLASSIE Project Subject Level 2
Level 2
CLASSIE Project Subject Level 3
Level 3
CLASSIE Project Subject Level 4
Level 4

Outcomes Framework
application_id
Framework Name
Outcomes Engine Application Count
Framework Archived Flag

Domain
application_id
Domain Name
Domain Archived Flag

Grantmaker Outcome
application_id
Grantmaker Outcome Name
Grantmaker Outcome Archived Flag

Grantmaker Metric
application_id
Grantmaker Metric Name
Type of Metric
Aggregate Flag
Grantmaker Metric Archived Flag

Round
Database Round ID (round_id)
Round Name
Round Status
Round Status After Published
Opening Date
Closing Date
Round Archived Flag
Access Type
Total Pool
Minimum Grant Amount
Maximum Grant Amount
Justification Notes to Not Publish Grant
Publish to NSW Grant Finder

Instance
Instance ID (instance_id)
Account Name

Program
Database Program ID (program_id)
Program Name
Program Archived Flag

Key table (hidden)
Hidden table that connects entity and limit tables

Budget
Database Budget ID (budget_period_id)
Funding Source
Funding Source Notes
Funding Source ID Number
Funding Source Archived Flag
Budget Name
Budget Archived Flag
Budget Account Code (Financial Period)
Budget Account Code
Financial Period
Financial Period Start Date
Financial Period End Date
Financial Period Default Payment Date
Budget Amount
Budget Total Amount Allocated
Budget Total Amount Approved
Budget Total Amount Scheduled
Budget Total Amount Pending
Budget Total Amount Paid
Budget Total Amount Returned
Budget Total Amount Cancelled

Funding Allocation
Database Funding Allocation ID
Allocation Amount
Allocation Total Amount Approved
Allocation Total Amount Cancelled
Allocation Total Amount Paid
Allocation Total Amount Pending
Allocation Total Amount Returned
Allocation Total Amount Scheduled
Conditional Flag
Notes

Payment
application_id
budget_period_id
funding_allocation_id
instance_id
payee_id
program_id
round_id
Database Payment ID
Amount
Status
Scheduled Date
Paid Date
Payment Date
Last Updated Date
Amount Approved
Amount Pending
Amount Paid
Amount Returned
Amount Cancelled
Amount Scheduled
Milestone/Activity Description
Limit to Payment

Funding Allocation (Limit)
application_id
budget_period_id
funding_allocation_id
instance_id
program_id
round_id
Limit to Funding Allocation

Application (Limit)
application_id
instance_id
program_id
round_id
Limit to Application

Budget (Limit)
budget_period_id
instance_id
Limit to Budget

Program (Limit)
instance_id
program_id
Limit to Program

Round (Limit)
instance_id
program_id
round_id
Limit to Round

Unsubmitted Application
instance_id
program_id
round_id
Database Unsubmitted Application ID
Unsubmitted Application ID
User Full Name
User Email
User Organisation
Created Date
Last Updated Date
Limit to Unsubmitted Application

Task
application_id
instance_id
program_id
round_id
Database Task ID
Task Name
Task Status
Task Level
Due Date
Completed Date
Completed By
Task Owner
Task Rule
Assignees
Limit to Task

Task Assignee
application_id
instance_id
program_id
round_id
Assignee Name
Assignee Email
Assignee Organisation

Task Form
application_id
instance_id
program_id
round_id
Form Name
Form Purpose

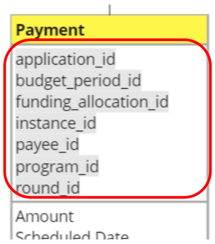
5 Designing a Widget: Steps

Not all tables in SmartyGrants Analytics can be used together in a widget. In some cases, the relationship between tables may mean no results will display. In other cases, the tool will allow users to use multiple tables in a widget, but without an appropriate filter, results may not be as expected. To design a widget correctly, follow the three steps below. In the next section, these steps are demonstrated in three worked examples of increasing complexity.

1. Find the tables containing the fields you will use

Fields in the data model are organised around entities, which are objects or concepts in SmartyGrants such as an application, round, program or budget. For guidance on how fields related to financial reporting are split between the application, payment, funding allocation and budget tables, refer to the section "Financial Reporting". To check the definitions of fields, refer to the data dictionary.

2. Check it is possible to combine these tables in a single widget

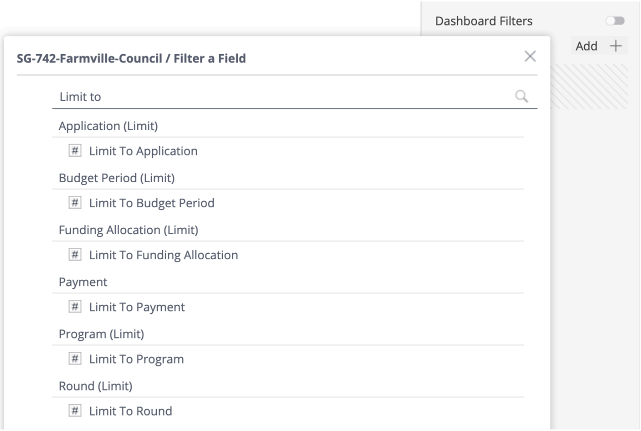


Payment
application_id
budget_period_id
funding_allocation_id
instance_id
payee_id
program_id
round_id
Amount
Scheduled Date

Once you have determined the tables you will use in your widget, check they can be combined in a single widget through the following rules. These rules reference the *key* and *key combination* of tables, found in the top half of tables in the data model diagram.

	Rule	Examples
Combining more than one entity table	Fields from two different entity tables can be used together in a widget if: • there is a limit table whose key combination contains the keys of both entities • OR both entity tables share the same key	Fields in the Program table can be used to group fields in the Round table because Program ID and Round ID are both used in the key combination of the Round (Limit) table: {Instance ID, Program ID, Round ID}.
Combining entity and limit tables	Fields in limit tables can be grouped by: • fields in any entity table whose key is contained in the limit table's key combination • fields within the same limit table.	Fields in the Payment table can be grouped by fields in the Application table because Application ID is used in the key combination of the Payment table.

3. Select a Limit To filter



SG-742-Farmville-Council / Filter a Field

Limit to

Application (Limit)
- Limit To Application

Budget Period (Limit)
- Limit To Budget Period

Funding Allocation (Limit)
- Limit To Funding Allocation

Payment
- Limit To Payment

Program (Limit)
- Limit To Program

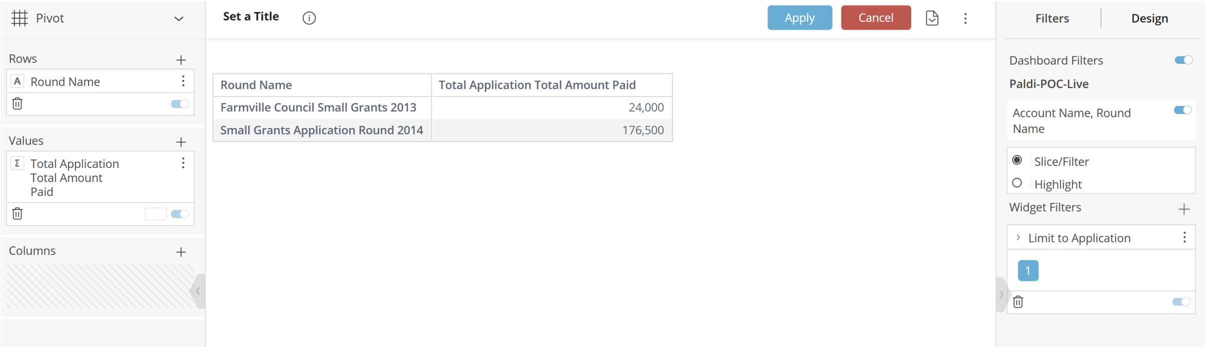
Round (Limit)
- Limit To Round

Every widget requires a “Limit to <entity>” field from a limit table applied as a widget filter. For an explanation of why this filter is needed, see the section “Limit Tables.”

To select the correct “limit to” filter, consider the entity being described by the field that the widget is aggregating (i.e. summing, counting). The entity being described is usually in the name of the table that the field comes from. For example, if a widget is summing “Original Amount Approved” from the Application Table, select “Limit to Application” for the widget filter.

Note

A “Limit to” filter can affect the tables which can be used in the widget. This is because a limit table can only be used in a widget with tables whose key/key combinations are all contained in the limit table’s key combination. After selecting a “Limit to” filter, it may be necessary to return to Step 2 and reconsider whether tables can be used in the widget. See the section “Widget design troubleshooting/FAQs” for an example.



Rows: Round Name

Values: Total Application Total Amount Paid

Round Name	Total Application Total Amount Paid
Farmville Council Small Grants 2013	24,000
Small Grants Application Round 2014	176,500

Dashboard Filters

Paldi-POC-Live

Account Name, Round Name

● Slice/Filter

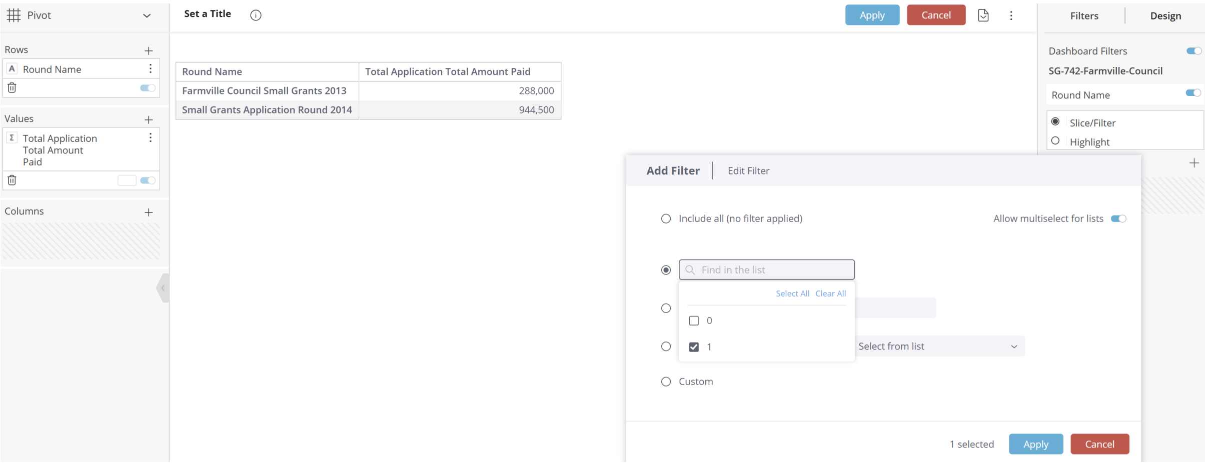
○ Highlight

Widget Filters

Limit to Application

1

To apply a “limit to” field as a widget filter, set it to include values equalling 1.



Rows: Round Name

Values: Total Application Total Amount Paid

Round Name	Total Application Total Amount Paid
Farmville Council Small Grants 2013	288,000
Small Grants Application Round 2014	944,500

Dashboard Filters

SG-742-Farmville-Council

Round Name

● Slice/Filter

○ Highlight

Add Filter | Edit Filter

○ Include all (no filter applied)

Allow multiselect for lists

● Find in the list

Select All Clear All

□ 0

☑ 1

○

○ Select from list

○ Custom

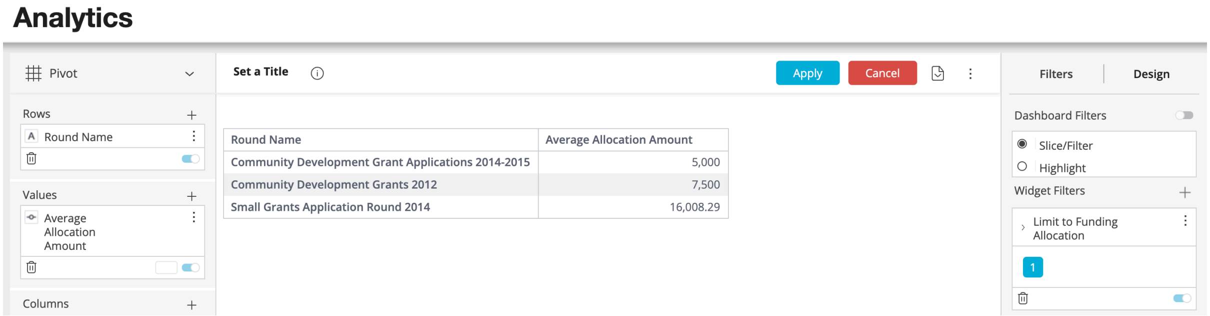
1 selected

Apply Cancel

6 Designing a Widget: Worked Examples

"I would like to find, for each round, the average funding allocation amount given to an application."

Step	**Reasoning**
Step 1: Find the tables containing the fields you will use	This widget would average the field "Allocation Amount" in Funding Allocation table. It would then split this data by "Round Name" in the Round table.
Step 2: Check it is possible to combine these tables in a single widget **Funding Allocation (Limit)** application_id budget_period_id funding_allocation_id instance_id program_id round_id Limit to Funding Allocation	This widget would combine multiple entity tables. Fields from two different entity tables can be used together in a widget if there is a limit table whose key combination contains the keys of both entity tables. The key of the Funding Allocation table is Funding Allocation ID. The key of the Round table is Round ID. There is a limit table whose key combination contains both these ids, such as Funding Allocation (Limit). This means Funding Allocation and Round tables can be used in a single widget.
Step 3: Select a Limit To filter	This widget is averaging "Allocation Amount," a field that gives information about funding allocations. Apply "Limit to Funding Allocation" from the Funding Allocation (Limit) table as a widget filter.



Analytics

Rows: Round Name

Values: Average Allocation Amount

Columns

Round Name	Average Allocation Amount
Community Development Grant Applications 2014-2015	5,000
Community Development Grants 2012	7,500
Small Grants Application Round 2014	16,008.29

Dashboard Filters

● Slice/Filter

○ Highlight

Widget Filters

Limit to Funding Allocation

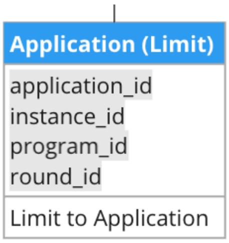
1

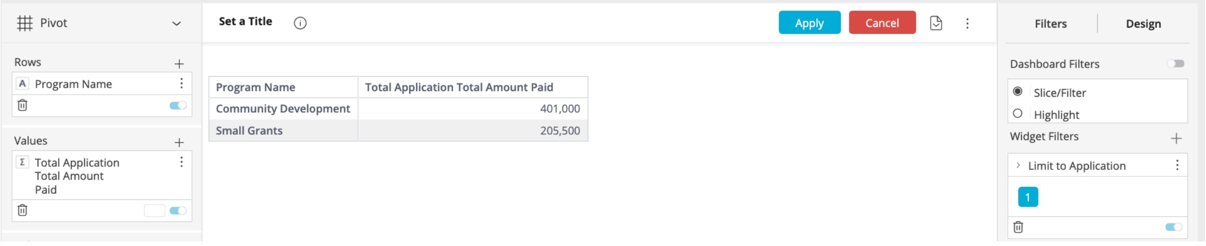
"I would like to sum the total amount paid to applications from each program."

Data about individual payments is contained in the Payments table, while the Application Financials table records the sum of payments made to each application in the field "Application Total Amount Paid." Both tables could be used to produce the same analysis.

Using Application Financials table:

Step	**Reasoning**
Step 1: Find the tables containing the fields you will use	The widget will sum "Application Total Amount Paid" from the Application Financials table. The widget

	would split (group) that data by “Program Name” from the Program table.
Step 2: Check it is possible to combine these tables in a single widget  Application (Limit) application_id instance_id program_id round_id Limit to Application	This widget would combine multiple entity tables. Fields from two different entity tables can be used together in a widget if there is a limit table whose key combination contains the keys of both entity tables. The key of the Application Financials table is Application ID. The key of Program table is Program ID. There is a limit table whose key combination contains both these ids, such as Application (Limit). This means Program and Application tables can be used in a single widget.
Step 3: Select a Limit To filter	This widget is aggregating “Application Total Amount Paid,” a field that gives information about applications. Apply “Limit to Application” from the Application (Limit) table as a widget filter.



Pivot

Set a Title

Apply

Cancel

Filters

Design

Rows

Program Name

Values

Total Application Total Amount Paid

Program Name	Total Application Total Amount Paid
Community Development	401,000
Small Grants	205,500

Dashboard Filters

● Slice/Filter

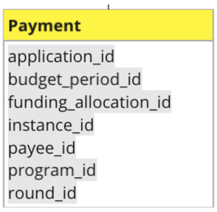
○ Highlight

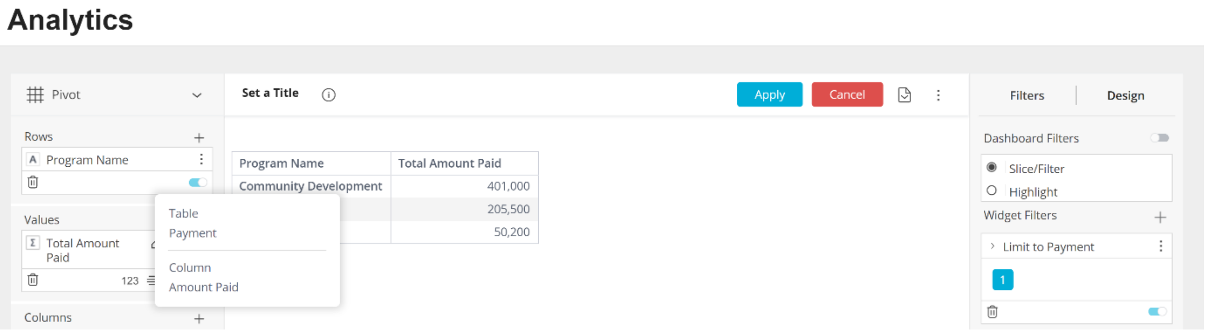
Widget Filters

Limit to Application

1

Using Payment table:

Step	**Reasoning**
Step 1: Find the tables containing the fields you will use	The widget will sum “Amount Paid” from the Payment table. The widget would split (group) that data by “Program Name” from the Program table.
Step 2: Check it is possible to combine these tables in a single widget  Payment application_id budget_period_id funding_allocation_id instance_id payee_id program_id round_id	This widget would group a field in a limit table by a field in an entity table. Fields in limit tables can be grouped by fields in any entity table whose key is contained in the limit table’s key combination. The key of Program table is Program ID. The Payment table is a limit table whose key combination uses Program ID. This means Payment and Application tables can be used in a single widget.
Step 3: Select a Limit To filter	This widget is aggregating “Amount Paid,” a field that gives information about payments. Apply “Limit to Payment” from the Payment table as a widget filter.



Rows: Program Name

Values: Total Amount Paid (Table: Payment; Column: Amount Paid)

Program Name	Total Amount Paid
Community Development	401,000
	205,500
	50,200

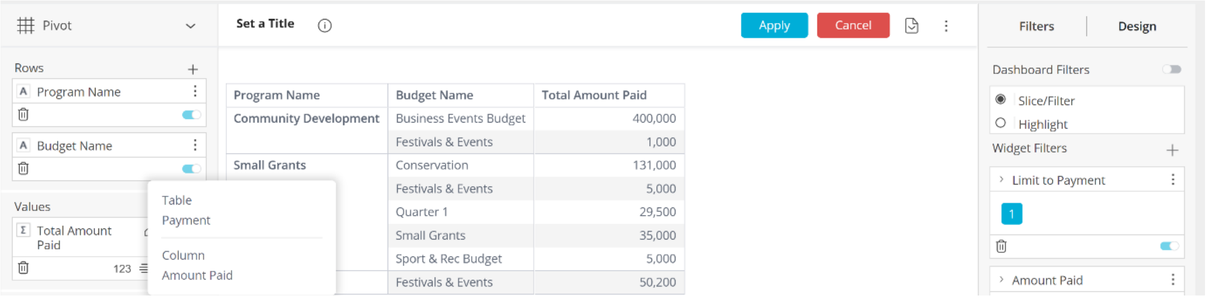
● Slice/Filter
○ Highlight

Limit to Payment: 1

“I would like to sum the total amount paid to applications from each program, broken down further by the budgets the funding came from.”

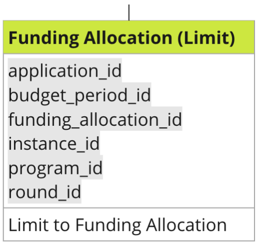
The previous example with a budget-related reporting requirement.

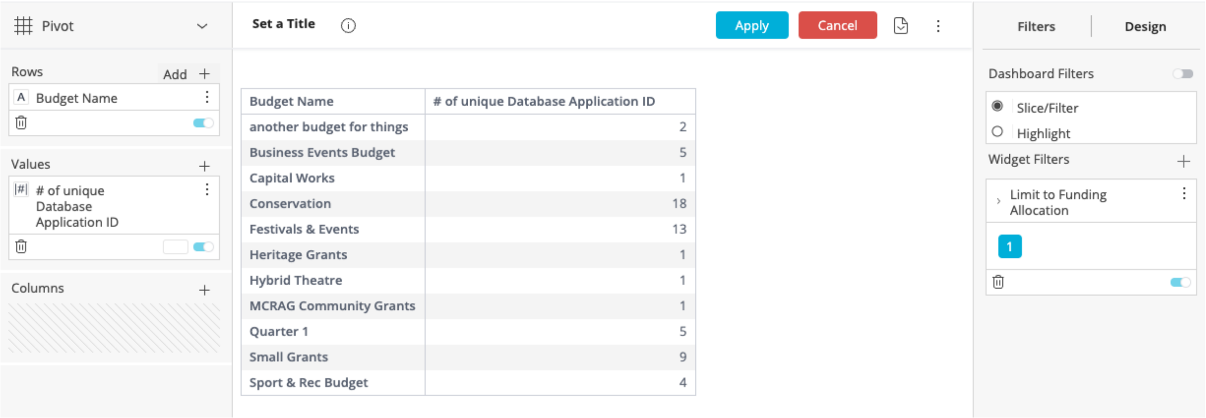
Step	**Reasoning**
Step 1: Find the tables containing the fields you will use	The widget would split (group) payments by “Program Name” from the Program table AND “Budget Name” from Budget table. Data about individual payments is contained in the Payments table, while the Application Financials table records the sum of payments made to each application in the field “Application Total Amount Paid.” However, an application may receive payments from multiple budgets. This means that “Application Total Amount Paid” cannot be broken down by budget and attempting to group this field by budget may lead to inflated numbers. The Application Financials table is not suitable for this analysis, which is reflected by the data model not allowing a user to use the Budget table with a Limit to Application filter (see next step.)
Step 2: Check it is possible to combine these tables in a single widget & Step 3: Select a “Limit to” filter	As shown in the previous example, it is possible to combine “Application Total Amount Paid” from the Application Financials Table and “Name” from the Program table in a widget, with “Limit to Application” set as a widget filter. However, Limit to Application can not be combined with fields from the Budget table, because the key of Budget table (Budget Period ID) is not contained in the key combination of Application (Limit). Budget Period ID *is* contained in the key combination of Payment (meaning “Amount Paid” from Payment table can be grouped by “Name” in the Budget Table.) Payment can be combined with Program table also. For this reporting example then, it makes sense to aggregate “Amount Paid” from the Payments table and apply “Limit to Payment” as the widget filter.



Program Name	Budget Name	Total Amount Paid
Community Development	Business Events Budget	400,000
	Festivals & Events	1,000
Small Grants	Conservation	131,000
	Festivals & Events	5,000
	Quarter 1	29,500
	Small Grants	35,000
	Sport & Rec Budget	5,000
	Festivals & Events	50,200

“I would like a list of applications that have received a funding allocation from each budget.”

Step	Reasoning
Step 1: Find the tables containing the fields you will use	This widget will provide a list of “Database Application IDs”, from the Application table. This list will be split by “Budget Name” from the Budget table.
Step 2: Check it is possible to combine these tables in a single widget  Funding Allocation (Limit) application_id budget_period_id funding_allocation_id instance_id program_id round_id Limit to Funding Allocation	This widget would combine multiple entity tables. Fields from two different entity tables can be used together in a widget if there is a limit table whose key combination contains the keys of both entity tables. They key of the Application table is Application ID. The key of the Budget table is Budget Period ID. There is a limit table whose key combination contains both these ids, such as the key combination of Funding Allocation (Limit). This means Application and Budget tables can be used in a single widget.
Step 3: Select a Limit To filter	Even though “Database Application ID” comes from Application table, creating a list of applications *that have received a funding allocation* technically involves listing the Application IDs associated with funding allocations. Select Limit to Funding Allocation from Funding Allocation (Limit) as the widget filter. Note that selecting “Limit to Payment” from the Payment table as the widget filter would return applications *that have received a payment.*



Rows: Budget Name

Values: # of unique Database Application ID

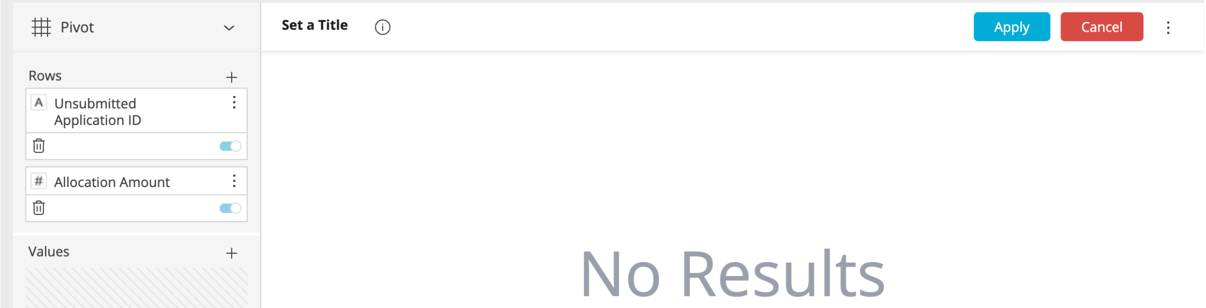
Widget Filters: Limit to Funding Allocation: 1

Budget Name	# of unique Database Application ID
another budget for things	2
Business Events Budget	5
Capital Works	1
Conservation	18
Festivals & Events	13
Heritage Grants	1
Hybrid Theatre	1
MCRAG Community Grants	1
Quarter 1	5
Small Grants	9
Sport & Rec Budget	4

7 Widget Design Troubleshooting/FAQs

If you have a question about designing a widget that is not contained in this list of FAQs, it is recommended to read the ensuing sections of this document which explain the data model in close detail.

Why does a widget show no results when I add a field? I know that data has been collected for that field.



Rows: Unsubmitted Application ID; Allocation Amount

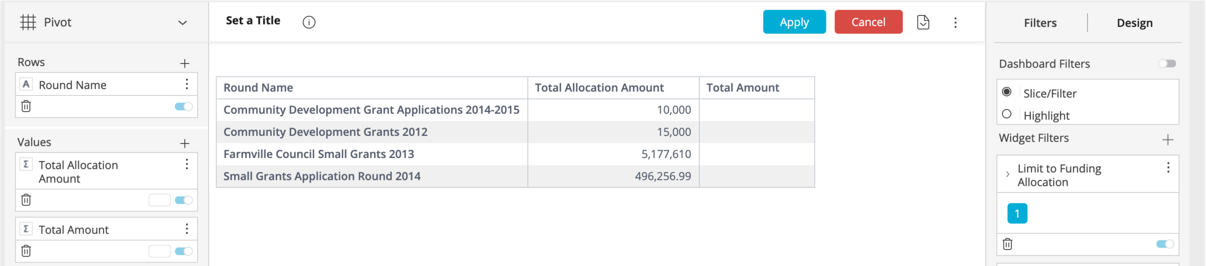
No Results

If a widget shows No Results, it is likely the combination of fields in the widget violates [the rules for combining tables in a widget](). In most cases, this result indicates a combination of fields that does not make logical sense.

For example, the widget in the image above combines “Unsubmitted Application ID” from the Unsubmitted Application table and “Allocation Amount” from the Funding Allocation table. Since unsubmitted applications should never have funding allocations (as they have not been approved for funding), it does not make logical sense to combine these fields.

That these fields cannot be logically combined is reflected in the rules for combining tables in a widget, whereby an entity table’s key must be contained in the key combination of any limit table it is combined with. Funding Allocation ID is not contained in the key combination of Unsubmitted Application.

Why are some (but not all) columns always coming up as blank or N/A in a widget? I know that data has been collected for that field.



Round Name	Total Allocation Amount	Total Amount
Community Development Grant Applications 2014-2015	10,000	
Community Development Grants 2012	15,000	
Farmville Council Small Grants 2013	5,177,610	
Small Grants Application Round 2014	496,256.99	

If some (but not all columns) come up as blank in a widget, those fields likely come from tables whose key/ key combination are not contained in the key combination of the Limit table used for the widget filter (see the note in the section on selecting "Limit to" filters). This usually occurs when the widget is aggregating a field describing an entity and you drag in a field describing another entity that the first entity is *not* nested under.

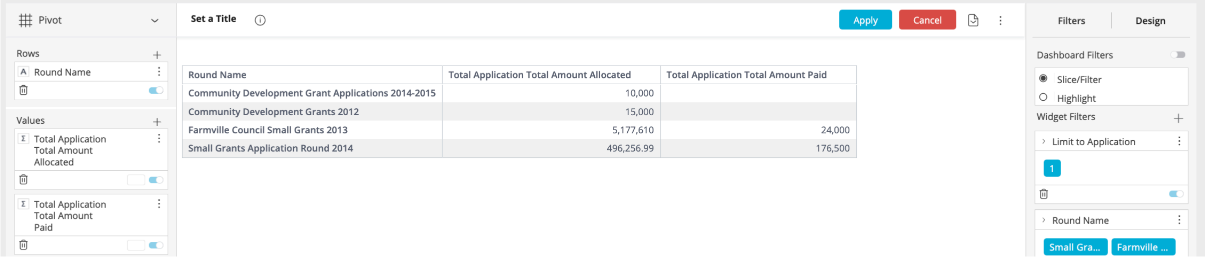
In the example above, the widget is summing "Allocation Amount" from the Funding Allocation table, meaning that "Limit to Funding Allocation" from the Funding Allocation (Limit) is applied as a widget filter. When "Amount" from the Payment table is added to the widget, all rows appear blank. This is because funding allocations do not come from payments, but the other way around. Payee ID (which appears in the key combination of the Payment table) does not appear in the key combination of Funding Allocation (Limit), which means fields from Payment table cannot be used with the Limit to Funding Allocation widget filter.

What should I do if the data model does not allow me to use fields together in a widget?

As explained in the first FAQ, if the fields you are trying to combine in a widget violate the rules for combining tables, this may indicate you have chosen a combination of fields that does not make logical sense. Otherwise, there may be a way to produce the analysis you have in mind through a combination of fields from other tables.

As explained further in the section on financial reporting, financial data is pre-aggregated at different levels in the Application, Funding Allocation, Payment and Budget tables. For example, individual payments are recorded in the Payment table. Payments are grouped by application in the Application Financials table and grouped by budget in the Budget table. This pre-aggregation maximises flexibility for the kind of financial reporting that is possible in SmartyGrants Analytics.

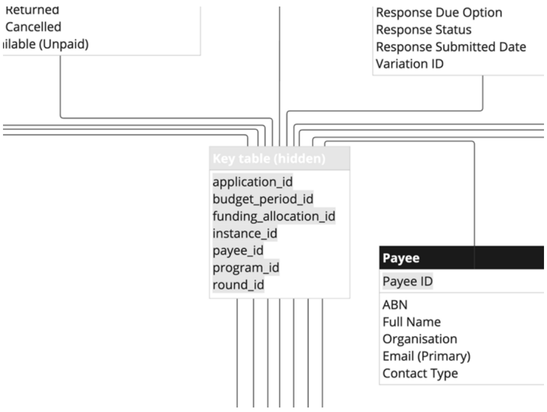
For example, the previous FAQ showed it was not possible to show the total amount paid and allocated to applications in each round by using the Payment and Funding Allocation tables together. However, it is possible to produce this analysis by using fields from the Application Financials table.



Round Name	Total Application Total Amount Allocated	Total Application Total Amount Paid
Community Development Grant Applications 2014-2015	10,000	
Community Development Grants 2012	15,000	
Farmville Council Small Grants 2013	5,177,610	24,000
Small Grants Application Round 2014	496,256.99	176,500

See here for another example of troubleshooting by using a different combination of tables.

8 The Key Table



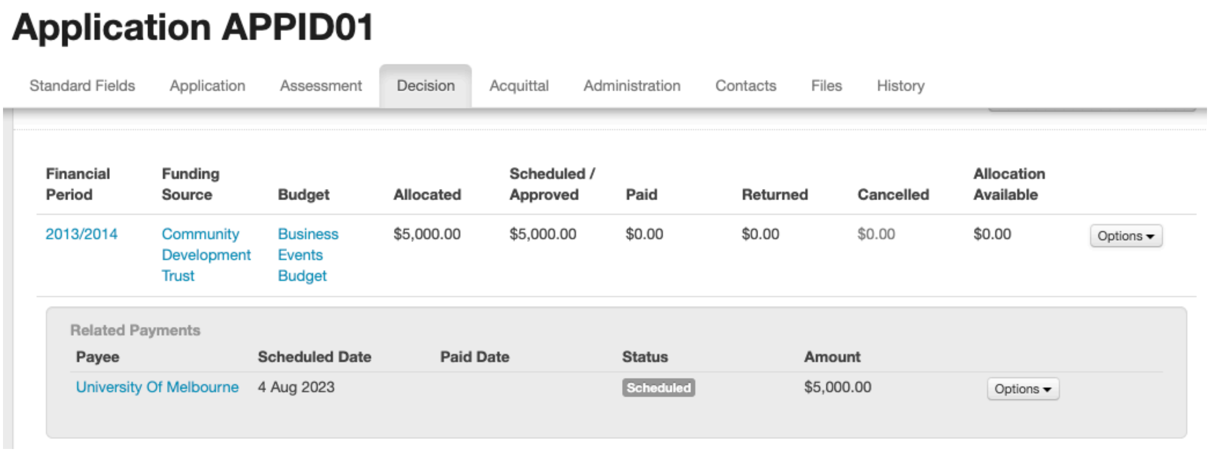
Why does the data model need the key table?

The key table, hidden from the end user, facilitates security filtering so that users only see data from instances they have access to. Every table in the data model is connected to the key table. Security filtering limits the Instance table to only relevant Instance IDs and flows through to every other table via the key table.

What is in the key table?

The key table contains seven columns, which are the seven keys used across all tables: Application ID, Round ID, Program ID, Instance ID, Budget Period ID, Funding Allocation ID and Payee ID.

As rows, the key table contains all actual permutations of every key combination (with irrelevant keys in a key combination always set to -1.) Consider the following data from SmartyGrants.



Application APPID01

Financial Period	Funding Source	Budget	Allocated	Scheduled / Approved	Paid	Returned	Cancelled	Allocation Available	
2013/2014	Community Development Trust	Business Events Budget	$5,000.00	$5,000.00	$0.00	$0.00	$0.00	$0.00	Options ▾

Related Payments

Payee	Scheduled Date	Paid Date	Status	Amount	
University Of Melbourne	4 Aug 2023		Scheduled	$5,000.00	Options ▾

The key combination for application is Application ID, Round ID, Program ID and Instance ID (key combinations are recorded in the data model diagram.) The application would appear in the key table with these IDs populated and all other IDs left as “-1”:

Application ID	*Round ID*	*Program ID*	*Instance ID*	*Budget ID*	*Funding Allocation ID*	*Payee ID*
APPID01	ROUID01	PROID01	INSID01	-1	-1	-1

APPID01 has one funding allocation. They key combination for funding allocation is Application ID, Round ID, Program ID, Instance ID, Budget Period ID and Funding

Allocation ID. The funding allocation would appear in the key table with these IDs populated and all other IDs left as “-1”:

Application ID	*Round ID*	*Program ID*	*Instance ID*	*Budget ID*	*Funding Allocation ID*	*Payee ID*
APPID01	ROUID01	PROID01	INSID01	-1	-1	-1
APPID01	ROUID01	PROID01	INSID01	BUDID01	FUNDID01	-1

APPID01 has also received one payment. Since the payment key combination uses all seven IDs, the payment would appear in the key table with all IDs populated:

Application ID	*Round ID*	*Program ID*	*Instance ID*	*Budget ID*	*Funding Allocation ID*	*Payee ID*
APPID01	ROUID01	PROID01	INSID01	-1	-1	-1
APPID01	ROUID01	PROID01	INSID01	BUDID01	FUNID01	-1
APPID01	ROUID01	PROID01	INSID01	BUDID01	FUNID01	PAYID01

Every “real” instance of an entity has a row corresponding to it in the key table. Storing key combinations in the key table allows us to group and slice entities by fields, which are contained in other tables joined to the key table.

9 Entity Tables

Entity tables contain fields related to an object or concept in SmartyGrants. For example, the Application table contains the fields “Current Stage” and “Decision Date”, with this data organised in rows by the key, Application ID.

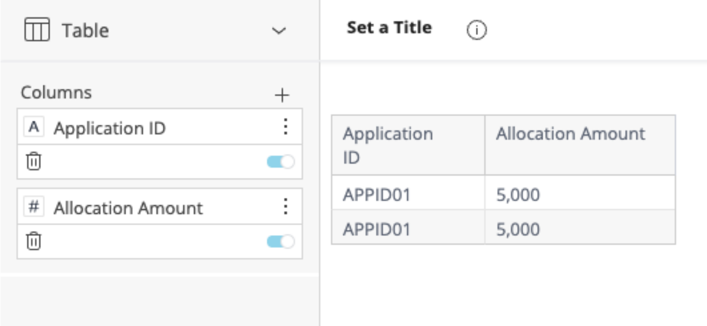
Application ID	Current Stage	Decision Date
APPID01	Evaluation	3/3/2024

The Funding Allocation table contains the fields “Allocation Amount” and “Conditional Flag”, with this data organised in rows by the key, Funding Allocation ID.

Funding Allocation ID	Allocation Amount	Conditional Flag
FUNDID01	$5000	Yes

Entity tables are connected to the key table by their key, allowing us to connect and group data across entities. Recall in our example that the key table contains all key combination permutations of applications and funding allocations (for now, disregard the row in the key table representing the payment):

Application ID	*Round ID*	*Program ID*	*Instance ID*	*Budget ID*	*Funding Allocation ID*	*Payee ID*
APPID01	ROUID01	PROID01	INSID01	-1	-1	-1
APPID01	ROUID01	PROID01	INSID01	BUDID01	FUNID01	-1



Application ID	Allocation Amount
APPID01	5,000
APPID01	5,000

Imagine that we select the fields “Application ID” and “Allocation Amount” in the widget builder to find the funding allocation amounts associated with each application.

In the background, the widget builder performs the following steps:

Step	Result
(1) Selection of a field from the Application table prompts widget builder to creates a list of Application IDs from that table	APPID01
(2) Creates a join to the key table on Application ID, and then finds Funding Allocation IDs associated with each Application ID.	APPID01 is joined to FUNDID01 because there is a row in the key table containing these two IDs.
(3) Creates a join to the Funding Allocation table on the Funding Allocation IDs.	In the Funding Allocation table, FUNID01 has Allocation amount of $5000. The widget builder connects the funding amount of $5000 to APPID01.

10 Limit Tables

What is in a limit table?

Like the key table, all limit tables contain the seven keys as columns, plus an additional field “Limit to <entity>”. For rows, each limit table contains all key combination permutations for the entity in its name (the key table is essentially a union of the seven ID columns across all limit tables.) The “Limit to <entity>” column is set to 1 for every row.

In our example, the Limit to Funding Allocation table would have the key combination permutations of the funding allocations in the key table, with an additional column set to “1”:

Application ID	*Round ID*	*Program ID*	*Instance ID*	*Budget ID*	*Funding Allocation ID*	*Payee ID*	*Limit to Funding Allocation*
APPID01	ROUID01	PROID01	INSID01	BUDID01	FUNID01	-1	1

Whereas entity tables are linked to the key table by a single key, limit tables are linked to the key table by all seven ID columns.

Why do some limit tables contain extra fields?

Some tables contain additional fields on top of the "Limit to" field. These tables refer to entities which cannot logically group any other entity because no entity is nested under it (as in the case of Payment, which comes from a Funding Allocation, which comes from a Budget. Nothing "comes from" a Payment.) Fields from these tables can be used in widgets with fields from the same table or can be grouped by fields from certain entity tables.

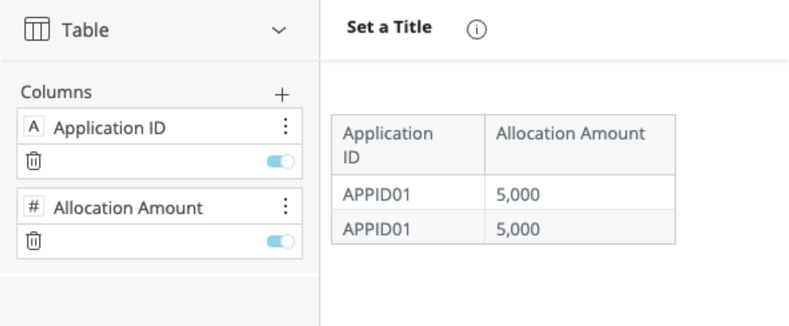
Why do we need Limit Tables?

Limit Tables allow the user to ensure widgets form the correct links between entities. Recall that:

- The key table contains all actual permutations of every key combination.
- There are overlapping keys across key combinations. For example, Application ID is used both in the key combination of Payment and the key combination of Funding Allocation.

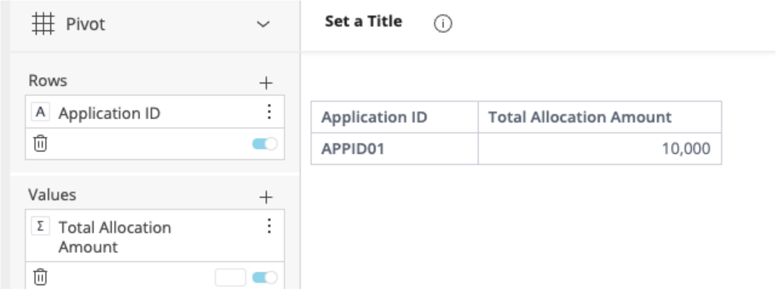
When a widget uses a table whose key is used in the key combinations of multiple entities, the widget query can form unexpected links between entities, leading to duplication.

Let's return to the example above where we selected the fields "Application ID" and "Allocation Amount" in the widget builder to find the funding allocation amounts associated with each application. You may have noticed that there are *two* rows associated with APPID01, each with allocation amount $5000.



Application ID	Allocation Amount
APPID01	5,000
APPID01	5,000

If we convert the widget type to a pivot table, this aggregates to $10,000. APPID01 has only one funding allocation, worth $5000, so some duplication is occurring.



Application ID	Total Allocation Amount
APPID01	10,000

The duplication occurs because the key table contains the key combinations of both payments and funding allocations. They key combinations of both these entities use Application ID and Funding Allocation ID. When the widget query makes joins to the key table on these keys, the join captures rows associated with both payment and funding allocations.

Application ID	*Round ID*	*Program ID*	*Instance ID*	*Budget ID*	*Funding Allocation ID*	*Payee ID*
APPID01	ROUID01	PROID01	INSID01	-1	-1	-1
APPID01	ROUID01	PROID01	INSID01	BUDID01	FUNID01	-1
APPID01	ROUID01	PROID01	INSID01	BUDID01	FUNID01	PAYID01

This duplication is documented in further detail at each step of the widget query:

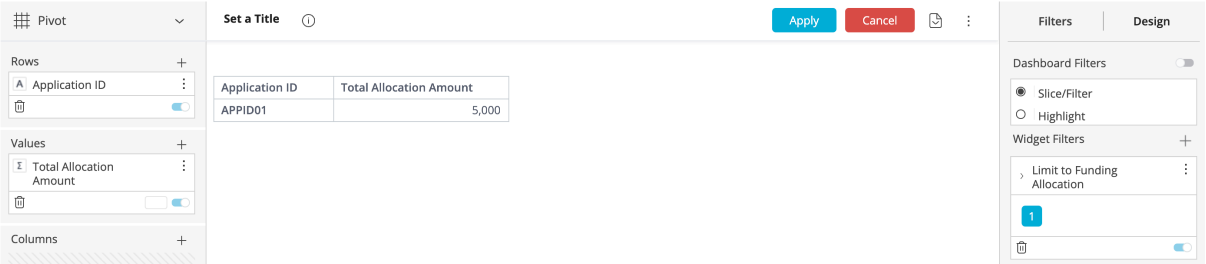
Step	Result
(1) Selection of a field from the Application table prompts widget builder to creates a list of Application IDs from that table	APPID01
(2) Creates a join to the key table on Application ID, and then finds Funding Allocation IDs associated with each Application ID.	APPID01 is joined to FUNDID01 in two rows because there are two rows with APPID01 and FUNID01. One of these rows is associated with a funding allocation. The other is associated with a payment.
(3) Creates a join to the Funding Allocation table on the Funding Allocation IDs.	In the Funding Allocation table, FUNDID01 has Allocation amount of $5000. The widget builder connects the funding amount of $5000 to APPID01 in both rows containing APPID01 and FUNDID01.

By making use of the limit tables, a user can ensure the widget query forms the correct link between entities, eliminating duplication.

How do Limit To filters work?

Limit Tables work by reducing the key table to the key combination permutations of a single entity. To use a limit table, consider which entity is being described by the field your widget is aggregating. In most cases the entity can be identified by the name of the table from which you have selected a field to count/sum/average etc. Then apply the "Limit to" field from the limit table of that entity as a widget filter, set to include only the value "1".

The "Limit to" filter affects the widget query by creating an additional join between the limit table and the key table that only keeps key combination permutations matching those from the limit table used in the filter (reducing the key combination permutations used in the widget query to those of a single entity.) This overcomes the problem of widget queries forming unexpected links between entities.



Rows: Application ID

Values: Total Allocation Amount

Columns

Application ID	Total Allocation Amount
APPID01	5,000

Dashboard Filters

● Slice/Filter
○ Highlight

Widget Filters

Limit to Funding Allocation

1

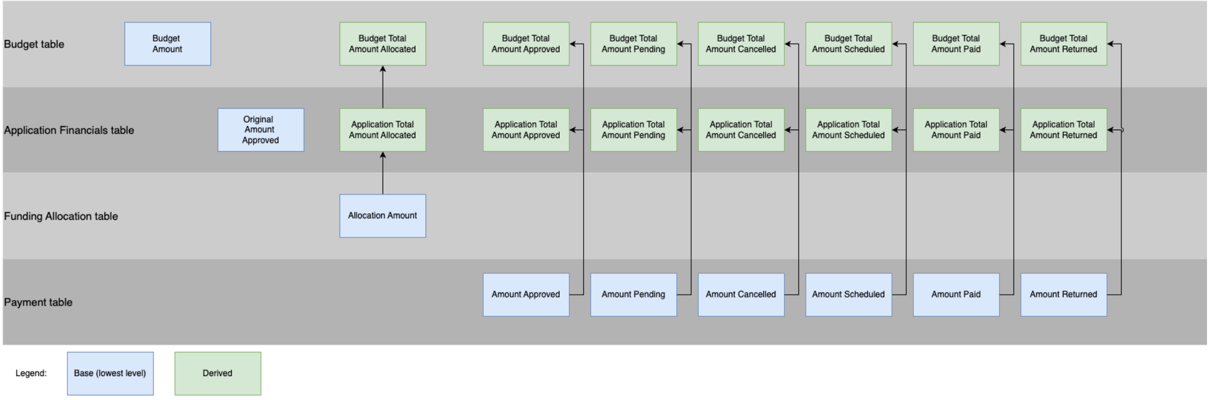
In our example, we are summing funding allocation amounts and so would apply the Limit to Funding Allocation field from the Funding Allocation (Limit) table as a widget filter. The deduplication effect can be stepped out:

Step	Example
(1) Selection of a field from the Application table prompts widget builder to creates a list of Application IDs from that table	APPID01
(2) Widget filter of "Limit to Funding Allocation = 1" creates a join from the Funding Allocation (Limit) table to the key table on all seven ID columns	Every row in the Funding Allocation (Limit) table has Limit to Funding Allocation set to "1", and those rows also have Payee ID set to "-1." The join to the key table on the seven keys therefore only preserves rows with Payee ID set to "-1" (and matching the key combination permutations in Funding Allocation (Limit) table across the other six keys). The row in the key table associated with a payment is omitted from the join.
(3) Creates a join from Application table to the key table on Application ID, and then finds Funding Allocation IDs associated with each Application ID.	APPID01 is joined to FUNID01 in only one row with both APPID01 and FUNID01. This row is associated with a funding allocation.
(4) Creates a join to the Funding Allocation table on the Funding Allocation IDs.	In the Funding Allocation table, FUNID01 has Allocation amount of $5000. The widget builder connects the funding amount of $5000 to APPID01.

11 Financial Reporting

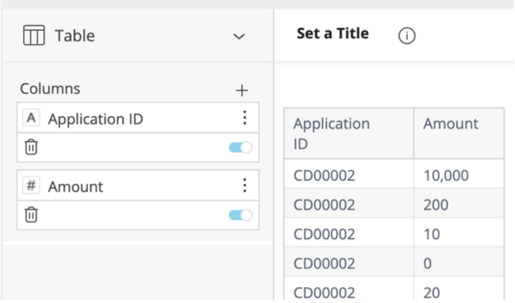
In the data model, financial data is pre-aggregated at different levels in the Application, Funding Allocation, Payment and Budget tables. For example, individual payments are recorded in the Payment table. Payments are grouped by application in the Application Financials table and grouped by budget in the Budget table.

The relationship between fields in these four tables is summarised in the image below, where "derived" means a field aggregates data collected at a different level. For example, "Application Total Amount Approved" sums all "Amount Approved" (approved payments) attached to an application. "Budget Total Amount Approved" sums all "Amount Approved" (approved payments) attached to a budget.


Budget table
Budget Amount
Budget Total Amount Allocated
Budget Total Amount Approved
Budget Total Amount Pending
Budget Total Amount Cancelled
Budget Total Amount Scheduled
Budget Total Amount Paid
Budget Total Amount Returned
Application Financials table
Original Amount Approved
Application Total Amount Allocated
Application Total Amount Approved
Application Total Amount Pending
Application Total Amount Cancelled
Application Total Amount Scheduled
Application Total Amount Paid
Application Total Amount Returned
Funding Allocation table
Allocation Amount
Payment table
Amount Approved
Amount Pending
Amount Cancelled
Amount Scheduled
Amount Paid
Amount Returned
Legend:
Base (lowest level)
Derived

This pre-aggregation maximises flexibility for the kind of financial reporting that is possible in SmartyGrants Analytics. To use these tables effectively, you should have a firm understanding of how funding is administered in SmartyGrants (or refer to the "Funding and Payments" page on the Help Hub).

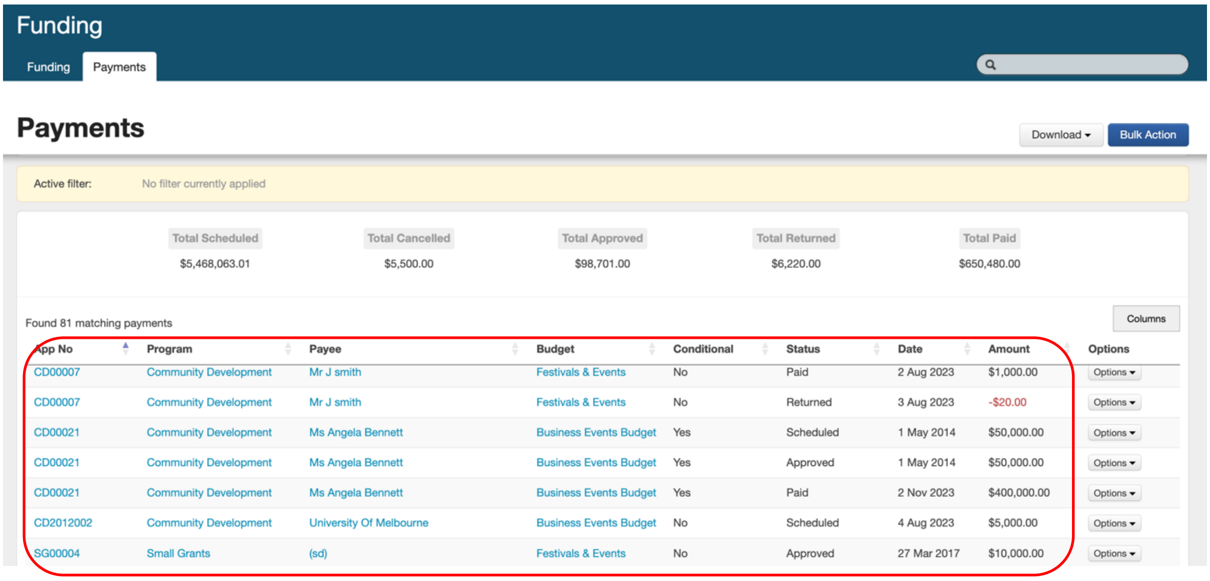
Payment table



Application ID	Amount
CD00002	10,000
CD00002	200
CD00002	10
CD00002	0
CD00002	20

The Payments table contains data around individual payments (i.e. one row per payment), similar to the Payments page in SmartyGrants. Applications can receive multiple payments, which are separated out into different rows in this table.

Note that the Payments table contains payments of all statuses (e.g. scheduled, approved, returned, paid.) When using this table, you may need to filter by "Status" or use the "amount" columns which have been pre-filtered (e.g. "Amount Scheduled", "Amount Approved" columns).



Payments

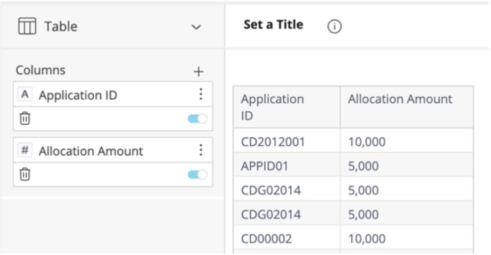
Active filter: No filter currently applied

Total Scheduled	Total Cancelled	Total Approved	Total Returned	Total Paid
$5,468,063.01	$5,500.00	$98,701.00	$6,220.00	$650,480.00

Found 81 matching payments

App No	Program	Payee	Budget	Conditional	Status	Date	Amount	Options
CD00007	Community Development	Mr J smith	Festivals & Events	No	Paid	2 Aug 2023	$1,000.00	Options ▾
CD00007	Community Development	Mr J smith	Festivals & Events	No	Returned	3 Aug 2023	-$20.00	Options ▾
CD00021	Community Development	Ms Angela Bennett	Business Events Budget	Yes	Scheduled	1 May 2014	$50,000.00	Options ▾
CD00021	Community Development	Ms Angela Bennett	Business Events Budget	Yes	Approved	1 May 2014	$50,000.00	Options ▾
CD00021	Community Development	Ms Angela Bennett	Business Events Budget	Yes	Paid	2 Nov 2023	$400,000.00	Options ▾
CD2012002	Community Development	University Of Melbourne	Business Events Budget	No	Scheduled	4 Aug 2023	$5,000.00	Options ▾
SG00004	Small Grants	(sd)	Festivals & Events	No	Approved	27 Mar 2017	$10,000.00	Options ▾

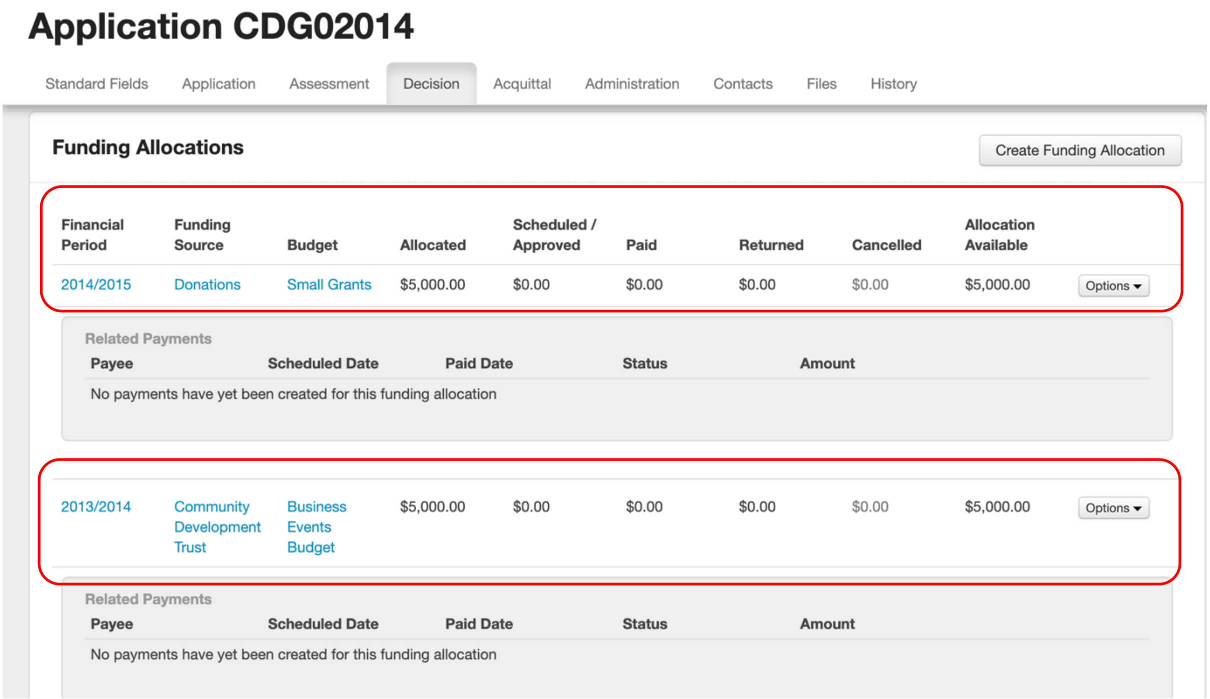
Funding Allocation table



Application ID	Allocation Amount
CD2012001	10,000
APPID01	5,000
CDG02014	5,000
CDG02014	5,000
CD00002	10,000

The Funding Allocation table contains data around funding allocations (i.e. one row per funding allocation). Applications can receive multiple funding allocation, which are separated out into different rows in this table.

Data in the Funding Allocation table can be matched to the "Decision Tab" of applications.



Financial Period	Funding Source	Budget	Allocated	Scheduled / Approved	Paid	Returned	Cancelled	Allocation Available
2014/2015	Donations	Small Grants	$5,000.00	$0.00	$0.00	$0.00	$0.00	$5,000.00
2013/2014	Community Development Trust	Business Events Budget	$5,000.00	$0.00	$0.00	$0.00	$0.00	$5,000.00

Related Payments

Payee	Scheduled Date	Paid Date	Status	Amount

No payments have yet been created for this funding allocation

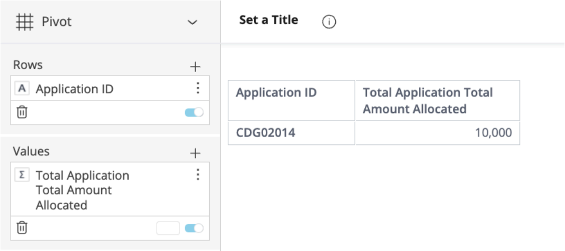
Related Payments

Payee	Scheduled Date	Paid Date	Status	Amount

No payments have yet been created for this funding allocation

Application Financials table

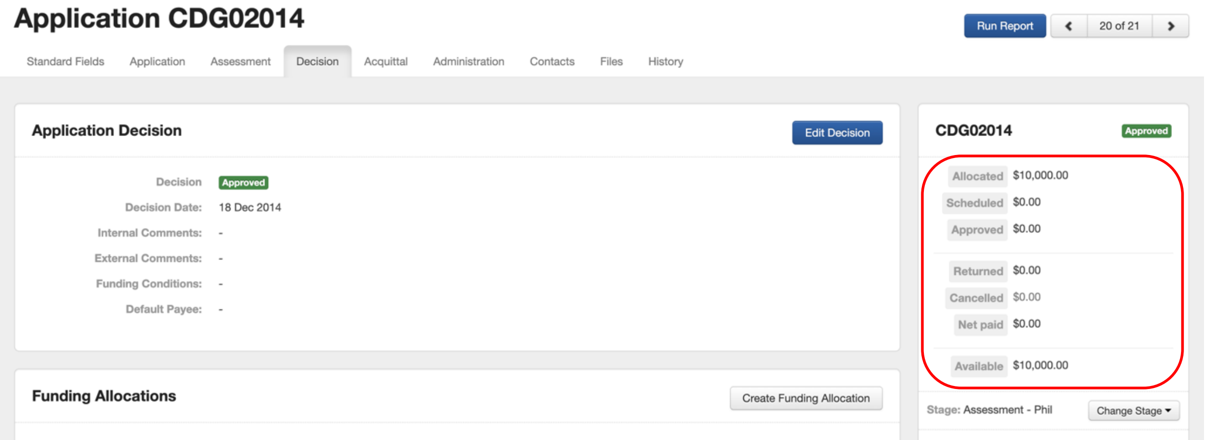
The Application Financials table records the sums of payments/funding allocations to each application (i.e. one row per application). It allows us to see the total payments and allocations attached to each application quickly.



Application ID	Total Application Total Amount Allocated
CDG02014	10,000

In the previous image, we saw that CDG02014 received two funding allocations of $5000 which were stored in two separate rows in the Funding Allocation table. In the Application Financials table, CDG02014 has "Application Total Amount Allocated" of $10,000.

Data in the Application Financials table can be matched to the right-hand summary column of applications in SmartyGrants.



Application CDG02014

Application Decision

Decision	Approved
Decision Date:	18 Dec 2014
Internal Comments:	-
External Comments:	-
Funding Conditions:	-
Default Payee:	-

Funding Allocations

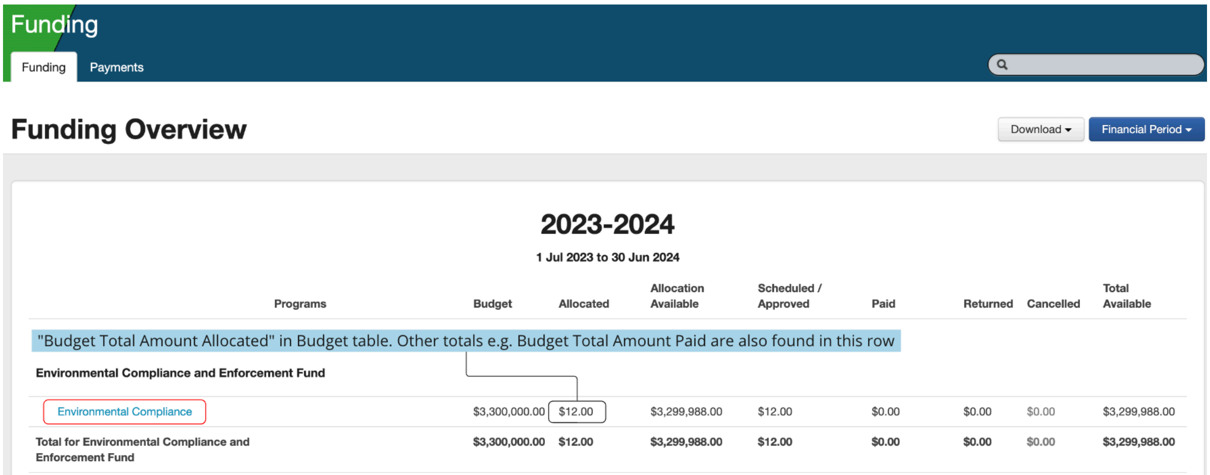
CDG02014 Approved

Allocated	$10,000.00
Scheduled	$0.00
Approved	$0.00
Returned	$0.00
Cancelled	$0.00
Net paid	$0.00
Available	$10,000.00

Stage: Assessment - Phil

Budget table

The Budget table contains data around budgets (i.e. one row per budget). It allows us to see the total payments and allocations which have come out of budgets quickly. Data in the Budget table can be matched to the Funding Overview page in SmartyGrants.



Funding

Funding Overview

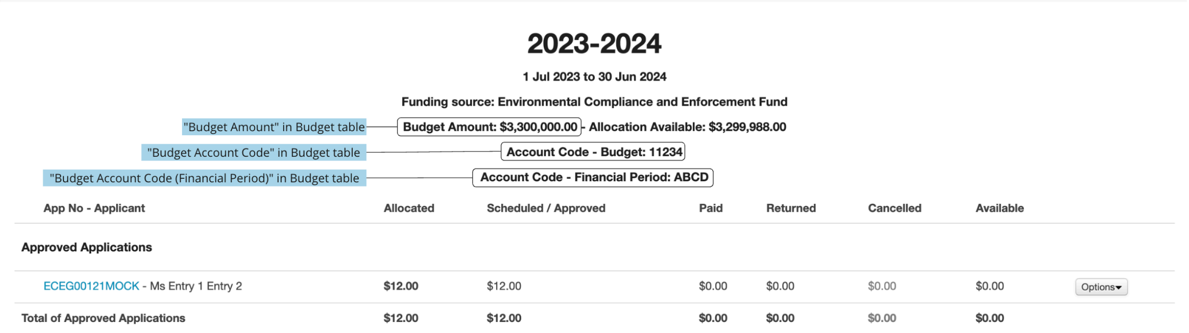
2023-2024

1 Jul 2023 to 30 Jun 2024

"Budget Total Amount Allocated" in Budget table. Other totals e.g. Budget Total Amount Paid are also found in this row

Programs	Budget	Allocated	Allocation Available	Scheduled / Approved	Paid	Returned	Cancelled	Total Available
Environmental Compliance and Enforcement Fund								
Environmental Compliance	$3,300,000.00	$12.00	$3,299,988.00	$12.00	$0.00	$0.00	$0.00	$3,299,988.00
Total for Environmental Compliance and Enforcement Fund	**$3,300,000.00**	**$12.00**	**$3,299,988.00**	**$12.00**	**$0.00**	**$0.00**	**$0.00**	**$3,299,988.00**

Clicking on a budget name in the Funding Overview shows further information also stored in the Budget table.



2023-2024

1 Jul 2023 to 30 Jun 2024

Funding source: Environmental Compliance and Enforcement Fund

"Budget Amount" in Budget table — Budget Amount: $3,300,000.00 - Allocation Available: $3,299,988.00

"Budget Account Code" in Budget table — Account Code - Budget: 11234

"Budget Account Code (Financial Period)" in Budget table — Account Code - Financial Period: ABCD

App No - Applicant	Allocated	Scheduled / Approved	Paid	Returned	Cancelled	Available	
Approved Applications							
ECEG00121MOCK - Ms Entry 1 Entry 2	$12.00	$12.00	$0.00	$0.00	$0.00	$0.00	Options
Total of Approved Applications	$12.00	$12.00	$0.00	$0.00	$0.00	$0.00	

12 Financial Reporting: Worked Examples

Below are some worked examples demonstrating the selection of tables according to reporting requirements.

"I would like to find the total amount of funding paid out to each application."

- This widget would group funding by application. The Application Financials table is most appropriate here because it sums payments/funding allocations given to each application (i.e. one row per application). The field in the Application Financials table that sums payments is “Application Total Amount Paid.’
- It is also possible to build out the same widget using the Payment Table (because applications receive payments, payments can be grouped by application). See here.
- “Budget Total Amount Paid” in the Budget table is not appropriate for this widget, because this field sums all payments from a budget (and cannot be broken down by application)

“I would like a list of payments that have been scheduled or approved, but not yet paid out to applications yet.”

- This widget would list individual payments in a table (i.e. one row per payment). Only Payment table is appropriate for this, as payments are summed per application and per budget in the Application Financials and Budget tables and cannot be broken down individually.

“I would like to find the total amount of funding paid out from each budget in this program.”

- This widget would group payments by budget. However, it would also group payments by program, and budgets can run across multiple programs. Budget table would be unsuitable for this analysis, as the field “Budget Total Amount Paid” in this table would give the amount paid from a budget to applications in *all* programs (and cannot be broken down by program).
- Grouping a field from the Budget Table by “Program Name” in Program table does not make sense and when “Limit to Budget” is applied as a widget filter, data will appear blank. See here for an explanation if required.
- Sum “Amount Paid” from the Payment table instead, which can be grouped by both Program and Budget tables. Payments are paid from a budget to an application that can only belong to one program, so this sum would give an accurate calculation of the amount paid from a budget to a specific program.

13 Financial Reporting: Troubleshooting/FAQs

What is the relationship between “Amount Paid” in the Payment table, “Application Total Amount Paid” in the Application Financials table and “Budget Total Amount Paid” in the Budget Period Table?

- “Amount Paid” in the Payment table contains data about individual payments (one row per payment). The key combination of payments includes Application ID and Budget Period ID since payments are paid to an application and come from a budget.
- “Application Total Amount Paid” in Application Financials table groups payments by application. If in the Payments table there are two payments worth $500 attached to Application ID APP01, in Application Financials table APP01 will have “Application Total Amount Paid” of $1000.
- “Budget Total Amount Paid” in Budget table groups payments by budget. If in the Payments table there are two payments worth $500 attached to Budget Period ID of BUD01, in Budget table BUD01 will have “Budget Total Amount Paid” of $1000.

This aggregation of payments at different levels is for user convenience and maximises the kind of financial reporting that is possible in SmartyGrants Analytics.

What is the difference between "Allocation Amount" in the Funding Allocation table, "Application Total Amount Allocated" in the Application Financials table and "Budget Total Amount Allocated" in the Budget table?

- "Allocation Amount" in the Funding Allocation table contains data about individual funding allocations (one row per funding allocation). The key combination of funding allocations includes Application ID and Budget Period ID since funding allocations are given to an application and come from a budget.
- "Application Total Amount Allocated" in Application Financials table groups funding allocations by application. If in the Funding Allocation table there are two funding allocations worth $500 attached to Application ID APP01, in Application Financials table APP01 will have "Application Total Amount Allocated" of $1000.
- "Budget Total Amount Allocated" in Budget table groups funding allocations by budget. If in the Funding Allocation table there are two funding allocations worth $500 attached to Budget Period ID of BUD01, in Budget table BUD01 will have "Budget Total Amount Allocated" of $1000.

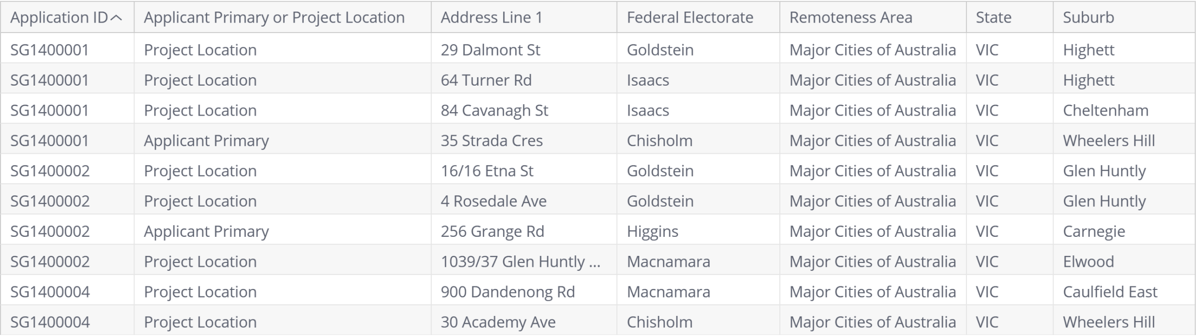
This aggregation of payments at different levels is for user convenience and maximises the kind of financial reporting that is possible in SmartyGrants Analytics.

Why are the numbers in my widget using Payment table not as expected?

The Payments table contains payments of all statuses (e.g. scheduled, approved, returned, paid.) When using this table, remember to filter by "Status" or use the "amount" columns which have been pre-filtered (e.g. "Amount Scheduled", "Amount Approved" columns).

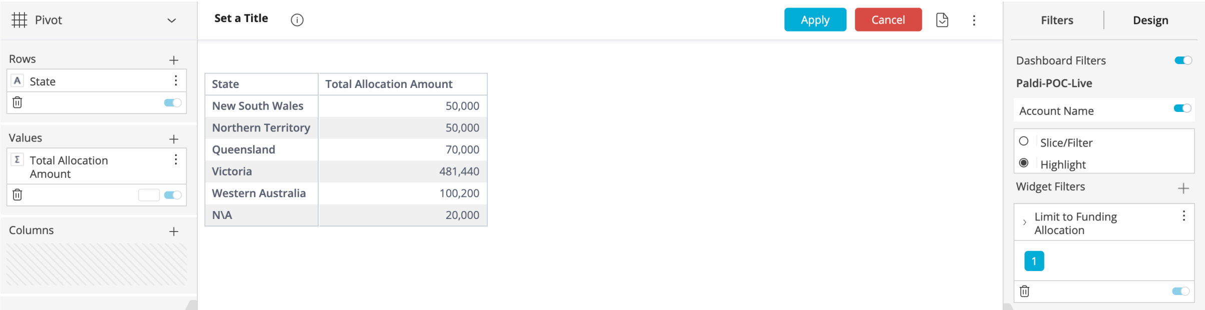
14 Location data and maps

The Location table stores Applicant Primary and Project Location addresses associated with applications. Addresses can fall in various boundaries (an address is in a certain state, suburb, state electorate etc.) Use fields in the Location table to find the boundaries associated with an application's addresses.



Application ID^	Applicant Primary or Project Location	Address Line 1	Federal Electorate	Remoteness Area	State	Suburb
SG1400001	Project Location	29 Dalmont St	Goldstein	Major Cities of Australia	VIC	Highett
SG1400001	Project Location	64 Turner Rd	Isaacs	Major Cities of Australia	VIC	Highett
SG1400001	Project Location	84 Cavanagh St	Isaacs	Major Cities of Australia	VIC	Cheltenham
SG1400001	Applicant Primary	35 Strada Cres	Chisholm	Major Cities of Australia	VIC	Wheelers Hill
SG1400002	Project Location	16/16 Etna St	Goldstein	Major Cities of Australia	VIC	Glen Huntly
SG1400002	Project Location	4 Rosedale Ave	Goldstein	Major Cities of Australia	VIC	Glen Huntly
SG1400002	Applicant Primary	256 Grange Rd	Higgins	Major Cities of Australia	VIC	Carnegie
SG1400002	Project Location	1039/37 Glen Huntly ...	Macnamara	Major Cities of Australia	VIC	Elwood
SG1400004	Project Location	900 Dandenong Rd	Macnamara	Major Cities of Australia	VIC	Caulfield East
SG1400004	Project Location	30 Academy Ave	Chisholm	Major Cities of Australia	VIC	Wheelers Hill

Boundary fields in the Location table can be used to group and aggregate data geographically. The pivot table below shows the funding allocated to applications whose Applicant Primary/ Project Location addresses fall in a particular state.



Rows: State

Values: Total Allocation Amount

Columns

State	Total Allocation Amount
New South Wales	50,000
Northern Territory	50,000
Queensland	70,000
Victoria	481,440
Western Australia	100,200
N\A	20,000

Dashboard Filters

Paldi-POC-Live

Account Name

○ Slice/Filter

● Highlight

Widget Filters

Limit to Funding Allocation

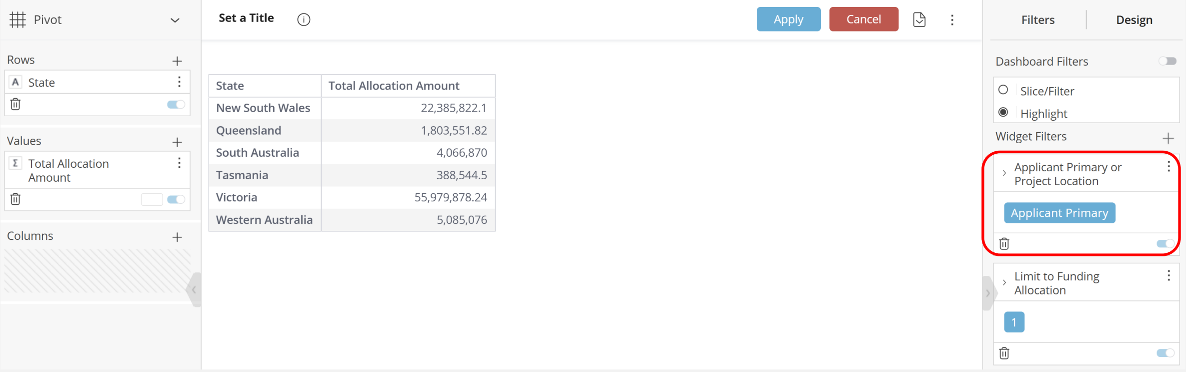
1

Warning: when using boundary fields to group data, apply “Applicant Primary or Project Location” from the Location table as a widget filter.

Each application has only one Applicant Primary address but may have several Project Location addresses. When using boundary fields from the Location table to group data, the association of an application with multiple addresses may lead to double counting. For example, if an application has both an Applicant Primary and a Project Location address in the state of Victoria, this may lead to the application’s funding being attributed to Victoria twice.

Since an application only ever has one Applicant Primary address, filtering for “Applicant Primary” removes any double counting.

- Compared to the pivot table above, the application of the widget filter removes double counting, leading to figures in the pivot table decreasing.



Rows: State

Values: Total Allocation Amount

Columns

State	Total Allocation Amount
New South Wales	22,385,822.1
Queensland	1,803,551.82
South Australia	4,066,870
Tasmania	388,544.5
Victoria	55,979,878.24
Western Australia	5,085,076

Dashboard Filters

○ Slice/Filter

● Highlight

Widget Filters

Applicant Primary or Project Location

Applicant Primary

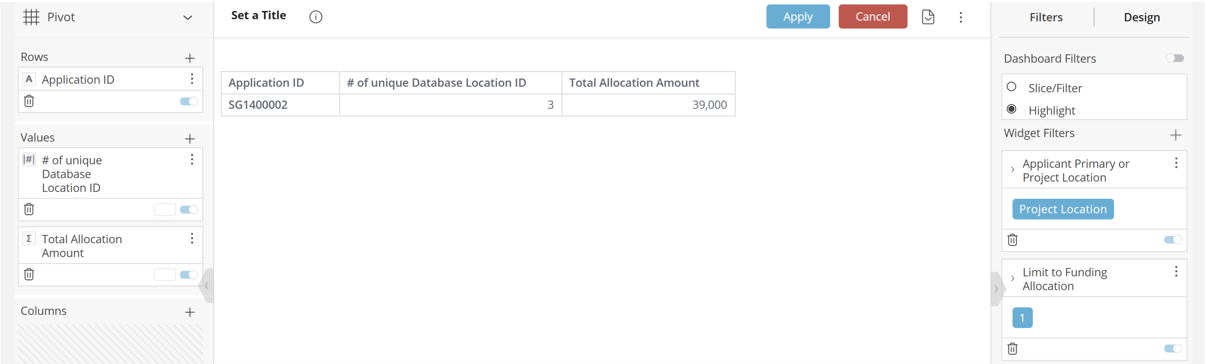
Limit to Funding Allocation

1

Warning: if the “Applicant Primary or Project Location” filter includes Project Location addresses, apply a custom formula.

You may apply “Applicant Primary or Project Location” to filter for “Project Location” addresses, in order to have an application’s funding attributed to the state, LGA, suburb etc. where its project locations are found. However, an application may have several project location addresses. If an application has multiple project location addresses in the same boundary (e.g. two project location addresses, both in the state of Victoria), this may lead to the application’s funding being double counted (e.g. attributed to Victoria twice.)

For example, the pivot table shows that the application has three Project Location addresses, and that the application has received funding allocations totalling $39,000.



Rows: Application ID

Values: # of unique Database Location ID; Total Allocation Amount

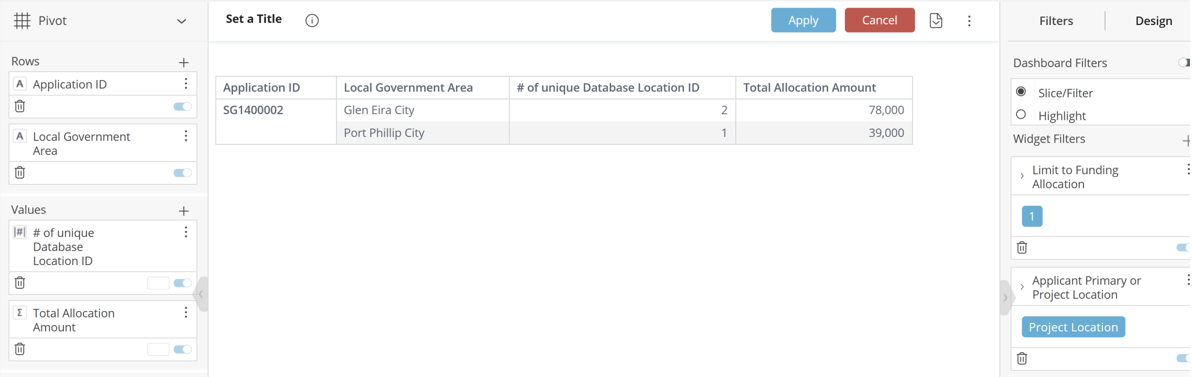
Columns

Applicant Primary or Project Location: Project Location

Limit to Funding Allocation: 1

Application ID	# of unique Database Location ID	Total Allocation Amount
SG1400002	3	39,000

If "Local Government Area" is added to the pivot table, two of the project location addresses are found in Glen Eira City. This leads to the application's funding allocations being counted towards this local government area twice, leading to the figure of $78,000.



Rows: Application ID; Local Government Area

Values: # of unique Database Location ID; Total Allocation Amount

Limit to Funding Allocation: 1

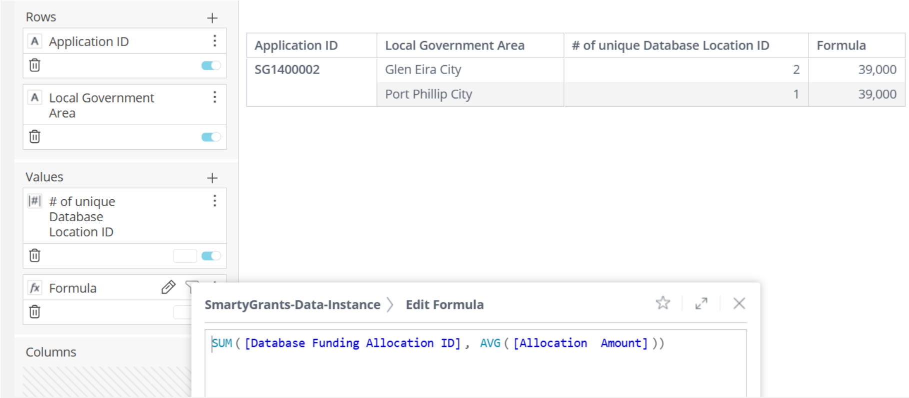
Applicant Primary or Project Location: Project Location

Application ID	Local Government Area	# of unique Database Location ID	Total Allocation Amount
SG1400002	Glen Eira City	2	78,000
	Port Phillip City	1	39,000

Depending on analysis requirements, this double counting may be desired. If not, you can use a custom formula in the **Values** box in the lefthand toolbar of the widget builder to ensure that an application is only counted towards a boundary once. To write a custom formula, add a field to the **Values** box that you would like to aggregate per boundary and click the pen icon in the box that appears.

- If summing fields from the Application Financials table, use the formula structure: SUM([Database Application ID], AVG([Application Total Amount Allocated]))
- If summing amounts from the Funding Allocation table, use: SUM([Database Funding Allocation ID], AVG([Amount Allocated]))
- If summing amounts from the Payment table, use: SUM([Database Payment ID], AVG([Amount]))

You may need to replace field names with those relevant to your analysis (e.g. replacing "Average Application Total Amount Allocated" with "Average Application Total Amount Paid".) If you would like to average data, replace SUM with AVG (or another aggregative function.)



Application ID	Local Government Area	# of unique Database Location ID	Formula
SG1400002	Glen Eira City	2	39,000
	Port Phillip City	1	39,000

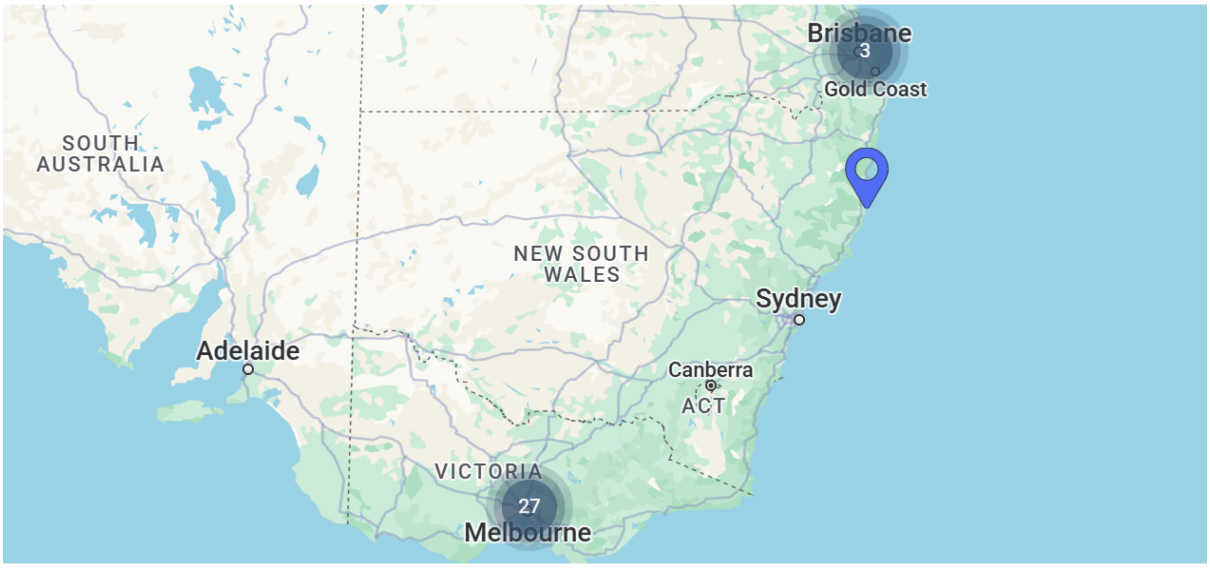
```
SUM ( [Database Funding Allocation ID] , AVG ( [Allocation  Amount] ))
```

Now the application's funding is now only counted once to each local government area that it has a project location in.

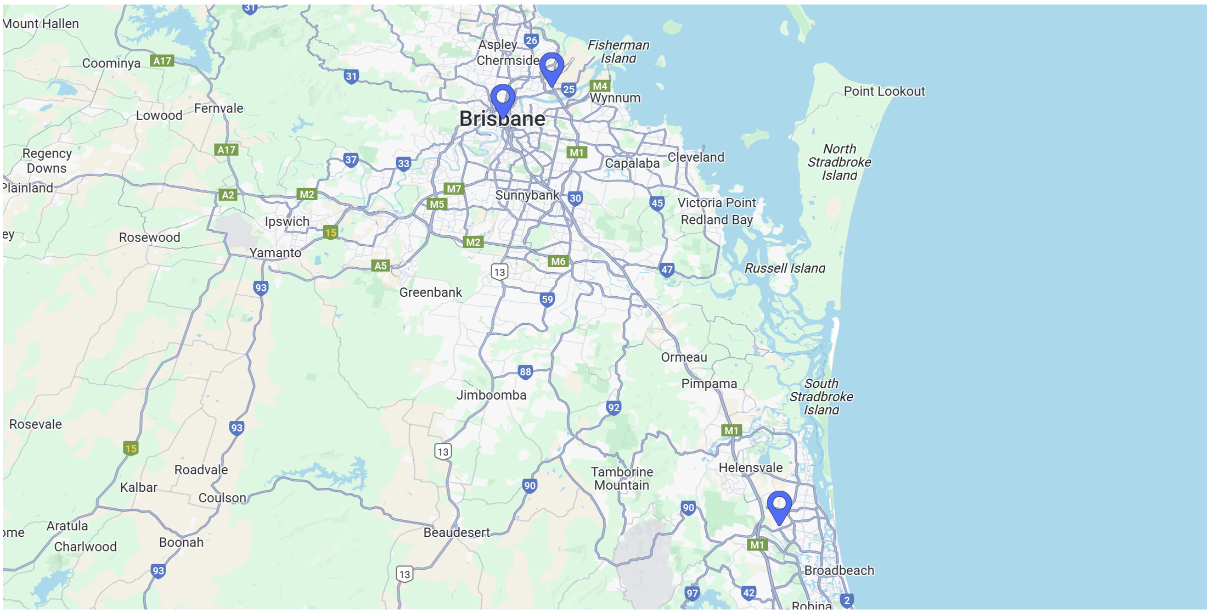
It is possible to write a custom formula to aggregate fields from tables besides Application Financials, Funding Allocation and Payment following the same structure of the provided formulas. The formula will always need an ID field that uniquely identifies every row in the table containing the field being aggregated (e.g. Database Application ID) and a field for aggregation (e.g. Application Total Amount Allocated.)

Building a scatter map

Scatter maps can be used to visualise the points where addresses are found on a map, as well as aggregations associated with each point. If there are multiple points located close together, they will be grouped together in a cluster. In the below map for example, there are three addresses near Brisbane clustered together and 27 addresses near Melbourne clustered together.

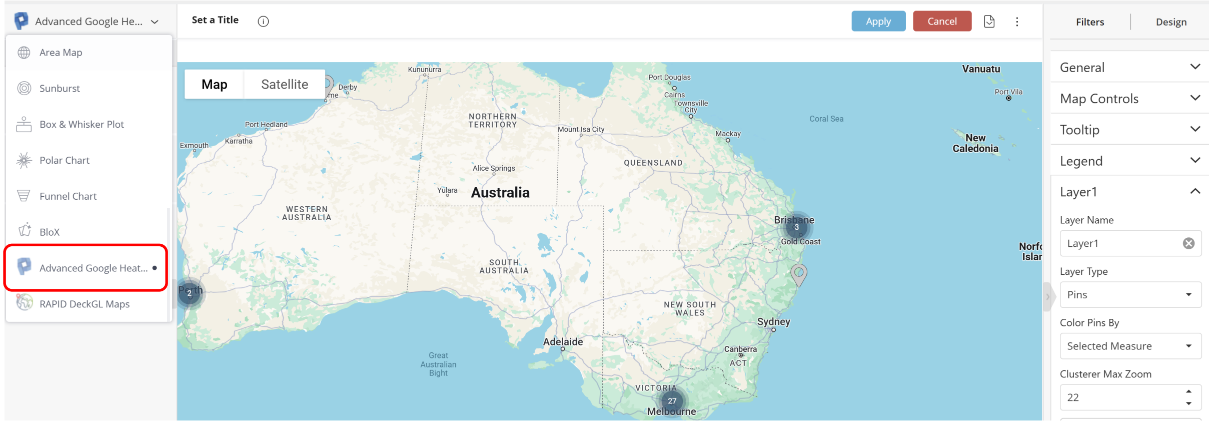


Zooming into Brisbane separates out the three individual addresses located in or around this city.



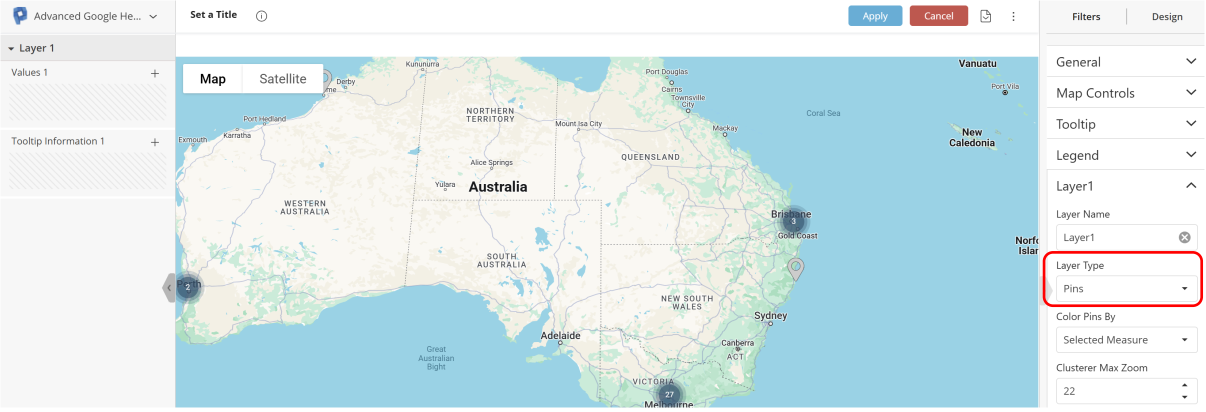
To build a scatter map:

1. Select the widget type **Advanced Google Heatmap**



Layer Name: Layer1
Layer Type: Pins
Color Pins By: Selected Measure
Clusterer Max Zoom: 22

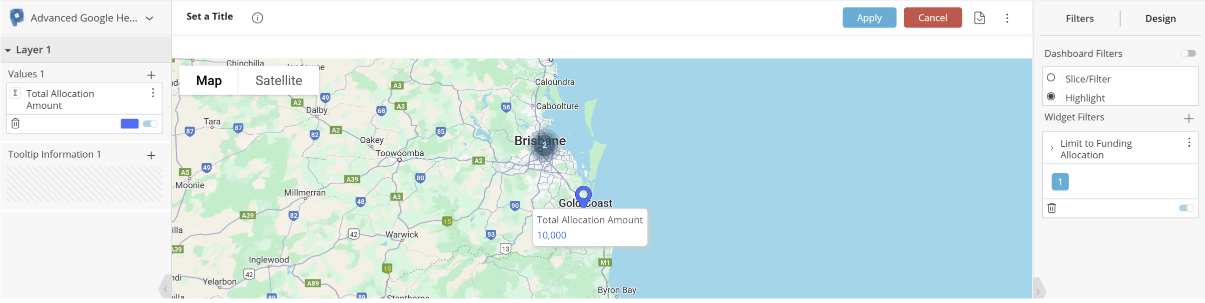
2. In the right-hand sidebar, expand **Layer1** and ensure **Layer Type** is set to “Pins”. This will generate a scatter map showing the points where addresses fall on a map.



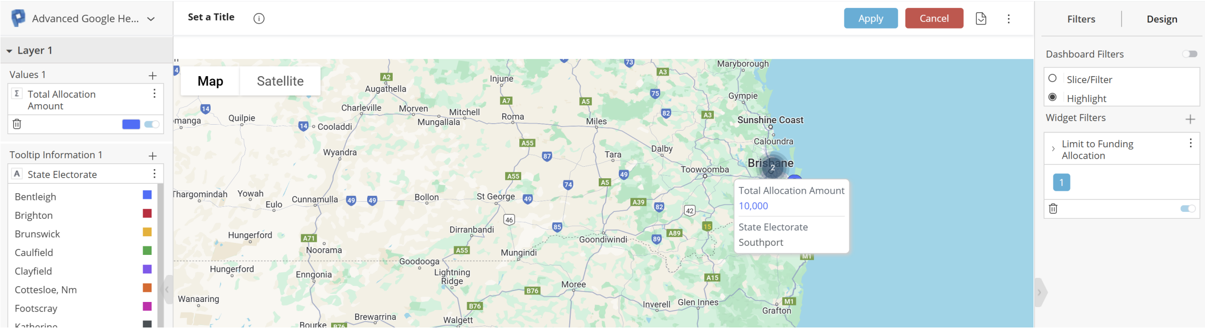
Layer Name: Layer1
Layer Type: Pins
Color Pins By: Selected Measure
Clusterer Max Zoom: 22

3. In the left-hand sidebar, add a field that you would like to aggregate (e.g. sum, average) per point to the **Values 1** box.

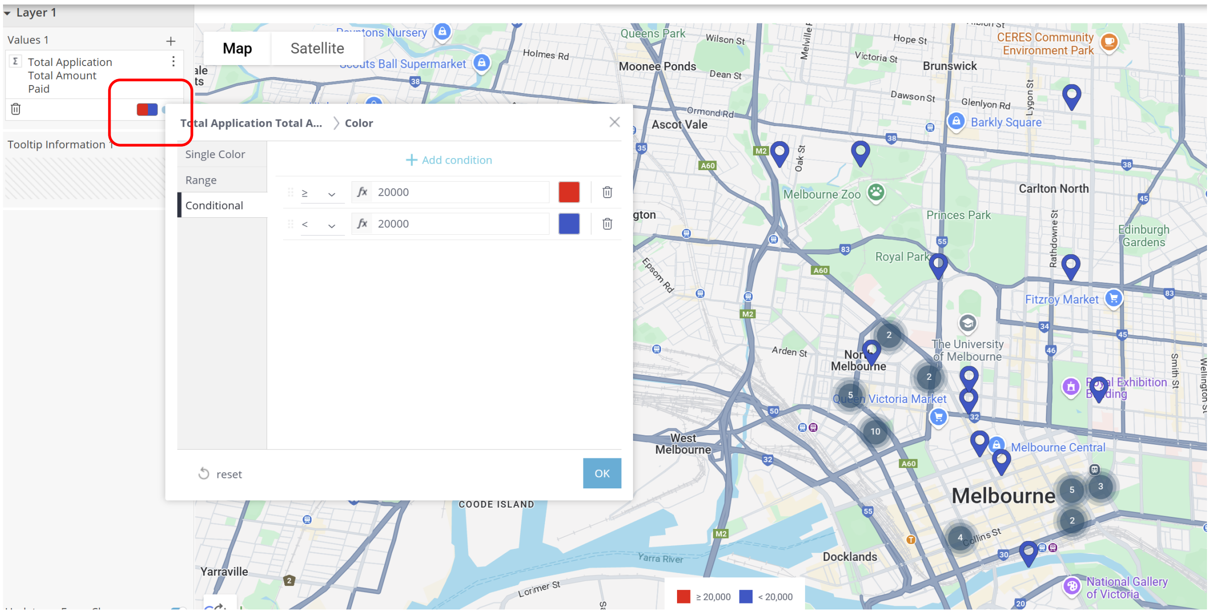
- In the image below, "Allocation Amount" from the Funding Allocation table has been added to the Values box. When a mouse hovers over an address, the tooltip now shows the total funding allocation amount associated with the application that listed this address as an Applicant Primary or Project Location address



4. Don't forget to add a Limit To widget filter as usual.
5. To add other information to the tooltip that appears when hovering over an address, add a field the **Tooltip Information 1** box in the lefthand sidebar.
   - For example, in the image below, "State Electorate" from the Location table has been added to the **Tooltip Information 1** box. When a mouse hovers over an address, the tooltip now also specifies the state electorate associated with it.



6. You can set rules for determining the colours of pinpoints. If you would like the colour of a pinpoint to be based on the value it is aggregating, click on the small box in the bottom right of the **Values 1** box. Selecting **Range** colours pinpoints along a gradient, while selecting **Conditional** allows for the input of specific rules.
   - In the example below, a pinpoint will be red if more than $20,000 in funding has been allocated to the application associated with the address.



Layer 1

Values 1

Total Application Total Amount Paid

Tooltip Information

Total Application Total A... 〉 Color

Single Color

Range

Conditional

+ Add condition

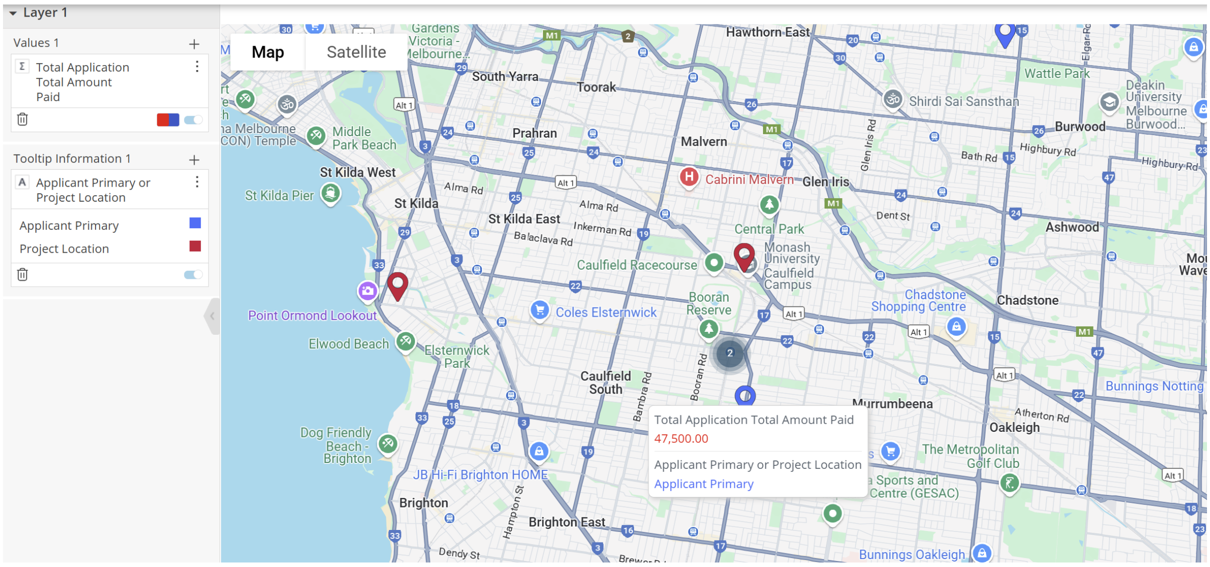
≥	*fx* 20000	🟥	🗑
<	*fx* 20000	🟦	🗑

reset

OK

🟥 ≥ 20,000 🟦 < 20,000

- You can also colour a pinpoint by a non-numerical attribute. In the example below, red pinpoints signify a Project Location while blue pinpoints signify an Applicant Primary location.



Layer 1

Values 1

Total Application Total Amount Paid

Tooltip Information 1

Applicant Primary or Project Location

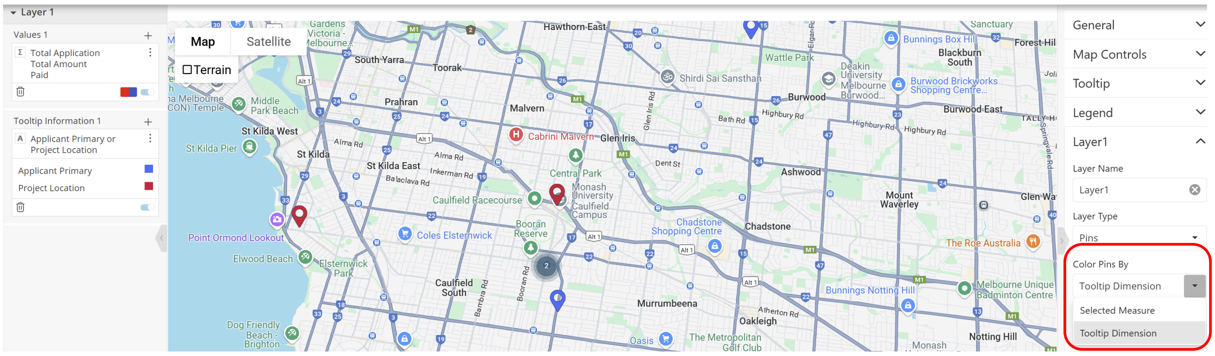
Applicant Primary 🟦

Project Location 🟥

Total Application Total Amount Paid
47,500.00

Applicant Primary or Project Location
Applicant Primary

- To achieve, this add the relevant grouping field to the Tooltip Information box (in this example, "Applicant Primary or Project Location" was added.) Then select Colour Pins by Tooltip Dimension" in the righthand sidebar.



Layer 1

Values 1

Total Application Total Amount Paid

Tooltip Information 1

Applicant Primary or Project Location

Applicant Primary 🟦

Project Location 🟥

General

Map Controls

Tooltip

Legend

Layer1

Layer Name: Layer1

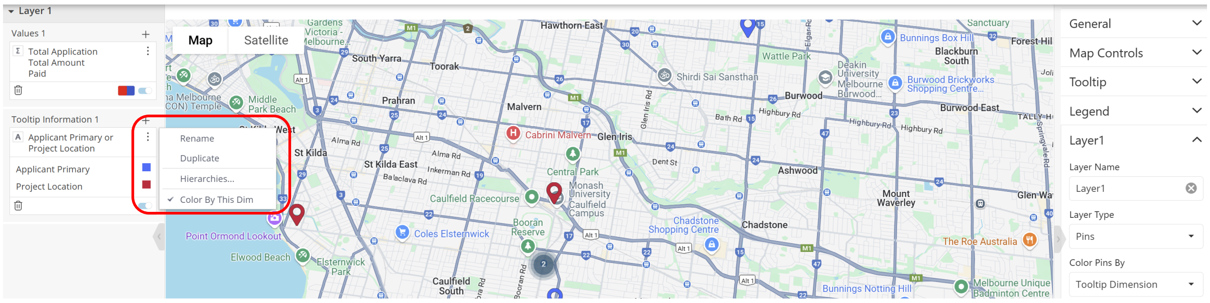
Layer Type: Pins

Color Pins By: Tooltip Dimension

Selected Measure

Tooltip Dimension

- Then under Tooltip Information in the lefthand side bar, select the three dots and "Color By This Dim"



Layer 1

Values 1

Total Application Total Amount Paid

Tooltip Information 1

Applicant Primary or Project Location

🟦 Applicant Primary

🟥 Project Location

Layer1

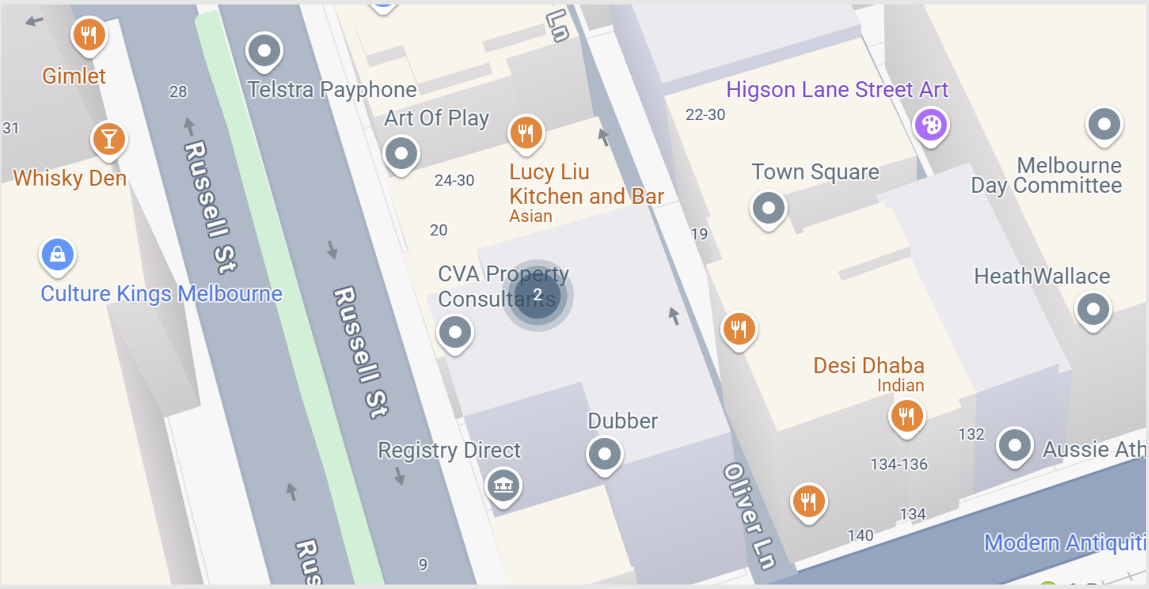
Layer Name: Layer1

Layer Type: Pins

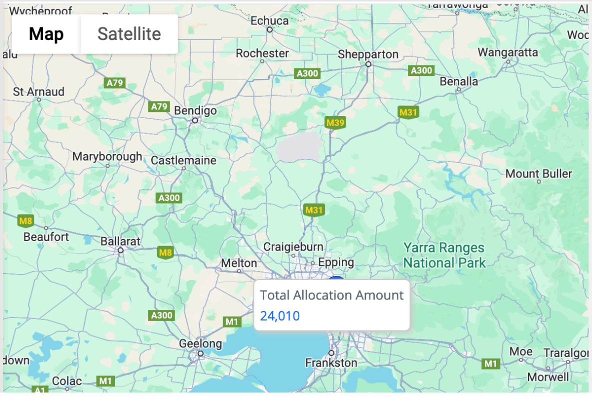
Color Pins By: Tooltip Dimension

Overlapping addresses

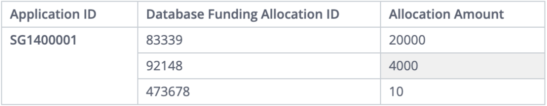
If more than one application has listed the same address as an Applicant Primary or Project Location, these addresses will be clustered and cannot be separated out into pinpoints.



If however a single application has multiple allocations/payments, these can be aggregated towards a single address. For example, in the example below the application has three funding allocations collectively totaling $24,010. When hovering over the application's Applicant Primary address, the tooltip shows $24,010.



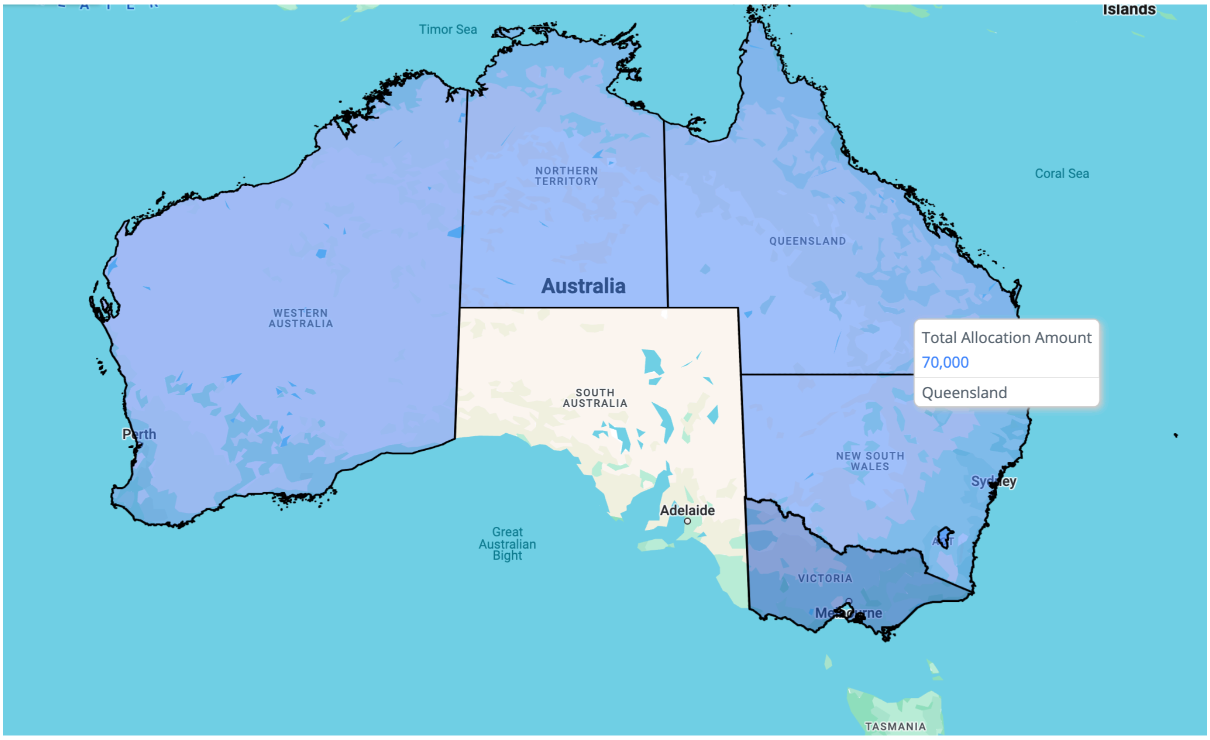
Total Allocation Amount
24,010



Application ID	Database Funding Allocation ID	Allocation Amount
SG1400001	83339	20000
	92148	4000
	473678	10

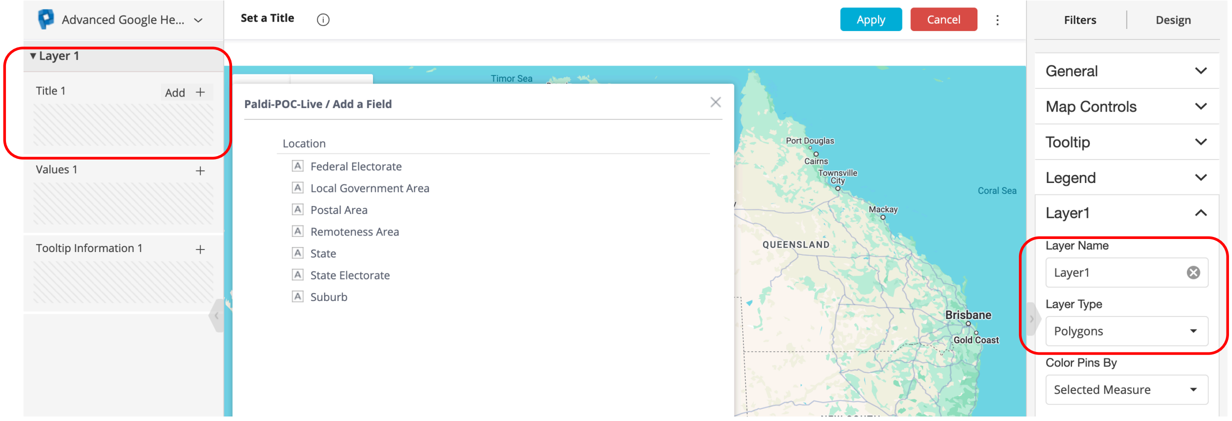
Building a heatmap

Heatmaps can be used to visualise data associated with areas or boundaries where Applicant Primary/ Project Location addresses fall. The heatmap below visualises funding allocations by state for example, with darker shading indicating that Victoria receives more funding compared to other states. Hovering over a state generates a tooltip showing the amount of funding allocated to applications whose Applicant Primary/ Project Location addresses fall in the state.

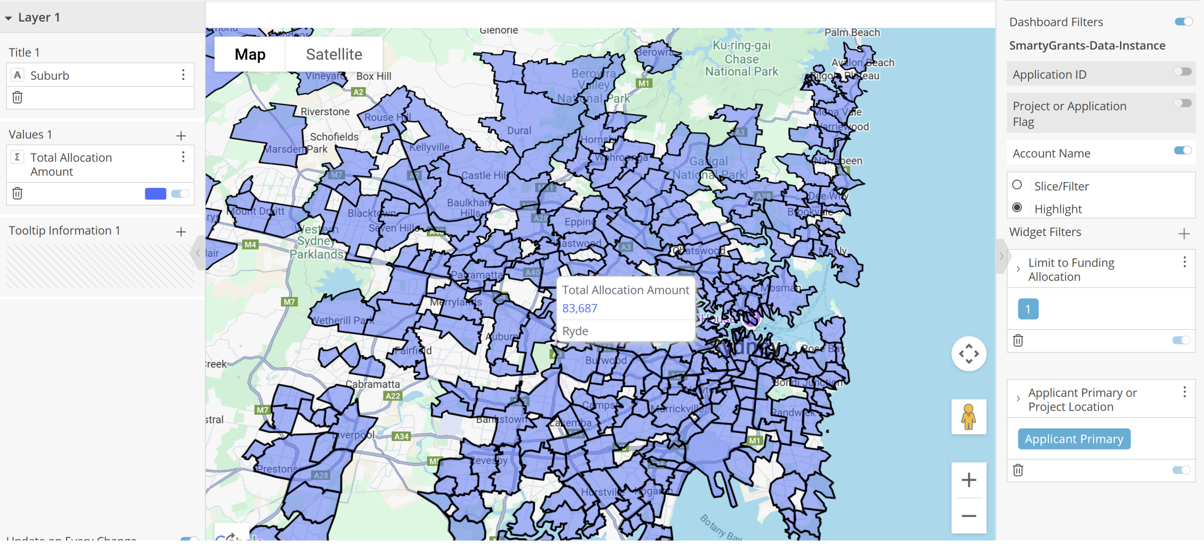


To build a heat map:

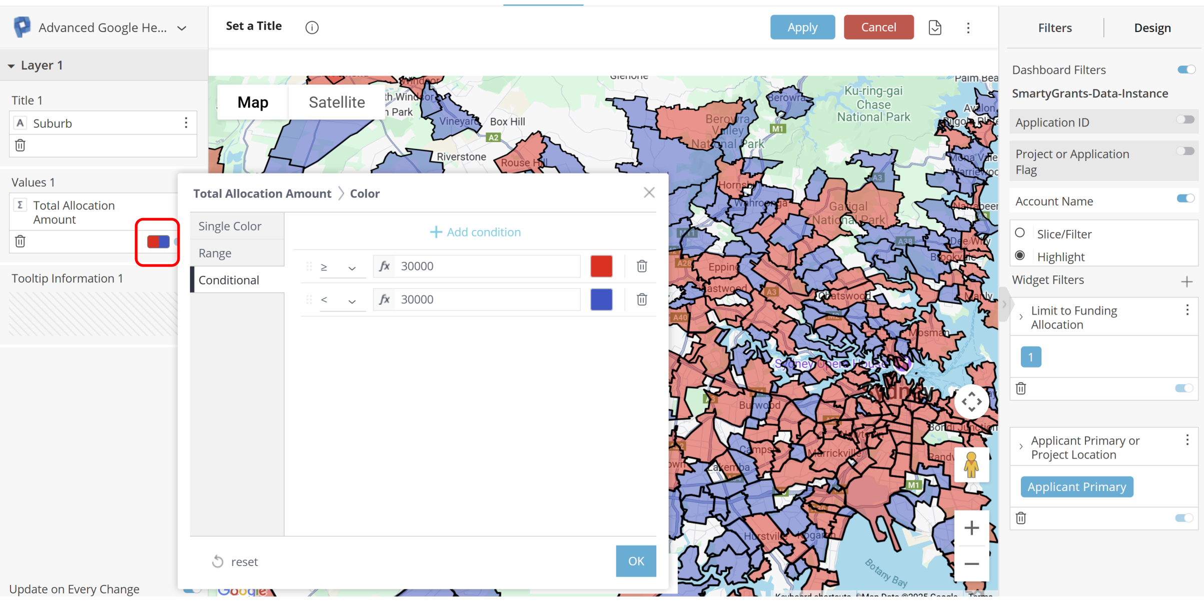
1. Select the widget type **Advanced Google Heatmap**
2. In the right-hand sidebar, select **Layer1** and ensure **Layer Type** is set to “Polygons”.
3. In the left-hand sidebar, click **Add** in the **Title 1** box which will generate a field selector. Select the boundary that you would like to visualise on your map.
   - For example, if you want to show suburbs on the map, select “Suburb” from the field selector.



4. Since we are grouping data by a boundary, add “Applicant Primary or Project Location” as a widget filter and filter for either “Application Primary” or “Project Location” addresses (not both).
5. In the left-hand sidebar, add a field that you would like to aggregate (e.g. sum, average) per boundary to the **Values 1** box. If you are visualising Project Location addresses, remember to add in a formula to remove double counting resulting from applications having multiple project locations.
   - In the image below, “Allocation Amount” from the Funding Allocation table has been added to the **Values 1** box. When a mouse hovers over a suburb, the tooltip will show the amount of funding associated with applications whose Applicant Primary addresses fall within this suburb.



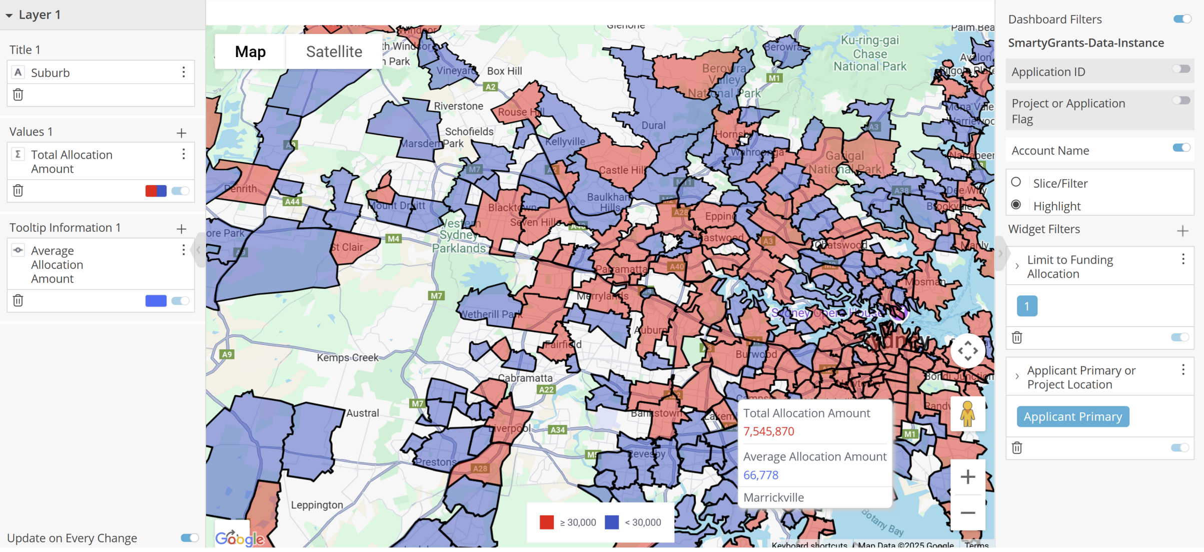
6. You can set rules for determining the colours of boundaries. If you would like the colour of a boundary to be based on the value it is aggregating, click on the small box in bottom right of the **Values 1** box. Selecting **Range** colours pinpoints along a gradient, while selecting **Conditional** allows for the input of more specific rules.
   - In the example below, a boundary will be red if more than $30,000 in funding has been allocated to applications whose Applicant Primary address fall with the boundary.



≥ 30000 🟥

< 30000 🟦

7. To add other information to the tooltip that appears when hovering over a boundary, add a field the **Tooltip Information 1** box in the lefthand sidebar.
   - For example, in the image below, “Allocation Amount” from the Funding Allocation table (with the Average function applied) has been added to the **Tooltip Information 1** box. When a mouse hovers over a suburb, the tooltip now also specifies the average amount of funding that an application typically receives in the suburb.



Layer 1

Title 1

Suburb

Values 1

Total Allocation Amount

Tooltip Information 1

Average Allocation Amount

Update on Every Change

Total Allocation Amount
7,545,870

Average Allocation Amount
66,778

Marrickville

🟥 ≥ 30,000 🟦 < 30,000

Dashboard Filters

SmartyGrants-Data-Instance

Application ID

Project or Application Flag

Account Name

○ Slice/Filter

● Highlight

Widget Filters

Limit to Funding Allocation

1

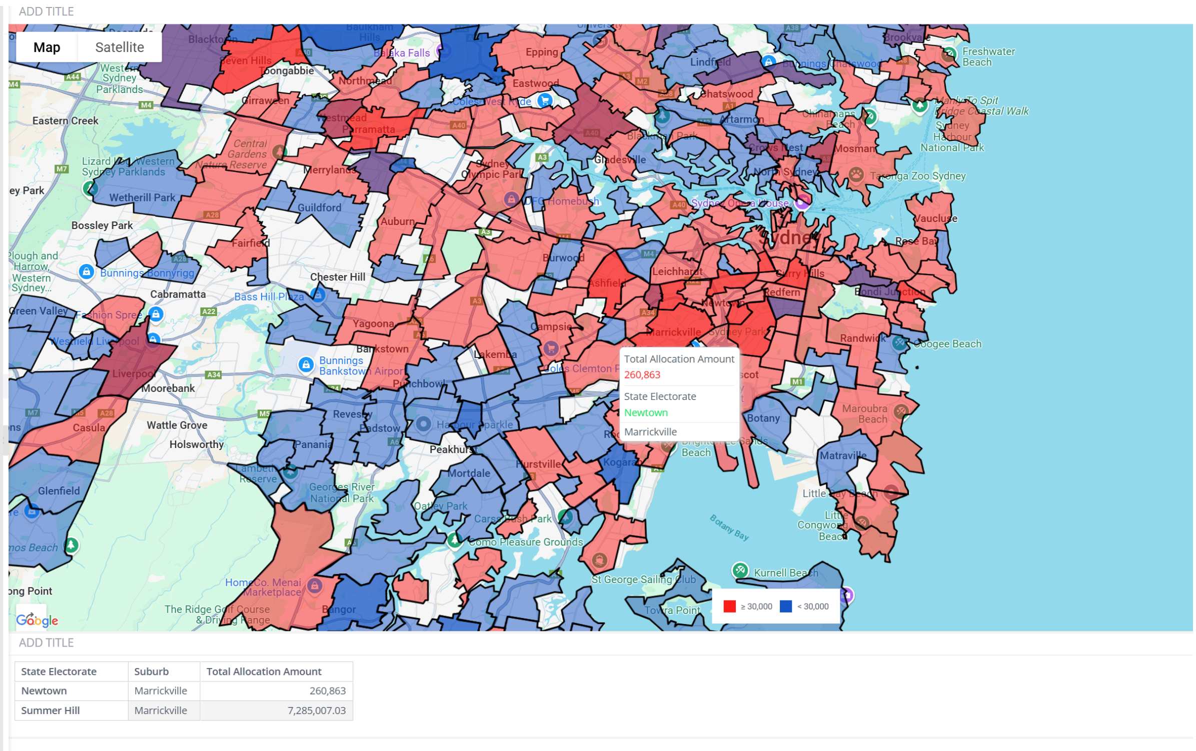
Applicant Primary or Project Location

Applicant Primary

Warning: when building a heatmap, only add a field to the Tooltip Information box if it has a unique value for every value in the Title box (a one-to-one relationship between the fields.)

For example, in a suburb heatmap, it would not be advised to add “State Electorate” to **Tooltip Information 1**. This is because a suburb can belong to multiple state electorates (there is not a one-to-one relationship between suburbs and state electorates.)

- The Suburb of Marrickville is split between the state electorates of Newtown and Summerhill. When hovering over the suburb of Marrickville, the tooltip would arbitrarily list Newtown or Summerhill as Marrickville’s state electorate, because the tooltip is only able to include one value per boundary.
- Breaking this rule can lead to inaccurate aggregation of figures in the tooltip. In the image below, the tooltip shows $250,863 which is the funding associated with applications with addresses in the Newtown section of Marrickville, excluding the $7,285,007 in funding associated with applications with addresses in the Summerhill side of Marrickville.

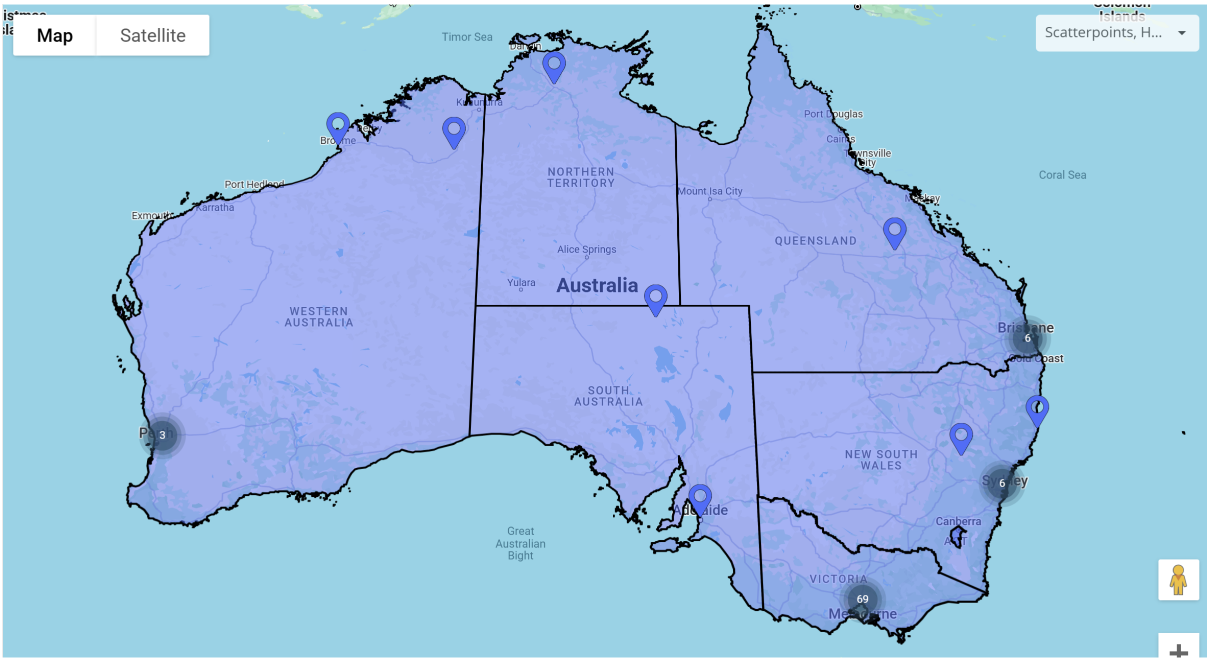


State Electorate	Suburb	Total Allocation Amount
Newtown	Marrickville	260,863
Summer Hill	Marrickville	7,285,007.03

Please note that among the boundaries in the Location table, there only exists one-to-one relationships between State and the other boundaries (for example, there is a one-to-one relationship between State and Postal Area, but there is not a one-to-one relationship between Postal Area and Suburb.)

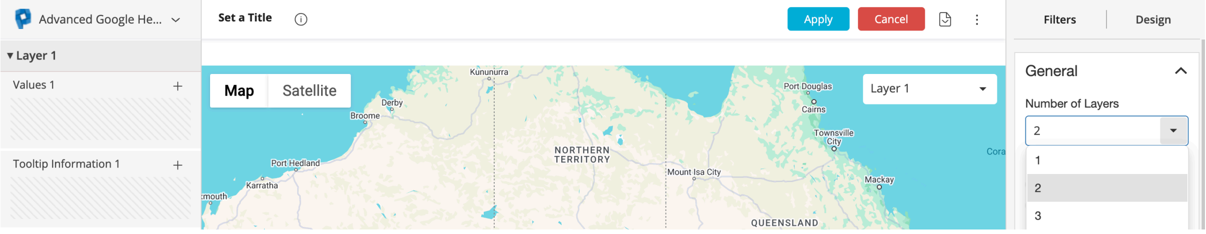
Building a multi-layer map

The Advanced Google Heatmap allows you to layer scatter points on top of a heatmap in a two-layer map. Two-layer maps allow you to toggle between seeing scatter points on top of a heatmap, seeing scatter points without a heatmap, and a heatmap without scatter points.



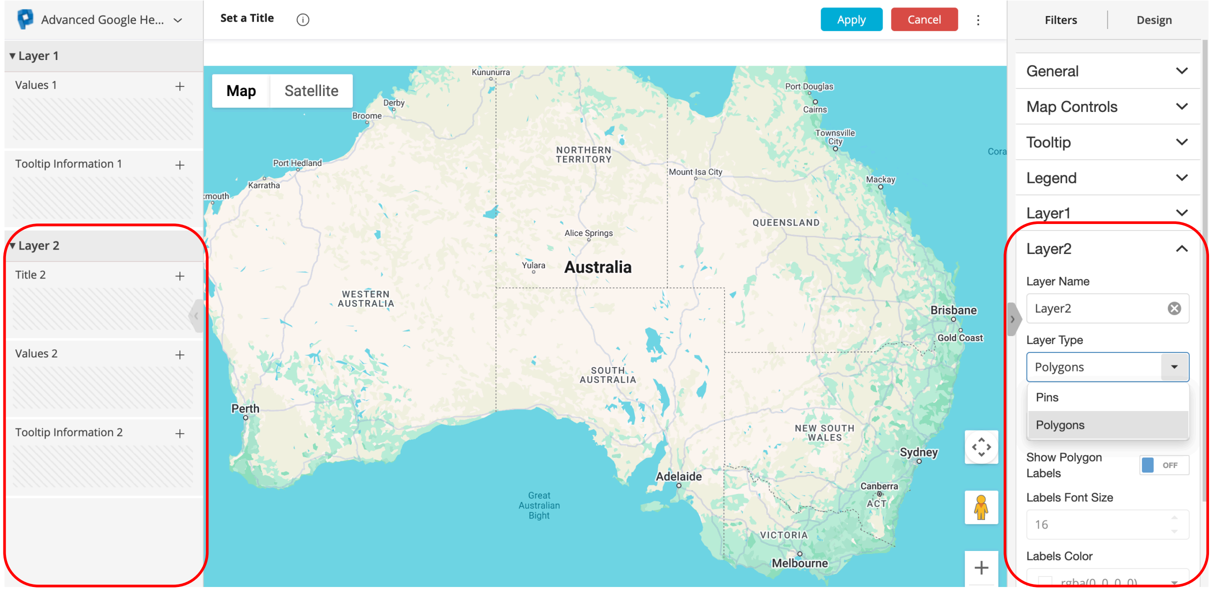
To build a two-layer map:

1. Select the widget type **Advanced Google Heatmap**
2. In the right-hand sidebar, select **General** and select “2” for **Number of Layers.**



Number of Layers: 2

8. This adds a second section **Layer 2** to both the right-hand toolbar and the left-hand sidebar. In the right-hand sidebar, expand **Layer1** and set **Type** to “Pins” and follow the instructions for building a scatter map in the **Layer 1** section of the left-hand sidebar. In the right-hand sidebar, expand **Layer2** and set **Type** to “Polygons”, then follow the instructions for building a heatmap in the **Layer 2** section of the left-hand sidebar.



Layer Name: Layer2

Layer Type: Polygons

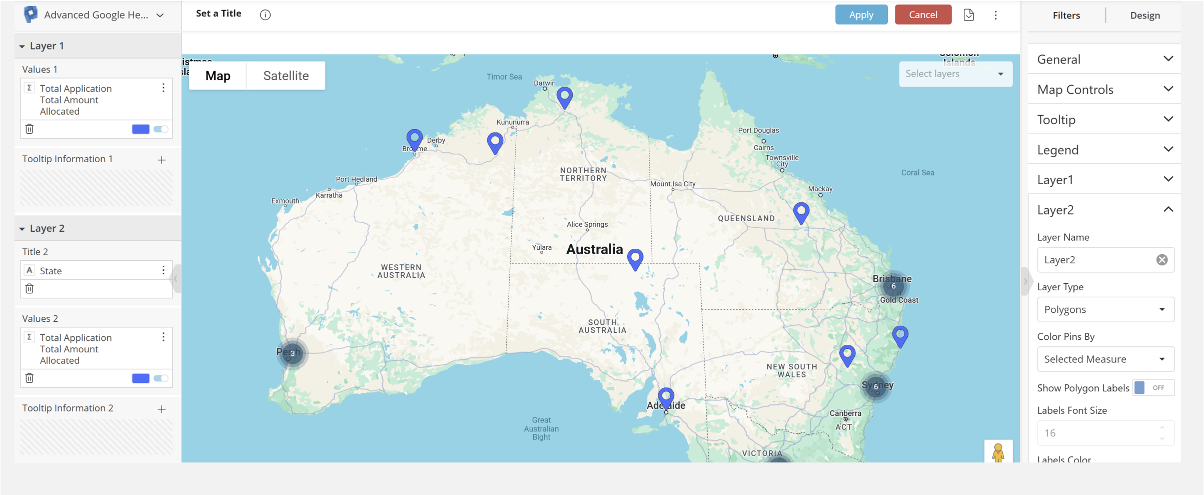
Show Polygon Labels: OFF

Labels Font Size: 16

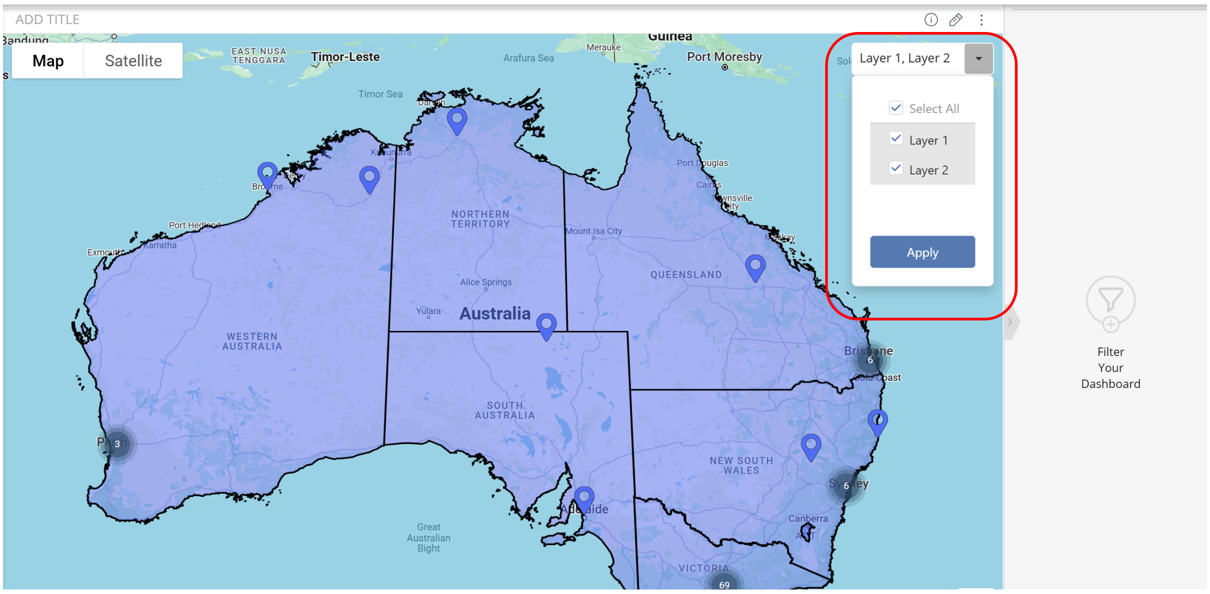
Labels Color: rgba(0, 0, 0, 0)

When editing a widget, the map defaults to showing only Layer 1. The drop-down menu to select layers can only be interacted with while viewing the dashboard.

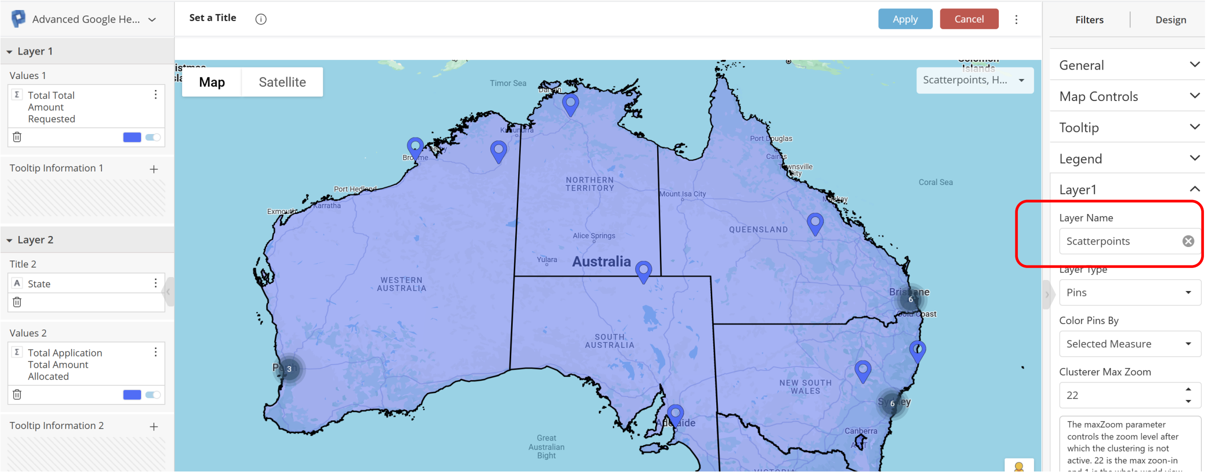
Though there is a “Select layers” dropdown selector in the top-right hand corner of the map, this is not clickable while building a widget. To change the layers visible within edit mode, exit to the dashboard and select layers using the dropdown selector and then re-enter the widget builder.



9. Exit the widget builder and while viewing the map widget in the dashboard, ensure Layer 1 and Layer 2 are ticked in the top right drop box to visualise both layers at once. If you would like to toggle between seeing the heatmap without scatter points or vice versa, tick only one layer.



10. Back in the widget builder, you can change the names of layers in the right-hand toolbar to make it easier to identify layers in the dropdown menu.



Layer 1

Values 1: Total Total Amount Requested

Tooltip Information 1

Layer 2

Title 2: State

Values 2: Total Application Total Amount Allocated

Tooltip Information 2

Layer Name: Scatterpoints

Layer Type: Pins

Color Pins By: Selected Measure

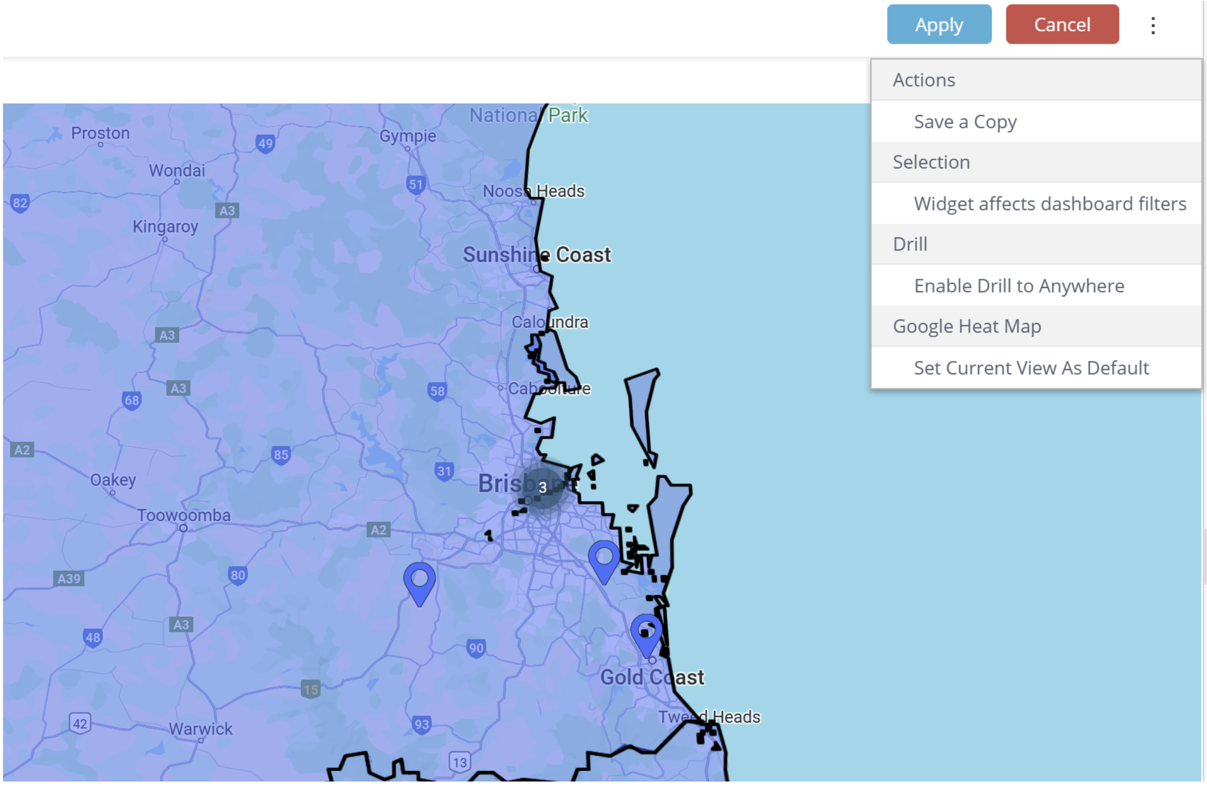
Clusterer Max Zoom: 22

The maxZoom parameter controls the zoom level after which the clustering is not active. 22 is the max zoon-in

Other map features

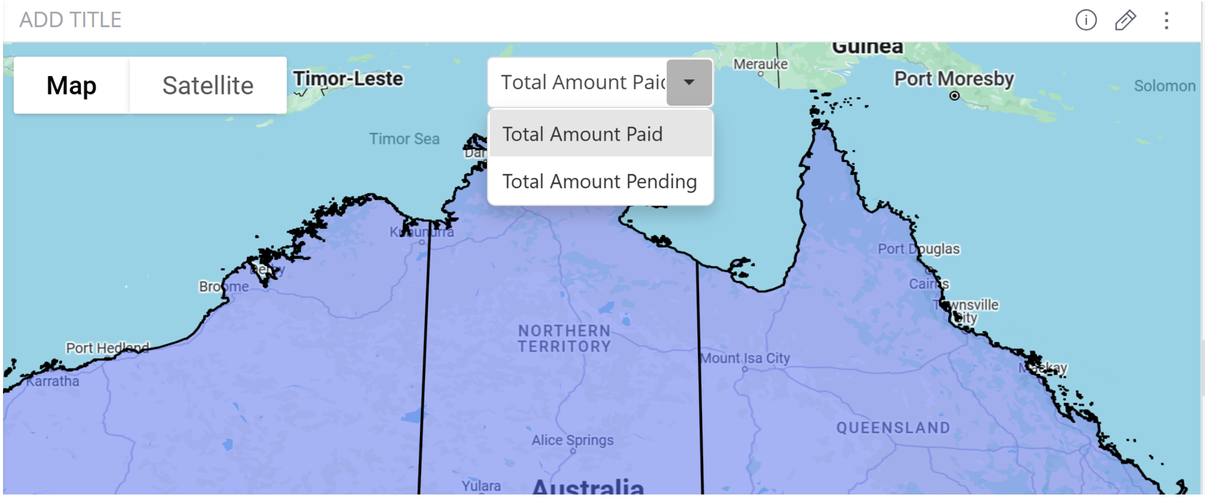
Default load view

If you would like a map widget to always load centred on a certain location, in the widget builder move the map to that location. Then in the top right-hand corner click on the three dots to generate a dropdown menu and select **Set Current View as Default**. Each time the below map is refreshed, for example, it will open centred on Brisbane.

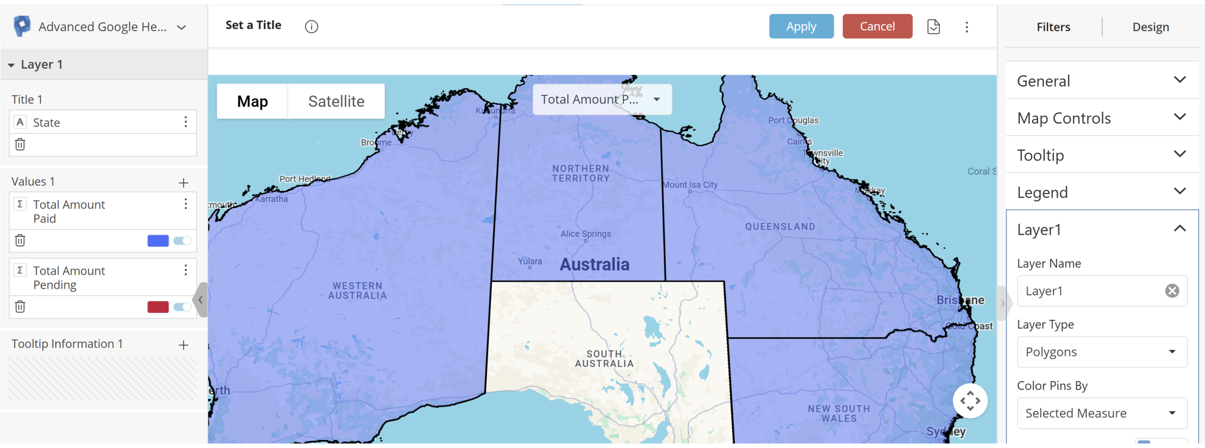


Multiple values in a single-layer map

In a single-layer map, it is possible to allow views to toggle between different fields to be aggregated. In the map below for example, a user can toggle between a map showing the sum of paid payments per state, and a map showing the sum of pending payments per state.



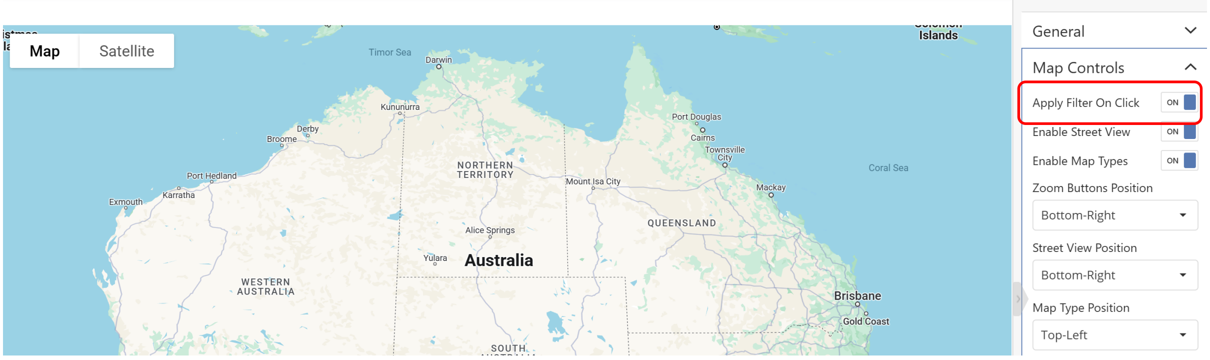
To achieve this, simply add more than one field to the **Values 1** box in the left-hand side bar when building a widget. This will generate a drop-down menu to toggle between fields to be aggregated at the top of the map, but this menu can only be interacted with when viewing the dashboard (once exiting the widget builder.)



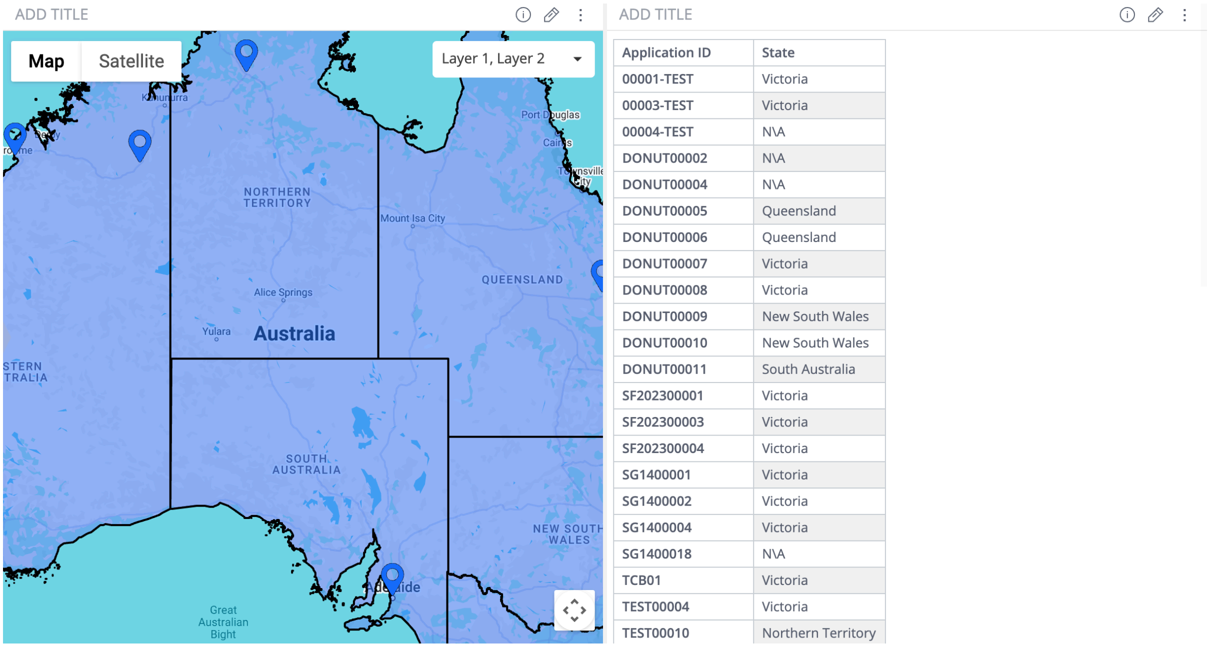
Map as filter

Turning on "Apply Filter On Click" enables interactive filtering on a map. This means in a dashboard:

- If a user clicks on a boundary in a heatmap, this will automatically filter the rest of the dashboard to only include data from the boundary
- If a user clicks on a pinpoint in a scatter map, this will automatically filter the rest of the dashboard to only include data for the application associated with the pinpoint.

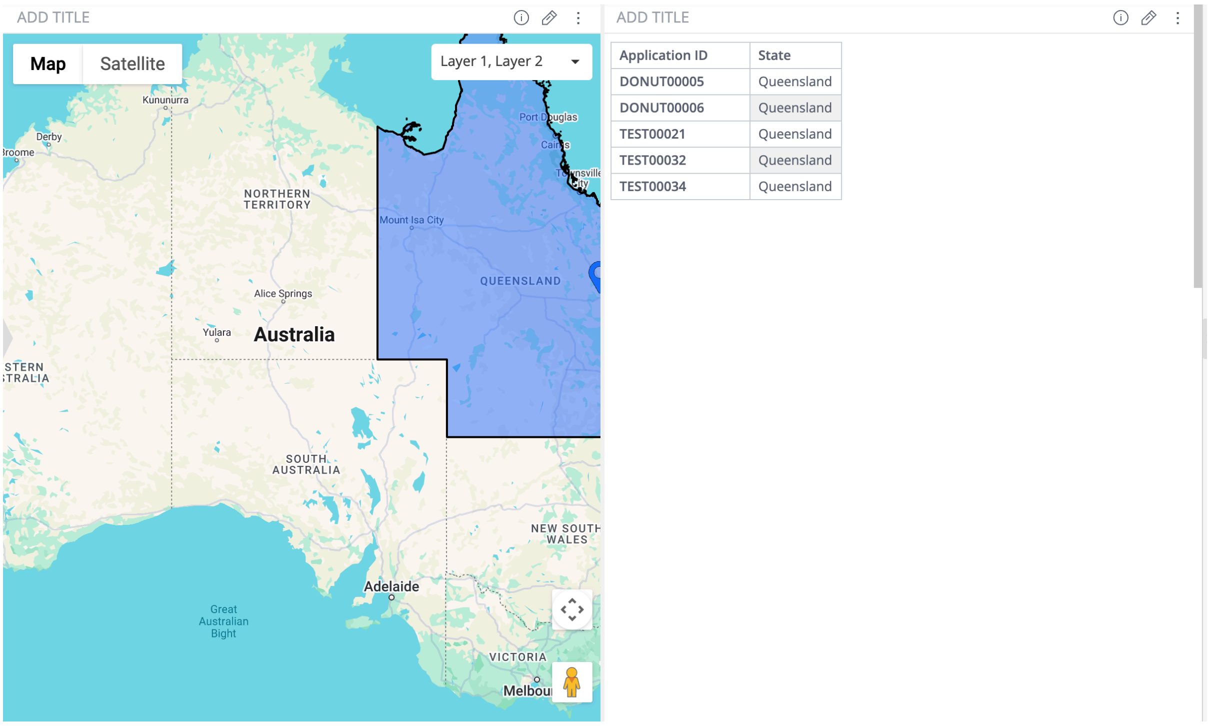


For example, the below dashboard contains a state heatmap and a list of application IDs.



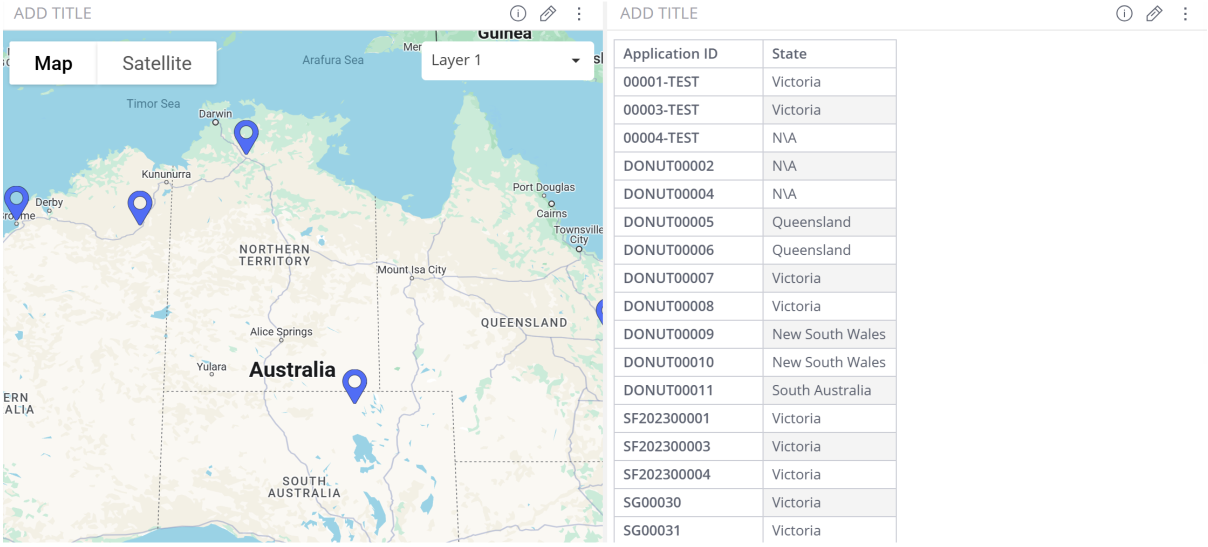
Application ID	State
00001-TEST	Victoria
00003-TEST	Victoria
00004-TEST	N\A
DONUT00002	N\A
DONUT00004	N\A
DONUT00005	Queensland
DONUT00006	Queensland
DONUT00007	Victoria
DONUT00008	Victoria
DONUT00009	New South Wales
DONUT00010	New South Wales
DONUT00011	South Australia
SF202300001	Victoria
SF202300003	Victoria
SF202300004	Victoria
SG1400001	Victoria
SG1400002	Victoria
SG1400004	Victoria
SG1400018	N\A
TCB01	Victoria
TEST00004	Victoria
TEST00010	Northern Territory

Clicking on the state of Queensland filters all other widgets in the dashboard, reducing the list of application IDs to only those with an address in Queensland.



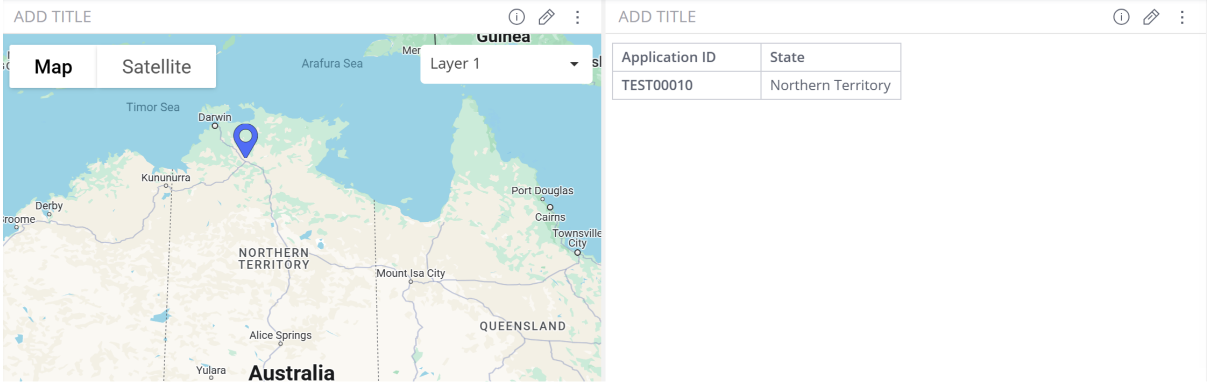
Application ID	State
DONUT00005	Queensland
DONUT00006	Queensland
TEST00021	Queensland
TEST00032	Queensland
TEST00034	Queensland

In the dashboard below, there is a scatter map with a list of application IDs (with funding allocations associated with each application.)



Application ID	State
00001-TEST	Victoria
00003-TEST	Victoria
00004-TEST	N\A
DONUT00002	N\A
DONUT00004	N\A
DONUT00005	Queensland
DONUT00006	Queensland
DONUT00007	Victoria
DONUT00008	Victoria
DONUT00009	New South Wales
DONUT00010	New South Wales
DONUT00011	South Australia
SF202300001	Victoria
SF202300003	Victoria
SF202300004	Victoria
SG00030	Victoria
SG00031	Victoria

Clicking on a scatter point filters other widgets in the dashboard, reducing the list of application IDs to the application associated with the address that was clicked on.



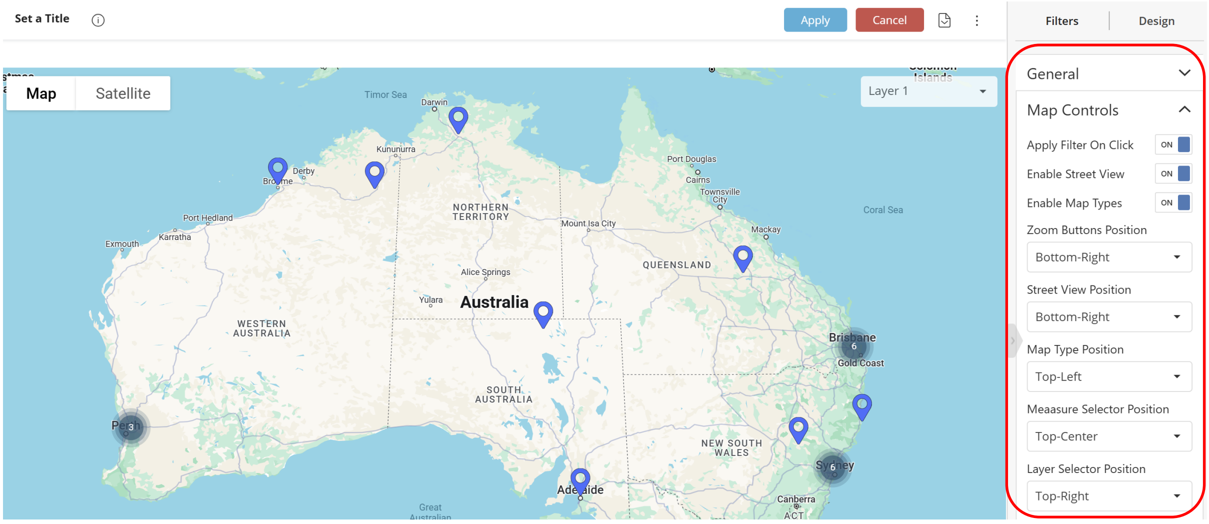
Application ID	State
TEST00010	Northern Territory

If an application has multiple addresses on the map, clicking on a scatter point will cause the map to only display one address (the address that was clicked on.)

For example, if clicking on the Applicant Primary address of an application (with no “Applicant Primary or Project Location” filter applied on the map), project location addresses associated with the application disappear from the map.

Customising maps

Customise the look, style and functionality available to viewers of maps you build in the right-hand toolbar. Expand **Map Controls** to customise the placement of buttons on the map and disable functionality for the viewer. Expand the section **Tooltip** to customise the font, size etc. of tooltips. Expand **Legend** to customise the placement, size etc. of legends.



General

Map Controls

Apply Filter On Click ON

Enable Street View ON

Enable Map Types ON

Zoom Buttons Position: Bottom-Right

Street View Position: Bottom-Right

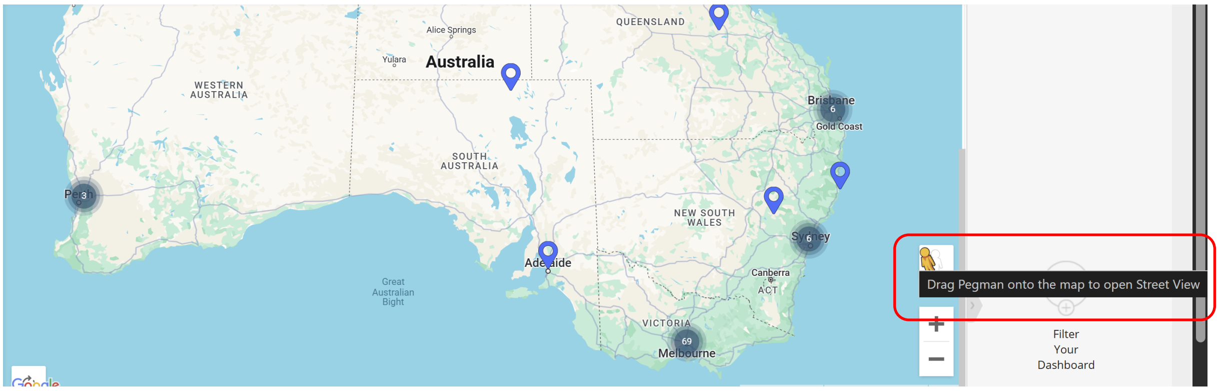
Map Type Position: Top-Left

Meaasure Selector Position: Top-Center

Layer Selector Position: Top-Right

Google Streetview

The Advanced Google Heatmap has a Google Streetview functionality that allows viewers to see the street view associated with a pinpoint, or any other part of a map. To use this function, drag the yellow peg in the bottom righthand corner of a map onto a point of the map.

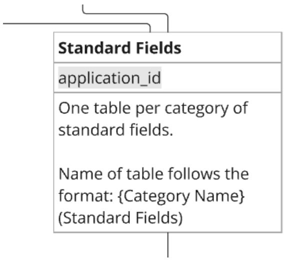


Drag Pegman onto the map to open Street View

15 Dynamic Tables

Tables are created “dynamically” for entities where the number and name of reportable fields varies from instance to instance. Examples include form-related data (where instance administrators can choose the number of questions in a form and the text of those questions) and standard fields (where instance administrators can choose the number of standard fields to set up and the text of those standard fields).

Dynamic tables are represented in the data model diagram by white tables. A white table indicates that there may be several of each table type in an instance’s actual data model. In the diagram below, “Standard Fields” represents a table that is dynamically created to capture responses to standard fields. If an instance has five categories of standard fields, there would be five tables of this type in its data model.



Standard Fields

application_id

One table per category of standard fields.

Name of table follows the format: {Category Name} (Standard Fields)

The naming structure for tables can be found in the data model diagram. The name of dynamic tables and the fields within them depends on how entities have been set up in an instance (e.g. varying by form name, by section name.)

- For example, the naming structure for standard field tables is {Category Name} (Standard Fields). If there is a category of standard fields in an instance called "Historical Information", this would translate to a table name of Historical Information (Standard Fields).

Why are there sometimes multiple tables for a single form/standard section/ standard field?

Fields from a single form/standard section/ standard field are generally split into tables by question type (single-response vs multi-choice questions.) This split across multiple tables minimises data duplication that would occur if we placed questions/sections allowing multiple responses and questions allowing a single response in the same table.

The following sections cover how fields are split across tables in detail, however it is not necessary to understand how and why fields are organised to use them. You can always **find a field by searching the relevant label/name** (question, section name, form name, standard field category etc.) in the widget's field selector.

Refreshing data in dynamic tables

Data related to dynamic tables, including both:

- New responses
- New sections and fields/questions

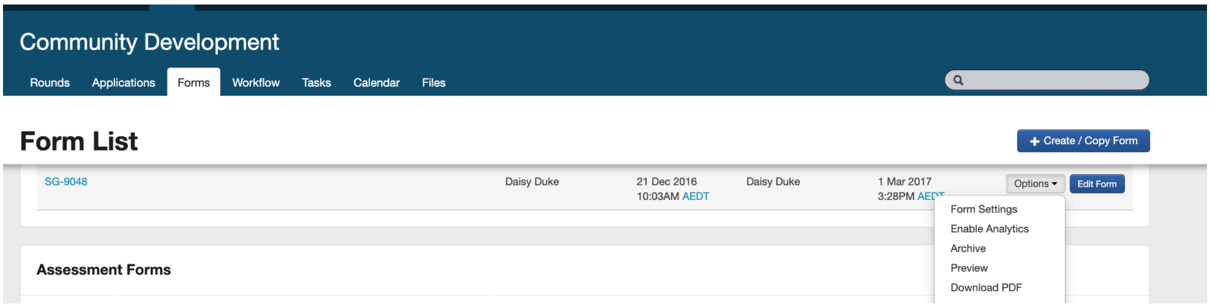
are refreshed nightly (the nightly refresh window is 1am - 6am. However accounts are enabled progressively and most accounts should refresh within an hour.) This means if you enable analytics for a form, responses to the form will appear in SmartyGrants Analytics the next day. If you make a change to a form for which analytics is already enabled (e.g. removing a question, adding a standard field), these changes will not be reflected in SmartyGrants Analytics until the next day.

16 Forms

Enabling reporting on forms

SmartyGrants Analytics facilitates reporting on responses to forms. However, data only appears in SmartyGrants Analytics for forms which have reporting enabled (responses to a form will not appear in SmartyGrants Analytics automatically.)

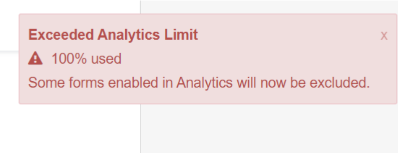
To enable reporting on a form, go to the **Forms** tab of SmartyGrants. Select **Options** for the form you want to enable reporting on and click **Enable Analytics**.



SG-9048 Daisy Duke 21 Dec 2016 10:03AM AEDT Daisy Duke 1 Mar 2017 3:28PM AEDT

Forms which were enabled for reporting then archived still appear in Analytics to allow for reporting on historical responses.

Warning: users should only enable Analytics for forms that they need to report on (do not enable reporting for every form.) There is a limit to each instance's data model file size. If the limit is exceeded, some of the oldest forms (least recently modified among the analytics enabled forms) will be dropped from the data model to bring the file size within the limit. The dropped forms will not appear for selection when building a widget even if they were enabled for reporting (but more recently modified forms will still appear.)



Exceeded Analytics Limit x
⚠ 100% used
Some forms enabled in Analytics will now be excluded.

It is difficult to advise on the number of forms that can be enabled for reporting since each form contains a different number of fields. However, a warning message will appear on the Analytics page if an instance is nearing the limit.

What should you do when the data model size limit is exceeded?

- Review archived forms and if no longer needed for reporting, reactivate and disable analytics
- Review current forms and disable analytics.

If you would like a list of forms which have been dropped from reporting due to exceeding the data model size limit, please contact the SmartyGrants support team.

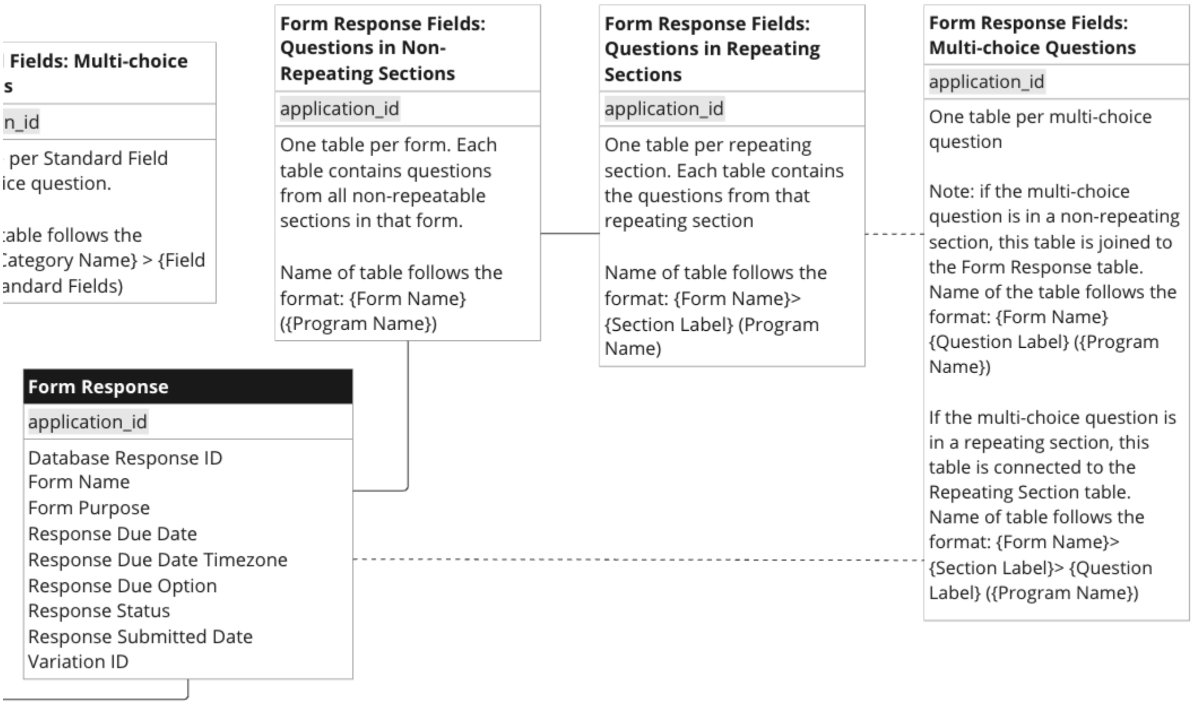
Form fields in the data model

When building a widget, the easiest way to find a form-related field is to simply type part of the question label or form name into the field selector. Fields from a single form may sometimes be split across multiple tables to minimise data duplication (which would occur if we placed questions/sections allowing multiple responses and questions allowing a single response in the same table.) However, it is not necessary to understand how and why fields are organised to use them.

Form Response Table

The Form Response table contains "metadata" on responses submitted to forms (that is, it does not contain responses to questions in forms, but information about the responses, such as time of submission and the form name). Please note that an applicant can submit the same form multiple times, which would generate multiple rows in the table.

Dynamic form tables



Fields: Multi-choice
s

n_id

per Standard Field
ice question.

:able follows the
:ategory Name} > {Field
andard Fields)

Form Response Fields: Questions in Non-Repeating Sections

application_id

One table per form. Each table contains questions from all non-repeatable sections in that form.

Name of table follows the format: {Form Name} ({Program Name})

Form Response Fields: Questions in Repeating Sections

application_id

One table per repeating section. Each table contains the questions from that repeating section

Name of table follows the format: {Form Name}> {Section Label} (Program Name)

Form Response Fields: Multi-choice Questions

application_id

One table per multi-choice question

Note: if the multi-choice question is in a non-repeating section, this table is joined to the Form Response table. Name of the table follows the format: {Form Name} {Question Label} ({Program Name})

If the multi-choice question is in a repeating section, this table is connected to the Repeating Section table. Name of table follows the format: {Form Name}> {Section Label}> {Question Label} ({Program Name})

Form Response

application_id

Database Response ID
Form Name
Form Purpose
Response Due Date
Response Due Date Timezone
Response Due Option
Response Status
Response Submitted Date
Variation ID

For each form, every single-response question across all non-repeatable sections are stored in a single main table. For each form, an additional table is created for:

- Every multi-choice question. Each table contains a single question
- Every repeatable section. Each table contains questions from a single repeatable section

For a single form, this can generate up to four table types (with potentially several cases of each type). The naming structure these four form table types is:

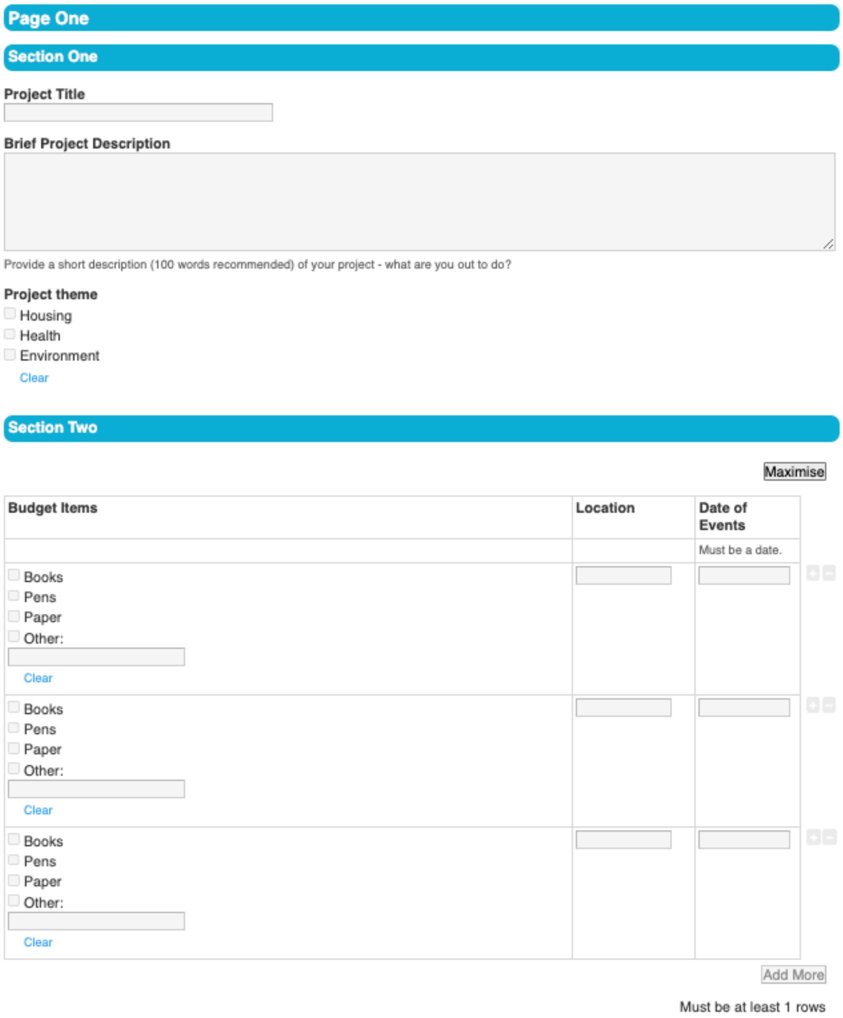
Tables for single-response questions in non-repeatable sections	Form Name (Program Name) e.g. Short Application Form (Sports Grants)
Tables for multi-choice questions in non-repeatable sections	Form Name > Question Label (Program Name) e.g. Short Application Form > Project Locations (Sports Grants)
Tables for single-response questions in repeatable sections	Form Name > Section Label (Program Name) e.g. Short Application Form > Project Details (Sports Grants)
Tables for multi-choice questions in repeatable sections	Form Name > Section Label > Question Label (Program Name) e.g. Short Application Form > Project Details > Budget Items (Sports Grants)

Note that responses to smart-choice questions are stored as a concatenated string in form tables.

- If the smart-choice question came from a non-repeatable section, responses are stored in the main table for the form containing single-response questions.

- If the smart-choice question came from a repeatable section, responses are stored in the table containing single-response questions from the repeatable section.

Consider the example of the form named “Small Grants Form” below from the program “Environmental Grants”:



Page One

Section One

Project Title

Brief Project Description

Provide a short description (100 words recommended) of your project - what are you out to do?

Project theme

□ Housing
□ Health
□ Environment
Clear

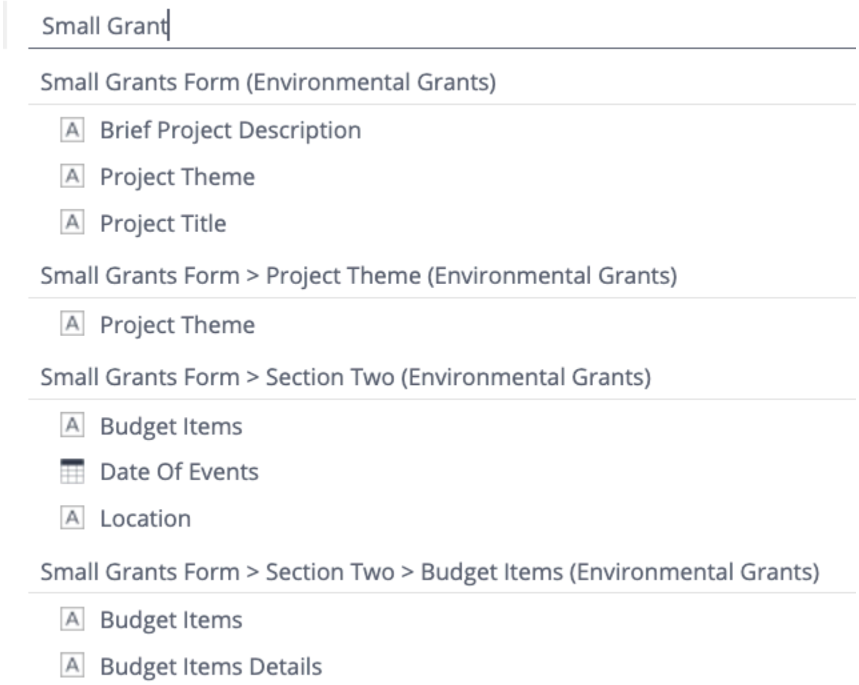
Section Two

Maximise

Budget Items	Location	Date of Events
		Must be a date.
□ Books □ Pens □ Paper □ Other: Clear		
□ Books □ Pens □ Paper □ Other: Clear		
□ Books □ Pens □ Paper □ Other: Clear		

Add More

Must be at least 1 rows



Small Grant

Small Grants Form (Environmental Grants)

- Brief Project Description
- Project Theme
- Project Title

Small Grants Form > Project Theme (Environmental Grants)

- Project Theme

Small Grants Form > Section Two (Environmental Grants)

- Budget Items
- Date Of Events
- Location

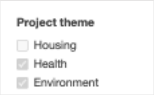
Small Grants Form > Section Two > Budget Items (Environmental Grants)

- Budget Items
- Budget Items Details

- Responses to “Project Title” and “Short Project Description” would be stored in the main table **Small Grants Form (Environmental Grants)** (the main table containing responses to single-response questions in non-repeatable sections)
- Responses to “Project Theme” would be stored in a separate table, **Small Grants Form > Project Theme (Environmental Grants)**. This is because “Project Theme” is a multi-choice question in a non-repeatable section. If there was a second multi-choice question in this non-repeatable section, another table would be generated to store responses to that question.
- Responses to “Location” and “Date of Event” would be stored in a separate table, **Small Grants Form > Section Two (Environmental Grants).** This is because these are single-response questions in a repeatable section. Note: if there was a second repeatable section, another table would be generated to store responses to single-response questions in that repeatable section.
- Responses to “Budget Items” would be stored in a separate table, **Small Grants Form > Section Two > Budget Items (Environmental Grants).** This is because this is a multi-choice question in a non-repeatable section. Note: if there was a second multi-choice question in this same non-repeatable section, another table would be generated to store responses to that question.

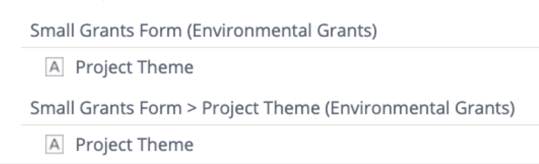
My form contains a multi-choice question. Why does this field appear in multiple tables?

As seen in the screenshot below, the multi-choice question “Project Theme” appears both in the main Small Grants Form (Environmental Grants) table (storing single-response questions from non-repeatable sections) and in a stand-alone table.



Project theme

☐ Housing
☑ Health
☑ Environment



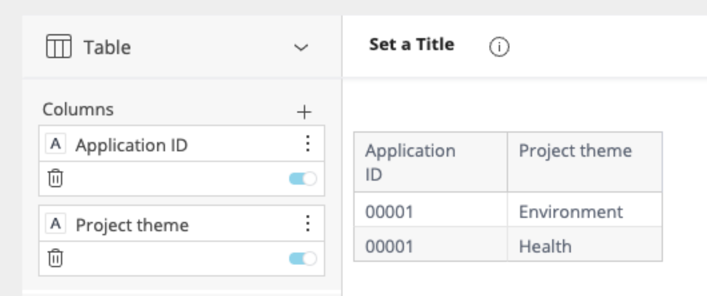
Small Grants Form (Environmental Grants)

- Project Theme

Small Grants Form > Project Theme (Environmental Grants)

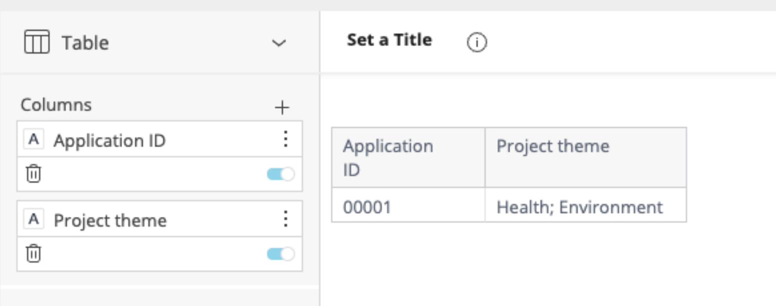
- Project Theme

In the stand-alone table, every choice/answer is separated out into a separate row. See here for an example of how multi-choice questions where the option of “other” is enabled appear.



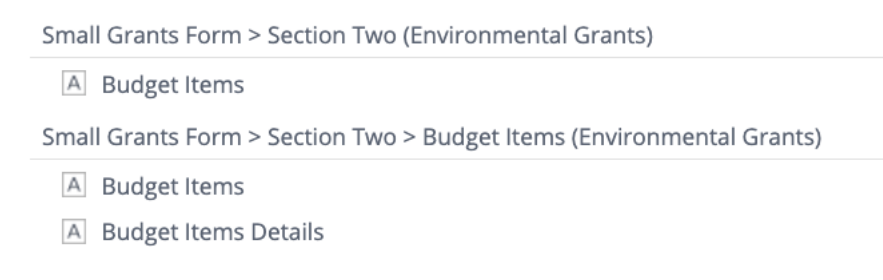
Application ID	Project theme
00001	Environment
00001	Health

Responses are also stored in a concatenated form in the main table (storing single-response questions from non-repeatable sections), where all choices are combined in a string separated by “;”.



Application ID	Project theme
00001	Health; Environment

Storing this data in different forms allows for different kinds of analysis (for example, having choices separated out into rows allows for grouping and slicing other data). Note that this storage of multi-choice questions in two different forms applies also to multi-choice questions in repeatable sections. In this case, the concatenated field is stored in the table containing single-response questions from the repeatable section.

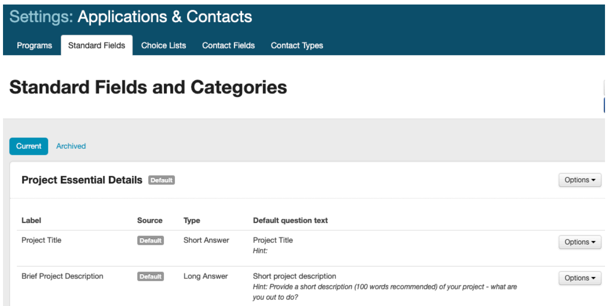

Small Grants Form > Section Two (Environmental Grants)

- Budget Items

Small Grants Form > Section Two > Budget Items (Environmental Grants)

- Budget Items
- Budget Items Details

17 Standard Fields



Standard Fields and Categories

Project Essential Details Default

Label	Source	Type	Default question text
Project Title	Default	Short Answer	Project Title *Hint:*
Brief Project Description	Default	Long Answer	Short project description *Hint: Provide a short description (100 words recommended) of your project - what are you out to do?*

Standard fields are organised in the data model by category. If you are not familiar with standard fields and categories, please see the Help Hub.

Standard fields do not need to be turned on for reporting the way forms do and are available in Analytics by default. For example, since Project Essential Details is a default standard field category, the table Project Essential Details (Standard Fields) will always be created in the data model.

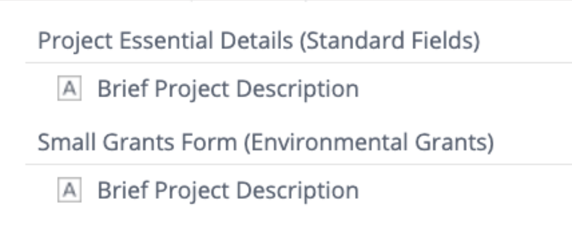
When building a widget, the easiest way to find a standard field it to simply type part of its label into the field selector. It is not necessary to understand how and why fields are organised to use them.

Please note archived standard fields are not included in reporting.

If a form contains a standard field, what is the difference between the response stored in the standard field table and the response stored in the form table?

If a form contains a standard field, it will be stored both in a form table and in a standard field table. Think of the form table version as a "point in time" record of how someone filled out a standard field, while the standard field version captures the latest response to that standard field.

Consider the standard field below "Brief Project Description," which was added to the form "Small Grants Form."



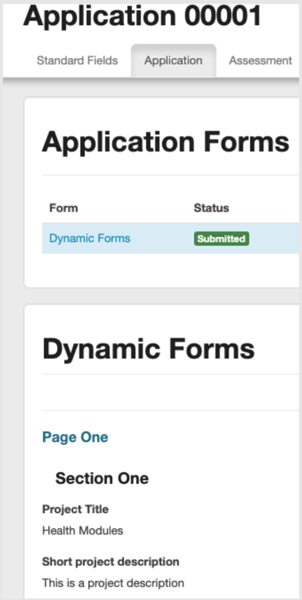
Project Essential Details (Standard Fields)

A Brief Project Description

Small Grants Form (Environmental Grants)

A Brief Project Description

If when filling out that form, an applicant answered this standard field as "This is a project description," the field in Small Grants Form (Environmental Grants) will record "This is a project description." Standard fields as recorded in form tables can be reconciled to the application/assessment/ acquittal/ administration tabs in an application depending on the form type.



Application Forms

Form	Status
Dynamic Forms	Submitted

Dynamic Forms

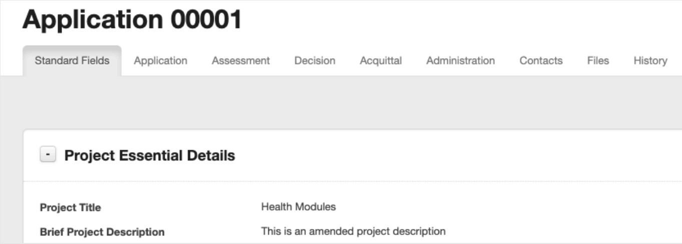
Page One

Section One

Project Title
Health Modules

Short project description
This is a project description

If an account administrator later enters the Applications tab of SmartyGrants and edits this applicant's standard field response to "This is an amended project description" (or the applicant answers the same standard field differently in another form), the field in Project Essential Details (Standard Fields) will record "This is an amended project description" (but the value in Small Grants Form (Environmental Grants) will remain 'This is a project description.") Data in standard field tables can be reconciled with the standard field tab of applications.



Application 00001

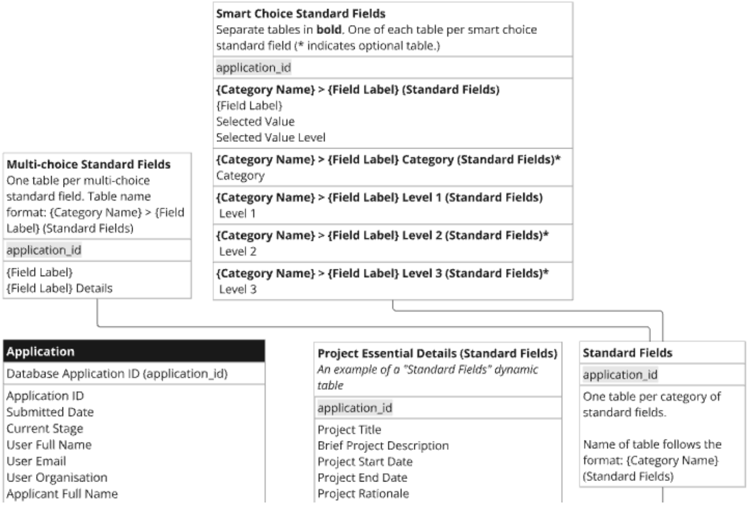
Project Essential Details

Project Title	Health Modules
Brief Project Description	This is an amended project description

Recall that fields in standard field tables capture the latest responses to standard fields, while fields in form tables capture "point in time" responses. This means the latest responses to standard fields will always be reportable by default, but you will need to turn on forms to report on those "point in time" responses.

In the above example, if Small Grants Form was not enabled for Analytics, the table Project Essential Details (Standard Fields) would still exist and will have a row for Application 00001 recording "This is amended project description." If the form was subsequently enabled for analytics, the table Small Grants Form (Environmental Grants) would be created, recording "This is a project description."

Single-response standard fields within a category will be stored in a main table (one table per category). If a category contains any multi-choice standard fields, responses to these will all be stored in separate, stand-alone tables (one table per multi-choice standard field, similar to form fields). If a category contains any smart choice fields, responses to these will also be stored in separate stand-alone tables (every smart-choice field generates a waterfall hierarchy of up to five tables. See the next section on Smart Choice Standard Fields for more information.)



Smart Choice Standard Fields
Separate tables in **bold**. One of each table per smart choice standard field (* indicates optional table.)
application_id
{Category Name} > {Field Label} (Standard Fields)
{Field Label}
Selected Value
Selected Value Level
{Category Name} > {Field Label} Category (Standard Fields)*
Category
{Category Name} > {Field Label} Level 1 (Standard Fields)
Level 1
{Category Name} > {Field Label} Level 2 (Standard Fields)*
Level 2
{Category Name} > {Field Label} Level 3 (Standard Fields)*
Level 3

Multi-choice Standard Fields
One table per multi-choice standard field. Table name format: {Category Name} > {Field Label} (Standard Fields)
application_id
{Field Label}
{Field Label} Details

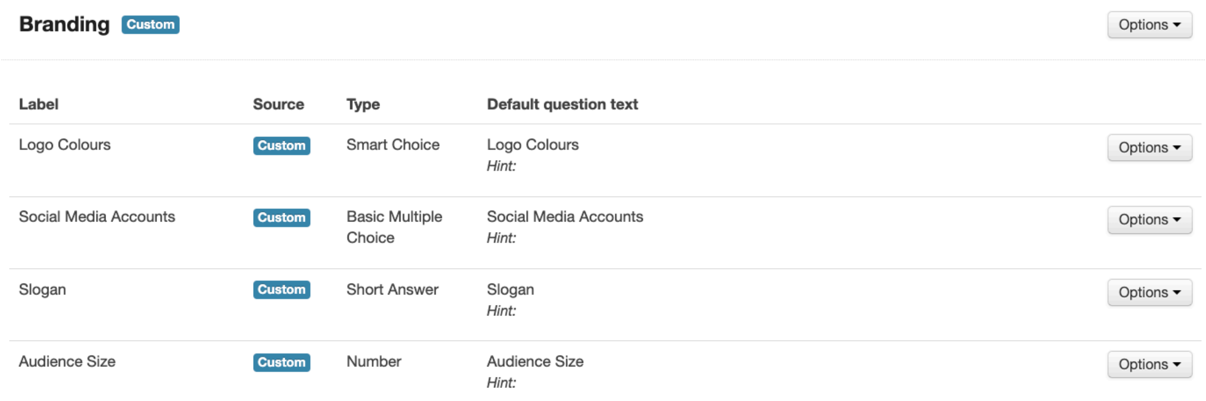
Application
Database Application ID (application_id)
Application ID
Submitted Date
Current Stage
User Full Name
User Email
User Organisation
Applicant Full Name

Project Essential Details (Standard Fields)
An example of a "Standard Fields" dynamic table
application_id
Project Title
Brief Project Description
Project Start Date
Project End Date
Project Rationale

Standard Fields
application_id
One table per category of standard fields.

Name of table follows the format: {Category Name} (Standard Fields)

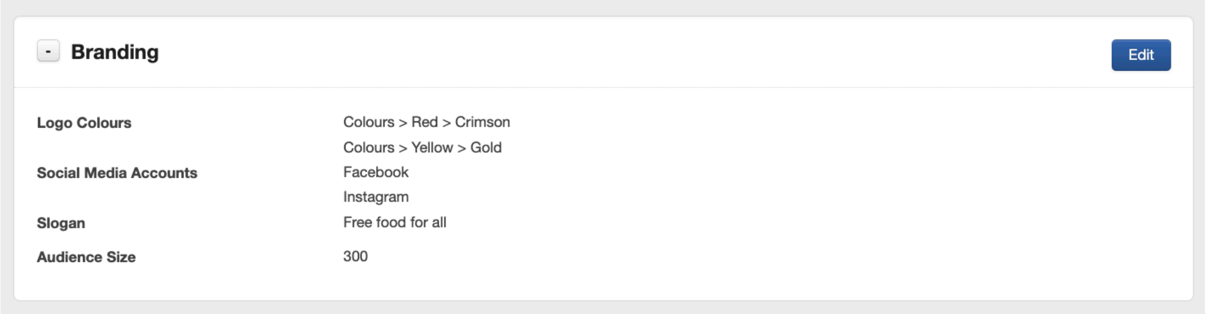
As an example, consider the following standard field category "Branding" which contains two single-response questions "Slogan" and "Audience Size", a multi-choice question "Social Media Accounts" and a smart choice question "Logo Colours."



Branding Custom

Label	Source	Type	Default question text
Logo Colours	Custom	Smart Choice	Logo Colours *Hint:*
Social Media Accounts	Custom	Basic Multiple Choice	Social Media Accounts *Hint:*
Slogan	Custom	Short Answer	Slogan *Hint:*
Audience Size	Custom	Number	Audience Size *Hint:*

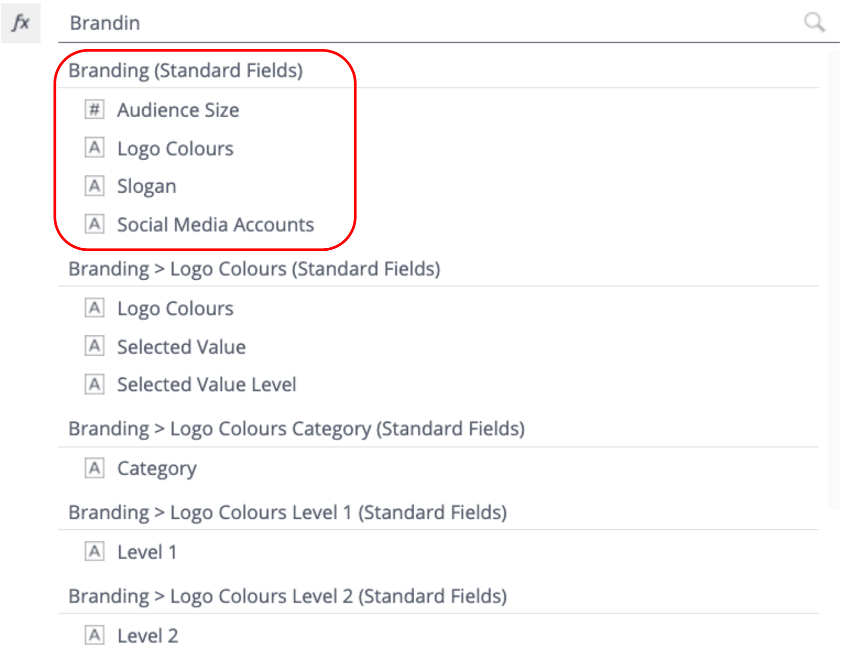
These standard fields have been filled out for the Application 0001.



Branding

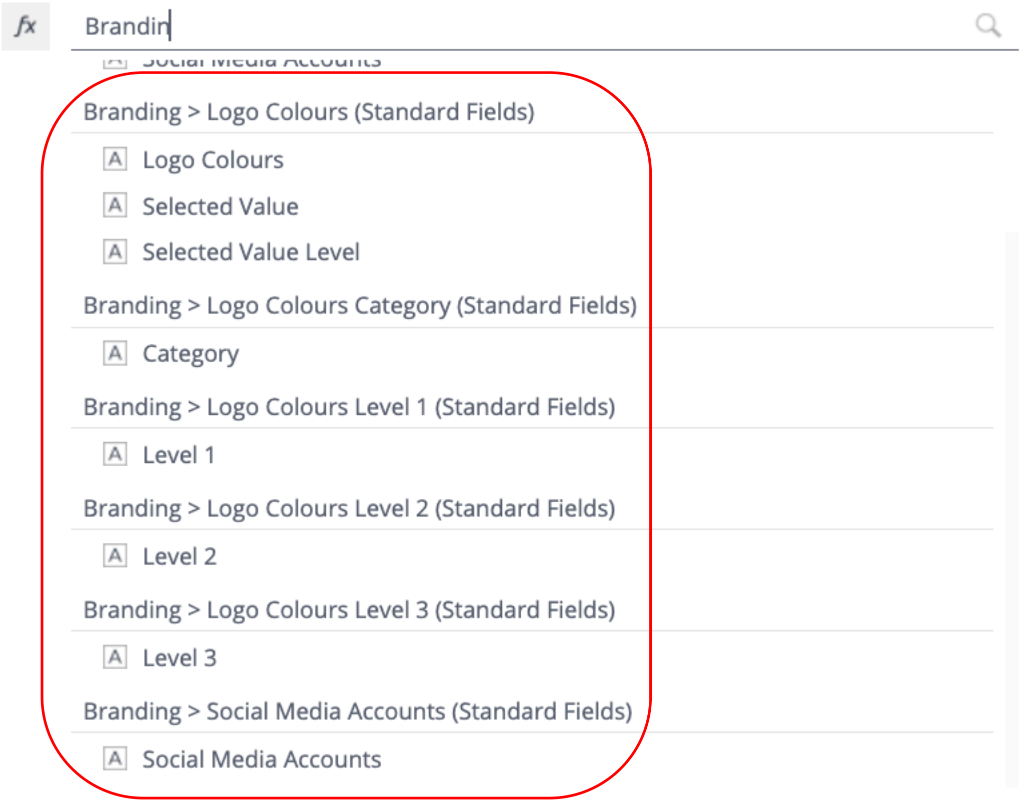
Logo Colours	Colours > Red > Crimson Colours > Yellow > Gold
Social Media Accounts	Facebook Instagram
Slogan	Free food for all
Audience Size	300

Responses to the two single-response questions "Audience Size" and "Slogan" are stored in the main table for the standard field category, Branding (Standard Fields). For every multi-choice and smart choice question, Branding (Standard Fields) also contains a corresponding field that concatenates choices responses (see below for an example.)



fx
Brandin
Branding (Standard Fields)
Audience Size
Logo Colours
Slogan
Social Media Accounts
Branding > Logo Colours (Standard Fields)
Logo Colours
Selected Value
Selected Value Level
Branding > Logo Colours Category (Standard Fields)
Category
Branding > Logo Colours Level 1 (Standard Fields)
Level 1
Branding > Logo Colours Level 2 (Standard Fields)
Level 2

The multi-choice and smart-choice questions "Logo Colours" and "Social Media Accounts" are stored in separate stand-alone tables.



Brandin

Social Media Accounts

Branding > Logo Colours (Standard Fields)

- A Logo Colours
- A Selected Value
- A Selected Value Level

Branding > Logo Colours Category (Standard Fields)

- A Category

Branding > Logo Colours Level 1 (Standard Fields)

- A Level 1

Branding > Logo Colours Level 2 (Standard Fields)

- A Level 2

Branding > Logo Colours Level 3 (Standard Fields)

- A Level 3

Branding > Social Media Accounts (Standard Fields)

- A Social Media Accounts

Consider the difference between using the concatenated “Social Media Accounts” field from the main Branding (Standard Fields) table, compared to the “Social Media Accounts” field from the stand-alone Branding > Social Media Accounts (Standard Fields) table which splits responses into separate rows.

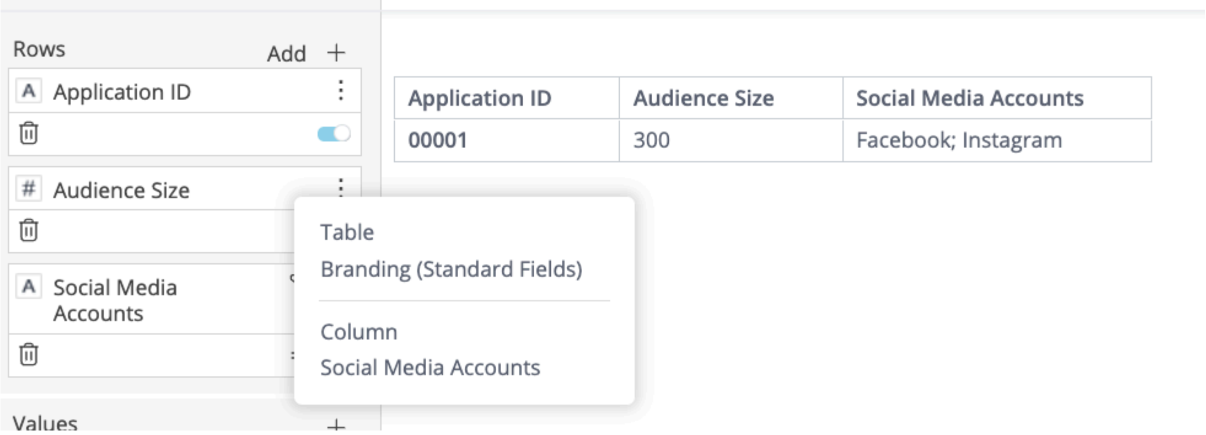
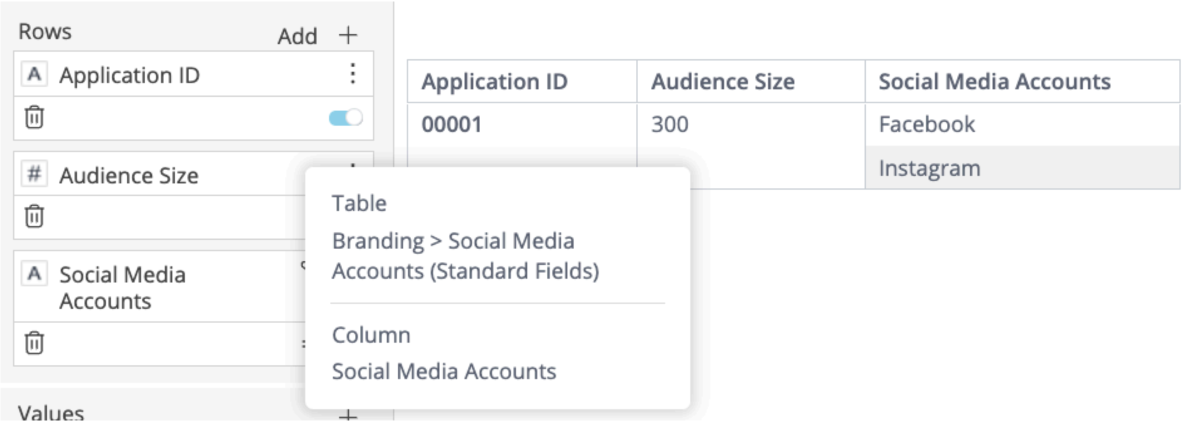


Table
Branding (Standard Fields)

Column
Social Media Accounts

Application ID	Audience Size	Social Media Accounts
00001	300	Facebook; Instagram



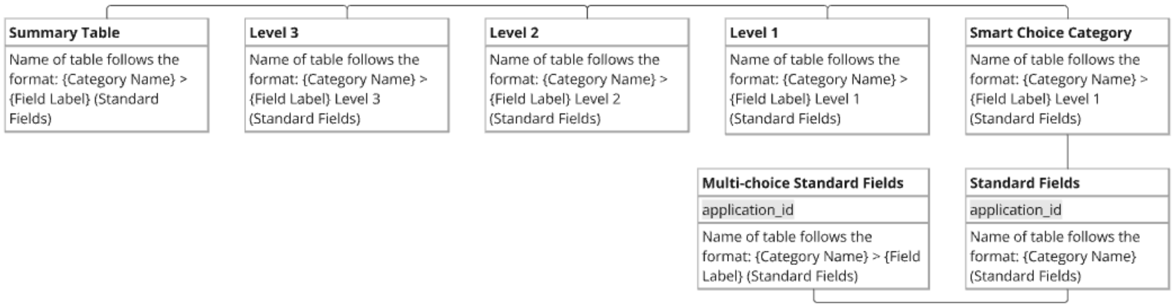
Application ID	Audience Size	Social Media Accounts
00001	300	Facebook
		Instagram

The next section goes into the various tables that are generated for a single smart-choice question.

18 Smart Choice Standard Fields

Responses to a single smart choice standard field are split across up to five separate tables, representing the hierarchical nature of smart choice responses. Please refer to the Help Hub if you are not familiar with smart choice lists and their layered hierarchy, which allow applicants to select a choice from a set list where each choice may be grouped under or belong to a parent choice.

If an instance utilises multiple smart choice standard fields, each standard field generates a separate hierarchy of up to five tables.



Summary Table
Name of table follows the format: {Category Name} > {Field Label} (Standard Fields)

Level 3
Name of table follows the format: {Category Name} > {Field Label} Level 3 (Standard Fields)

Level 2
Name of table follows the format: {Category Name} > {Field Label} Level 2 (Standard Fields)

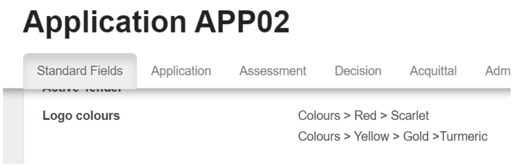
Level 1
Name of table follows the format: {Category Name} > {Field Label} Level 1 (Standard Fields)

Smart Choice Category
Name of table follows the format: {Category Name} > {Field Label} Level 1 (Standard Fields)

Multi-choice Standard Fields
application_id
Name of table follows the format: {Category Name} > {Field Label} (Standard Fields)

Standard Fields
application_id
Name of table follows the format: {Category Name} (Standard Fields)

The easiest way to understand how smart choice fields are stored in the data model is through an example. An example of a smart choice response could be to the question “Logo Colours”, which asks the user to select colours. Colours can be nested under umbrella colours.

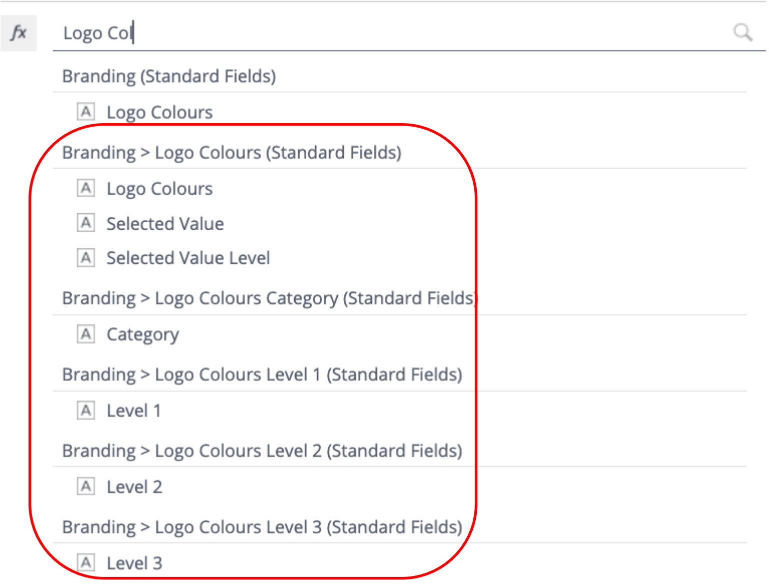


Logo colours — Colours > Red > Scarlet
Colours > Yellow > Gold >Turmeric

If an application selects “Turmeric”, this is a Level 3 choice with “Turmeric” nested under the hierarchy “Colours > Yellow > Gold > Turmeric”. In that hierarchy, “Turmeric” is nested under the Level 2 choice “Gold”, which is nested under the Level 1 choice “Yellow” which is nested under the non-selectable category “Colours”.

Every smart choice standard field dynamically generates a “summary” table that shows response values in their hierarchy (through a concatenated string), in addition to up to four “level” tables that split out that hierarchy.

- In the reporting tool, searching for “Logo Colours” shows five different tables besides the main Branding (Standard Fields) table: one for each level in the smart choice hierarchy (each with one field “Category”, “Level 1”, “Level 2”, “Level 3” respectively), and the summary table (with fields “Selected Value” and “Selected Value Level”).



Branding (Standard Fields)
- Logo Colours

Branding > Logo Colours (Standard Fields)
- Logo Colours
- Selected Value
- Selected Value Level

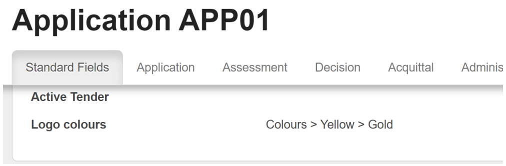
Branding > Logo Colours Category (Standard Fields)
- Category

Branding > Logo Colours Level 1 (Standard Fields)
- Level 1

Branding > Logo Colours Level 2 (Standard Fields)
- Level 2

Branding > Logo Colours Level 3 (Standard Fields)
- Level 3

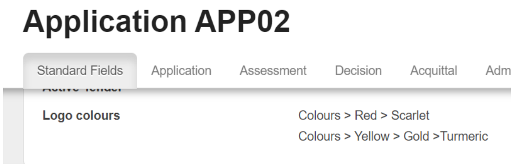
Consider two applications with the following standard field choices. APP02 selected the Level 3 choice “Turmeric” (also selecting a Level 2 choice “Scarlet”.) APP01 selected the Level 2 choice “Gold” and did not drill down to a level 3 selection.



Application APP01

Active Tender

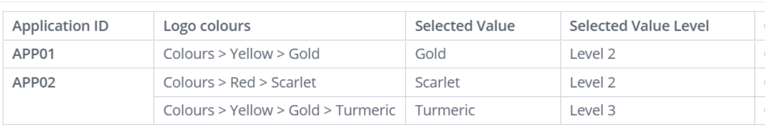
Logo colours: Colours > Yellow > Gold



Application APP02

Logo colours: Colours > Red > Scarlet
Colours > Yellow > Gold >Turmeric

Below is a widget using fields from the summary Branding > Logo Colours (Standard Fields) table. Summary smart choice tables follow the naming structure {Standard Field Category} > {Question Label} (Standard Fields).

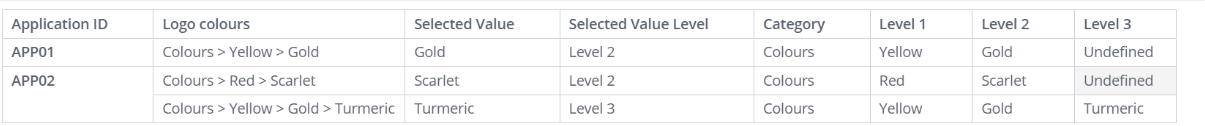


Application ID	Logo colours	Selected Value	Selected Value Level
APP01	Colours > Yellow > Gold	Gold	Level 2
APP02	Colours > Red > Scarlet	Scarlet	Level 2
	Colours > Yellow > Gold > Turmeric	Turmeric	Level 3

- “Logo Colours” (the name of this field is dynamic depending on the label of the smart choice question) shows values concatenated in their layered hierarchy
  - APP01: Colours > Yellow > Gold
  - APP02: Colours > Yellow > Gold > Turmeric
- “Selected Value” gives the lowest level value that the response drilled down to
  - APP01: Gold
  - APP02: Turmeric
- “Selected Value Level” gives the level that the response drilled down to
  - APP01: Level 2
  - APP02: Level 3

Fields from the level tables break down the layered selection. These tables have the naming structure {Standard Field Category} > {Question label} {Level} (Standard Fields). For example, APP02's selection of Colours > Yellow > Gold > Turmeric has:

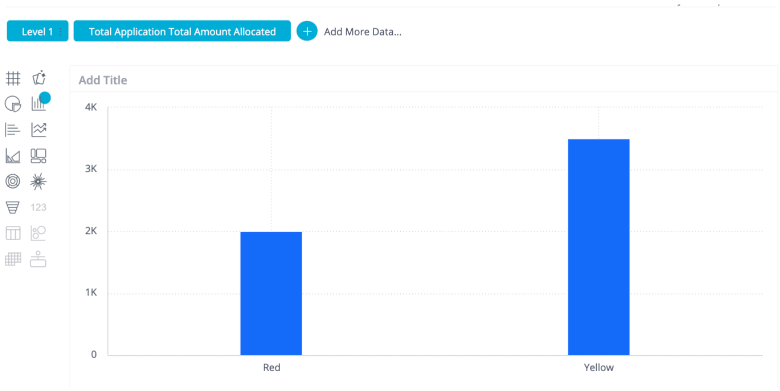
- "Turmeric" for the field "Level 3" from Branding > Logo Colours Level 3 (Standard Fields)
- "Gold" for the field "Level 2" from Branding > Logo Colours Level 2 (Standard Fields)
- "Yellow" for the field "Level 1" from Branding > Logo Colours Level 1 (Standard Fields)
- "Colours" for the field "Category" from Branding > Logo Colours Category (Standard Fields)



Application ID	Logo colours	Selected Value	Selected Value Level	Category	Level 1	Level 2	Level 3
APP01	Colours > Yellow > Gold	Gold	Level 2	Colours	Yellow	Gold	Undefined
APP02	Colours > Red > Scarlet	Scarlet	Level 2	Colours	Red	Scarlet	Undefined
	Colours > Yellow > Gold > Turmeric	Turmeric	Level 3	Colours	Yellow	Gold	Turmeric

Since APP01's selection of "Scarlet" was at Level 2 and did not drill into Level 3, it has "Undefined" for the field "Level 3" from Branding > Logo Colours Level 3 (Standard Fields)

Use the level tables to group data by different smart choice levels. The below chart uses "Level 1" from Branding > Logo Colours Level 1 (Standard Fields) to aggregate the amount allocated to applications that selected different Level 1 values.



Warning: Smart Choice Templates need to contain only non-overlapping hierarchies

SmartyGrants smart choice templates are designed for non-overlapping hierarchies, where a choice can only ever be nested under a single choice at the higher level.

An example of a non-overlapping hierarchy could be Food Group > Sub-group, where the choices for Food Group are "Animal-based" and "Plant-based". Sub-Group choices nested under "Animal-based" are "Meats and Poultry", "Dairy" and "Fish." Sub-Group choices nested under "Plant-based" are "Fruits" and "Vegetables." Under this hierarchy, selecting "Fruits" implies only one possible Food Group.

An example of an overlapping hierarchy could be Seniority > Gender, where the choices for Seniority are "Executive", "Middle Management" and "Junior." All three of these Seniority groups nest the Gender options of "Woman", "Man" or "Other". Under this overlapping hierarchy selecting "Man" implies multiple possible seniorities.

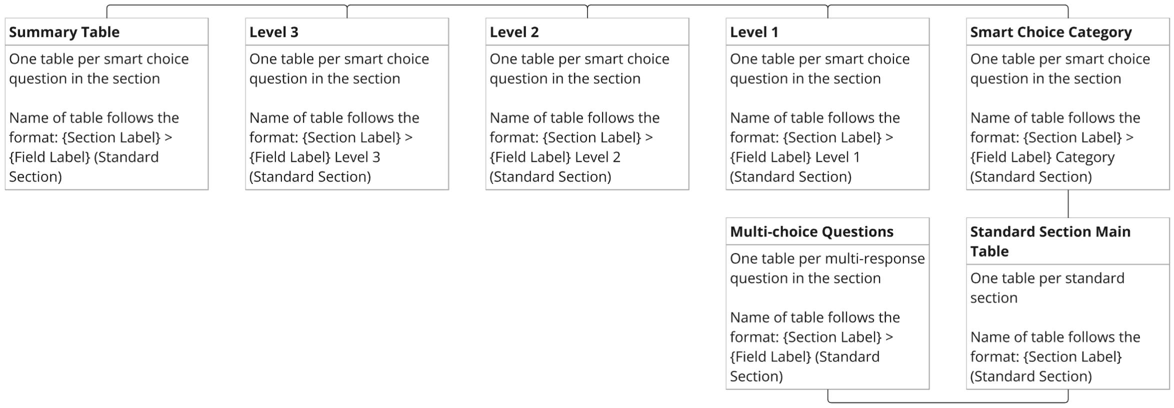
Inputting overlapping hierarchies into a smart choice template may lead to duplication (inflated figures) in the reporting tool. This duplication can be resolved through a calculated field in the widget builder, but orthodox hierarchies are encouraged to minimise unexpected results.

Note that in smart choice templates “Choice Category”, “Level 2” and “Level 3” columns are optional. If they are not utilised, the corresponding tables will not be created in the data model. For example, if “Logo Colours” used a Smart Choice List that only had the mandatory “Level 1” column filled out, in the data model only the summary and Level 1 table would be created.

19 Standard Sections

In the data model, each standard section is represented in its own dynamic table. If a standard section contains any multi-choice questions, these will all be stored in separate, stand-alone tables (one table per multi-choice question, similar to form fields). If a standard section contains any smart-choice questions, these will all be stored in separate, stand-alone tables (one waterfall hierarchy of tables per smart choice question, similar to smart-choice standard fields).

When building a widget, the easiest way to find a field from a standard section is to simply type the section label or the label of the question into the field selector. It is not necessary to understand how and why fields are organised to use them. If you are not familiar with standard sections and their grid structure, please see the Help Hub.



Summary Table

One table per smart choice question in the section

Name of table follows the format: {Section Label} > {Field Label} (Standard Section)

Level 3

One table per smart choice question in the section

Name of table follows the format: {Section Label} > {Field Label} Level 3 (Standard Section)

Level 2

One table per smart choice question in the section

Name of table follows the format: {Section Label} > {Field Label} Level 2 (Standard Section)

Level 1

One table per smart choice question in the section

Name of table follows the format: {Section Label} > {Field Label} Level 1 (Standard Section)

Smart Choice Category

One table per smart choice question in the section

Name of table follows the format: {Section Label} > {Field Label} Category (Standard Section)

Multi-choice Questions

One table per multi-response question in the section

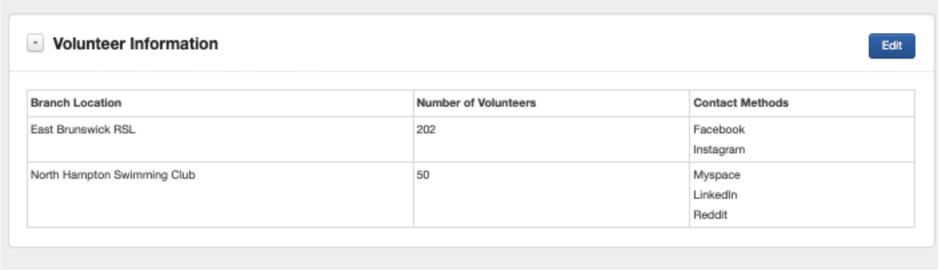
Name of table follows the format: {Section Label} > {Field Label} (Standard Section)

Standard Section Main Table

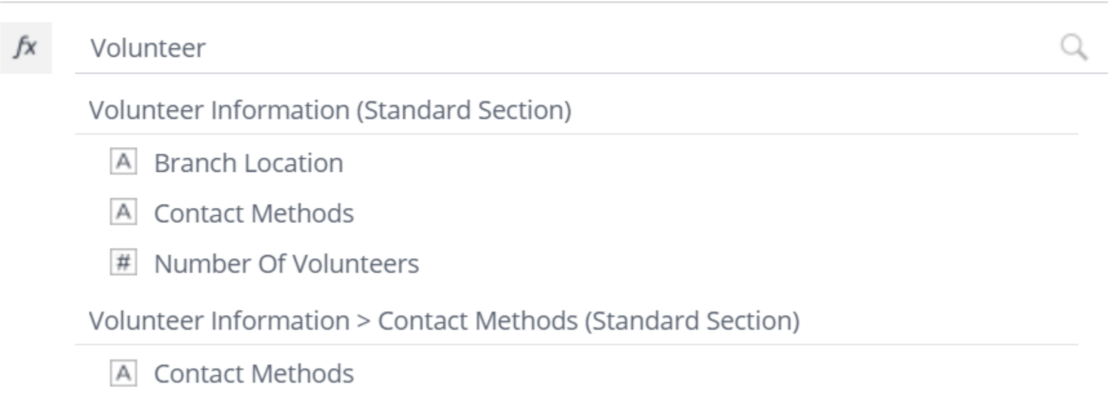
One table per standard section

Name of table follows the format: {Section Label} (Standard Section)

Consider the example of a standard section below “Volunteer Information,” which contains two single-response questions and one multi-choice question.



Branch Location	Number of Volunteers	Contact Methods
East Brunswick RSL	202	Facebook Instagram
North Hampton Swimming Club	50	Myspace LinkedIn Reddit



Volunteer Information (Standard Section)

- Branch Location
- Contact Methods
- Number Of Volunteers

Volunteer Information > Contact Methods (Standard Section)

- Contact Methods

The two single-response questions in the section ("Branch Location" and "Number of Volunteers") are stored in the section's main table called Volunteer Information (Standard Section).

- For every multi-choice and smart choice question in a standard section, the main standard section table contains a corresponding field that concatenates responses to a given question for a row. Using the "Contact Methods" field from Volunteer Information (Standard Section) would look like:

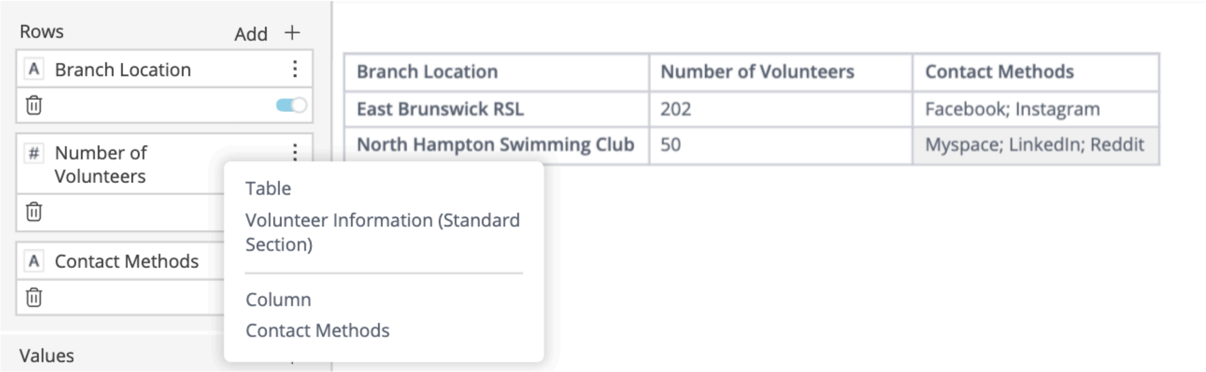


Table
Volunteer Information (Standard Section)

Column
Contact Methods

Branch Location	Number of Volunteers	Contact Methods
East Brunswick RSL	202	Facebook; Instagram
North Hampton Swimming Club	50	Myspace; LinkedIn; Reddit

The multi-choice question "Contact Methods" generates a separate table Volunteer Information > Contact Methods (Standard Section). Using the "Contact Methods" table from this table shows that this field separates choices out, rather than concatenating.

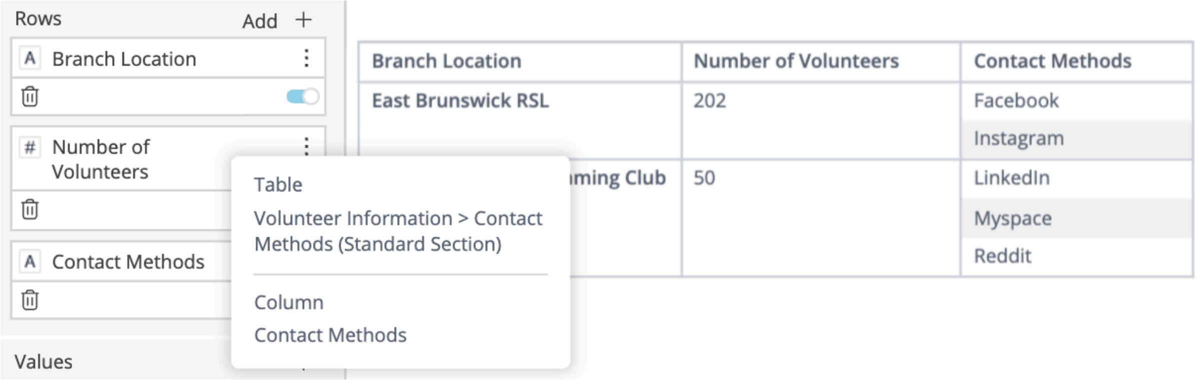


Table
Volunteer Information > Contact Methods (Standard Section)

Column
Contact Methods

Branch Location	Number of Volunteers	Contact Methods
East Brunswick RSL	202	Facebook
		Instagram
ming Club	50	LinkedIn
		Myspace
		Reddit

Similar [to standard fields](), standard section tables capture the latest responses to standard sections, while fields in form tables capture "point in time" responses ("point in time" responses to standard sections in forms are stored as [repeatable sections]().)

Standard sections share several other reporting characteristics as standard fields:

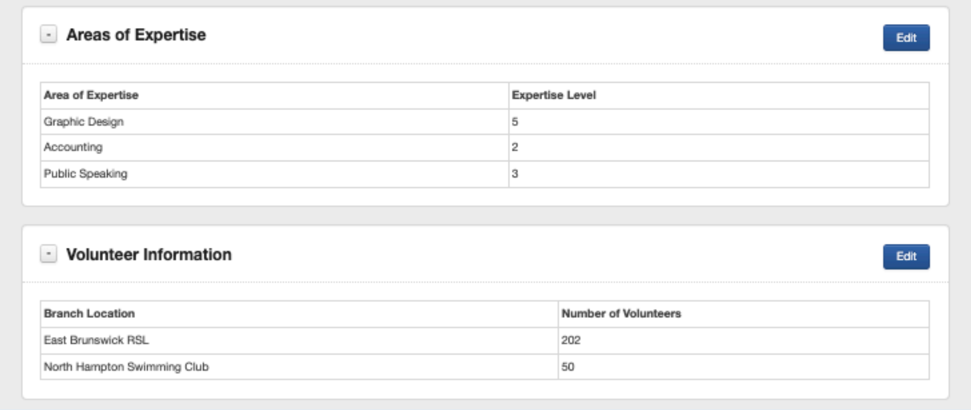
- Standard sections do not need to be turned on for reporting and are available in Analytics by default.
- Archived standard sections are not included in reporting.

Warning: be careful when combining fields from multiple sections (whether repeatable or standard sections) in a single widget

It is not recommended to combine fields from multiple sections in a single widget. This includes combining fields from multiple standard sections in a single widget and combining fields from a standard section and a repeatable section in a single widget.

If an application has filled out multiple rows in one or all sections, the join on application_id creates all possible permutations of values between fields used in the widget. This join can lead to misleading/nonsensical joining and duplication of values.

For example, in a table widget there would be a duplication of values across rows of the table (as well as potentially a misrepresentation that values from different sections are related to each other.) Consider the two example standard sections “Areas of Expertise” and “Volunteer Information.”



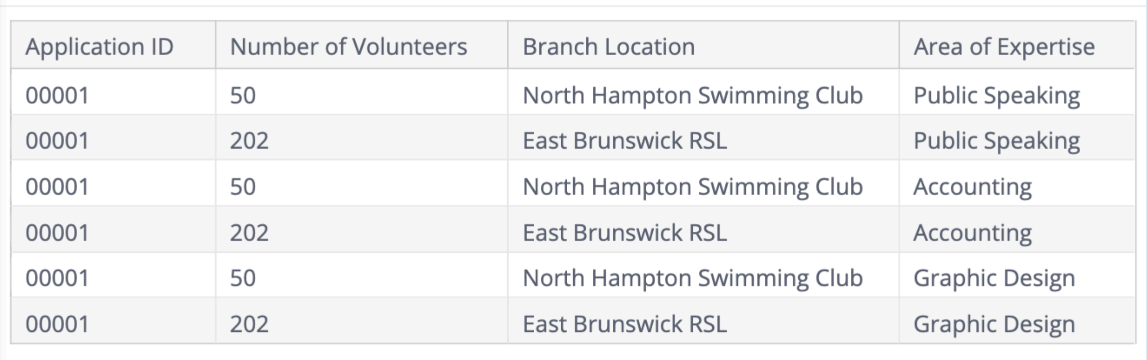
Areas of Expertise

Area of Expertise	Expertise Level
Graphic Design	5
Accounting	2
Public Speaking	3

Volunteer Information

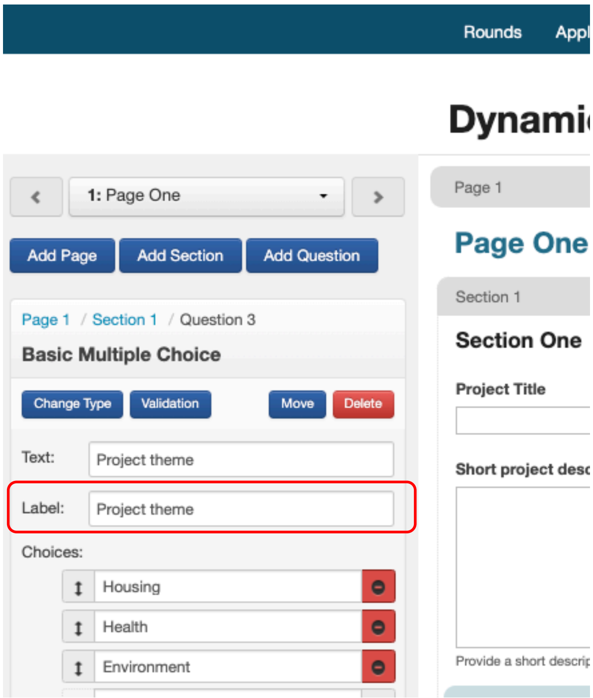
Branch Location	Number of Volunteers
East Brunswick RSL	202
North Hampton Swimming Club	50

When fields from both standard sections are used in a single widget, duplication of values occurs. The result may also suggest that the number of volunteers is related to an area of expertise, which may not be true.



Application ID	Number of Volunteers	Branch Location	Area of Expertise
00001	50	North Hampton Swimming Club	Public Speaking
00001	202	East Brunswick RSL	Public Speaking
00001	50	North Hampton Swimming Club	Accounting
00001	202	East Brunswick RSL	Accounting
00001	50	North Hampton Swimming Club	Graphic Design
00001	202	East Brunswick RSL	Graphic Design

20 Dynamic Tables: Troubleshooting/FAQs



Text: Project theme

Label: Project theme

Choices:
- Housing
- Health
- Environment

Form FAQs

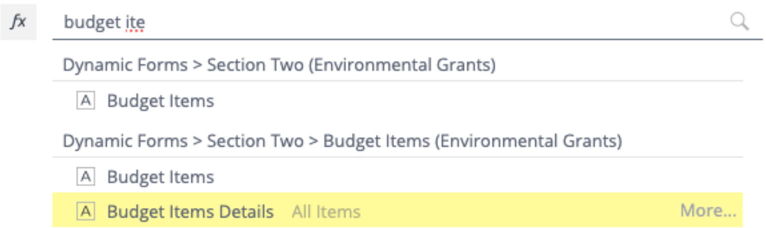
When building a widget, I can't find the field for a question in a form. What should I do?

First, make sure analytics is enabled for the form. If analytics is enabled, open the form editor to check the *label* of the question or the *label* of the section that the question is from if the question is from a repeatable question. Labels can be distinct from question text/ section name and are used to generate the names of dynamic tables.

For fields from forms to be located, these labels should be intuitive to allow for easy searching. Once you have set your labels, you can simply type them into the field selector when building a widget to narrow the search.

Why are there fields which end in "Details"?

If a multi-choice question allows users to select "Other" and input a free text response, the free-text response is stored in a field that is the question label followed by "Details".



budget ite

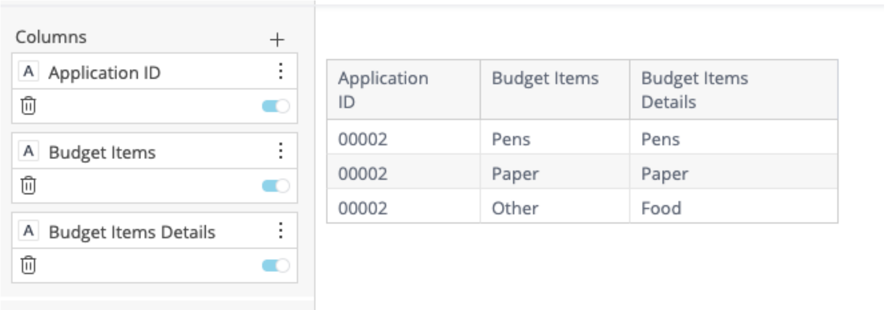
Dynamic Forms > Section Two (Environmental Grants)

- Budget Items

Dynamic Forms > Section Two > Budget Items (Environmental Grants)

- Budget Items
- Budget Items Details All Items

For responses where an applicant did not select "other", the choice is repeated in the "Details" column.

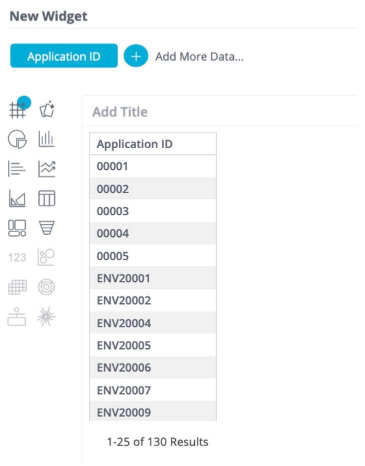


Application ID	Budget Items	Budget Items Details
00002	Pens	Pens
00002	Paper	Paper
00002	Other	Food

Why do some applications disappear from the widget when I use a field from a form table?

When using a field from a form table, the widget will automatically filter down to applications that have submitted that form.

On the left using “Application ID” from the Application table would generate a list of all Application IDs. On the right, a field from an assessment form is added to the widget, reducing the number of Application IDs appearing in the widget to those which had the assessment form submitted.



Application ID
00001
00002
00003
00004
00005
ENV20001
ENV20002
ENV20004
ENV20005
ENV20006
ENV20007
ENV20009

1-25 of 130 Results

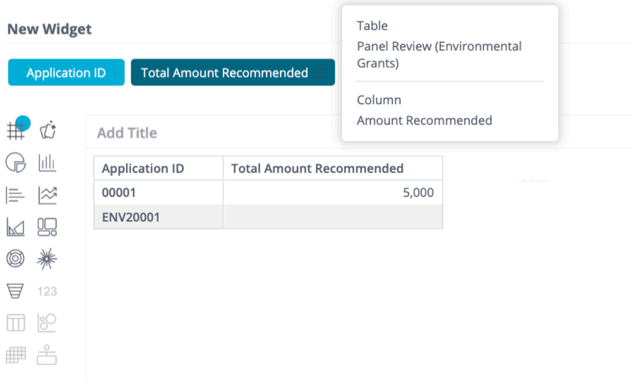


Table
Panel Review (Environmental Grants)

Column
Amount Recommended

Application ID	Total Amount Recommended
00001	5,000
ENV20001	

This filtering of Application IDs occurs because form tables only contain data for applications that have submitted forms (an inner join between form tables and other tables therefore excludes some Application IDs.)

Why do no results appear when I try to combine fields from multiple forms in a single widget?

It is not possible to combine fields from multiple forms in a single widget (besides fields within the static Form Response table.) This is because form tables contain a hidden response_id key that is unique for every response to a form, making joins between form tables impossible. However, you can include fields from different forms in separate widgets and combine them within a single dashboard.

Standard Field and Standard Section FAQs

I combined fields from two standard sections in a widget and the widget does not look as expected (too many rows, duplicated values etc.) What happened?

It is not recommended to combine fields from multiple standard sections in a single widget. See here for an explanation why.

I combined fields from a standard section and repeatable section (from a form) in a widget and the widget does not look as expected (too many rows, duplicated values etc.) What happened?

It is not recommended to combine fields from a standard section and a repeatable section in a single widget. See here for an explanation why.

Why do some applications disappear from the widget when I use a field from a multi-choice or smart-choice table (any table branching off a main standard section or main standard field table?)

When using a field from a multi-choice or smart-choice table, the widget will automatically filter down to applications that have answered the question.

The image below uses “Application ID” from the Application table and the field “Branch Location” from the main Volunteer Information standard section table. This generates a list of all Application IDs. Applications that do not have the Volunteer Information standard section filled out have “N/A” for “Branch Location.”

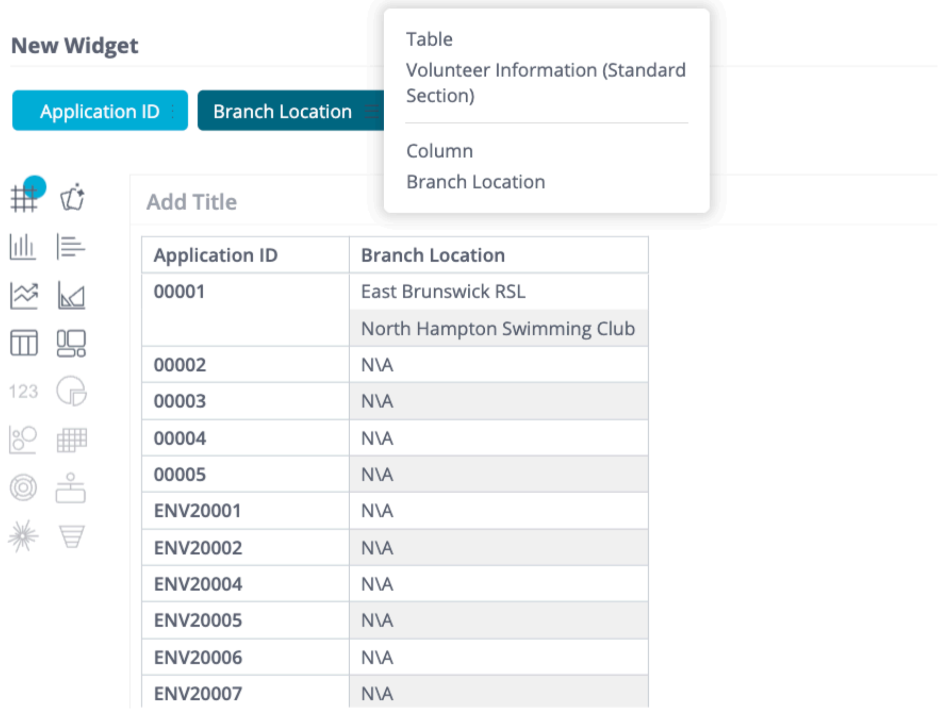


Table
Volunteer Information (Standard Section)

Column
Branch Location

Application ID	Branch Location
00001	East Brunswick RSL North Hampton Swimming Club
00002	N\A
00003	N\A
00004	N\A
00005	N\A
ENV20001	N\A
ENV20002	N\A
ENV20004	N\A
ENV20005	N\A
ENV20006	N\A
ENV20007	N\A

On the right, the multi-choice field “Contact Methods” from the Volunteer Information standard section (stored in a standalone table) is added to the widget, reducing the number of Application IDs appearing in the widget to those which have answered “Contact Methods”.

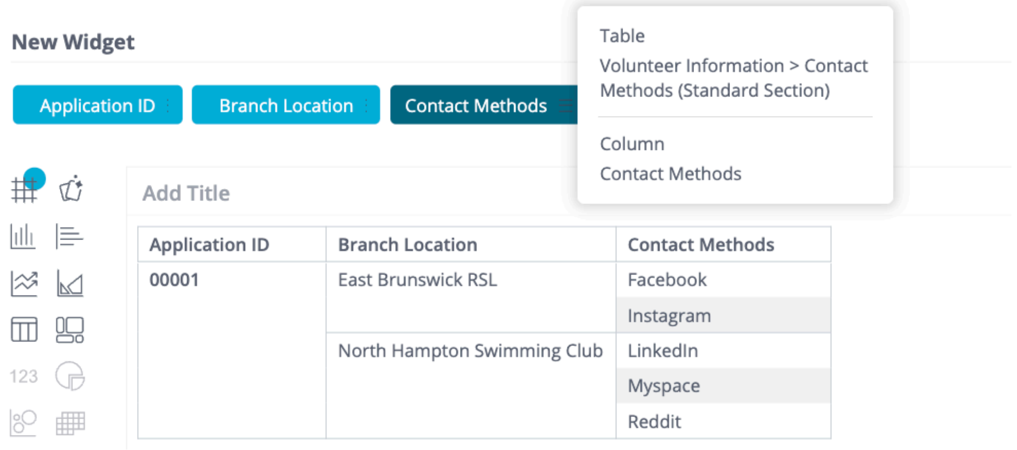


Table
Volunteer Information > Contact Methods (Standard Section)

Column
Contact Methods

Application ID	Branch Location	Contact Methods
00001	East Brunswick RSL	Facebook Instagram
	North Hampton Swimming Club	LinkedIn Myspace Reddit

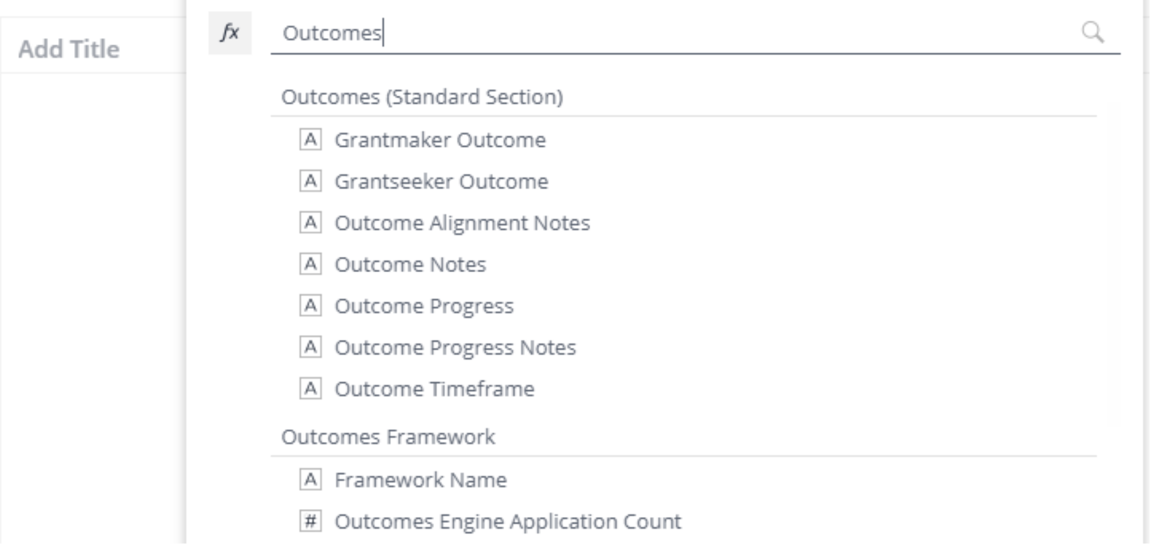
The filtering out of application IDs occurs because the multi-choice/smart-choice tables only contain data for applications that have answered the associated multi-choice/smart-choice question (an inner join between these tables and other tables therefore excludes some Application IDs.)

21 Outcomes Engine Reporting

Tables related to the Outcomes Engine will only appear for instances which have the Outcomes Engine turned on. If you are not familiar with the Outcomes Engine and Outcomes Engine default standard sections, please see the Help Hub.

Outcomes Engine Standard Sections

Outcomes Engine default standard sections (e.g. Grantmaker Metrics, Activities) are dynamically created in the data model as any other standard section, and can simply be searched for in the field selector when creating a widget.



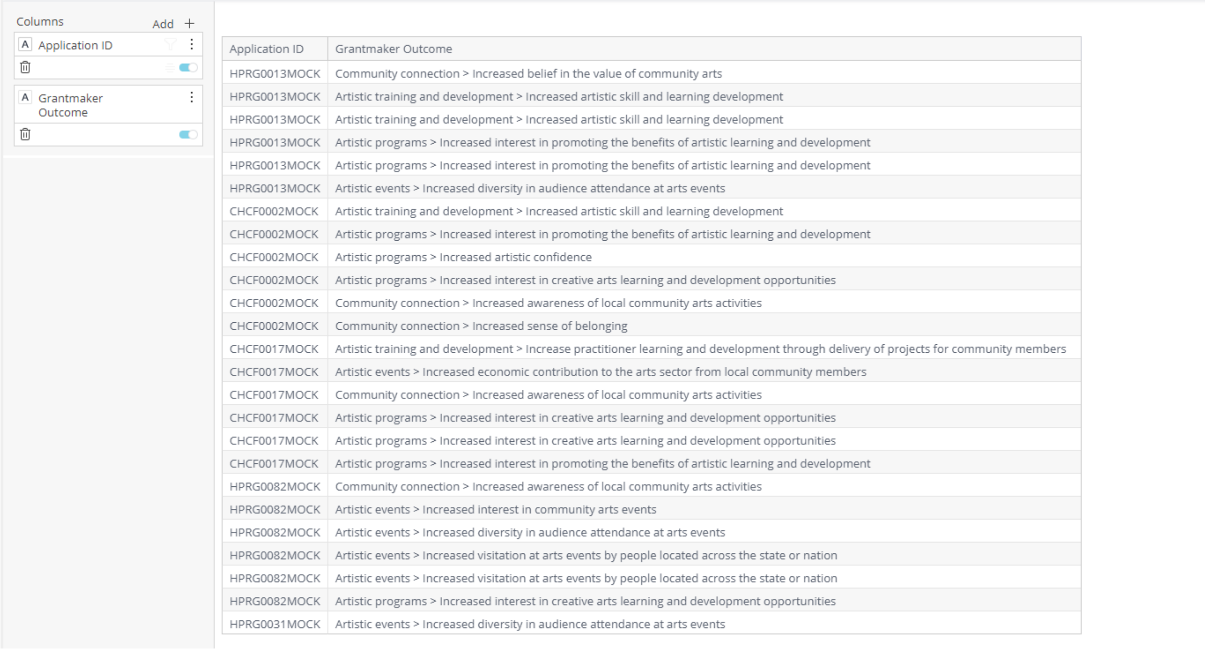
Outcomes (Standard Section)

- A Grantmaker Outcome
- A Grantseeker Outcome
- A Outcome Alignment Notes
- A Outcome Notes
- A Outcome Progress
- A Outcome Progress Notes
- A Outcome Timeframe

Outcomes Framework

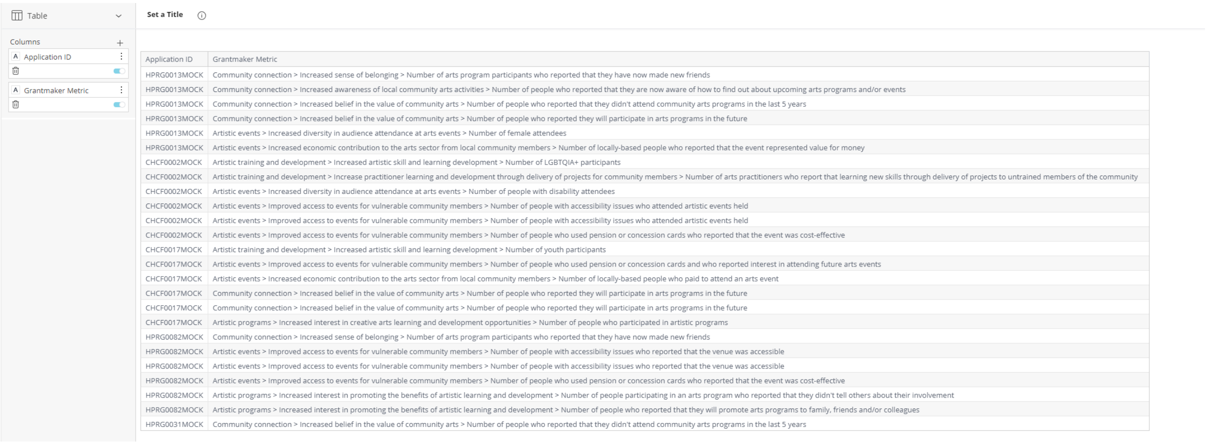
- A Framework Name
- # Outcomes Engine Application Count

Grantmaker domain, outcome and metric selections stored in tables ending with "(Standard Section)" will be in a concatenated form following the structure domain > outcome > metric. In the example below, a value from "Grantmaker Outcome" in Outcomes (Standard Section) is "Community Connection > Increased belief in the value of community arts", where "Community Connection" is a domain housing the outcome "Increased belief in the value of community arts."



Application ID	Grantmaker Outcome
HPRG0013MOCK	Community connection > Increased belief in the value of community arts
HPRG0013MOCK	Artistic training and development > Increased artistic skill and learning development
HPRG0013MOCK	Artistic training and development > Increased artistic skill and learning development
HPRG0013MOCK	Artistic programs > Increased interest in promoting the benefits of artistic learning and development
HPRG0013MOCK	Artistic programs > Increased interest in promoting the benefits of artistic learning and development
HPRG0013MOCK	Artistic events > Increased diversity in audience attendance at arts events
CHCF0002MOCK	Artistic training and development > Increased artistic skill and learning development
CHCF0002MOCK	Artistic programs > Increased interest in promoting the benefits of artistic learning and development
CHCF0002MOCK	Artistic programs > Increased artistic confidence
CHCF0002MOCK	Artistic programs > Increased interest in creative arts learning and development opportunities
CHCF0002MOCK	Community connection > Increased awareness of local community arts activities
CHCF0002MOCK	Community connection > Increased sense of belonging
CHCF0017MOCK	Artistic training and development > Increase practitioner learning and development through delivery of projects for community members
CHCF0017MOCK	Artistic events > Increased economic contribution to the arts sector from local community members
CHCF0017MOCK	Community connection > Increased awareness of local community arts activities
CHCF0017MOCK	Artistic programs > Increased interest in creative arts learning and development opportunities
CHCF0017MOCK	Artistic programs > Increased interest in creative arts learning and development opportunities
CHCF0017MOCK	Artistic programs > Increased interest in promoting the benefits of artistic learning and development
HPRG0082MOCK	Community connection > Increased awareness of local community arts activities
HPRG0082MOCK	Artistic events > Increased interest in community arts events
HPRG0082MOCK	Artistic events > Increased diversity in audience attendance at arts events
HPRG0082MOCK	Artistic events > Increased visitation at arts events by people located across the state or nation
HPRG0082MOCK	Artistic events > Increased visitation at arts events by people located across the state or nation
HPRG0082MOCK	Artistic programs > Increased interest in creative arts learning and development opportunities
HPRG0031MOCK	Artistic events > Increased diversity in audience attendance at arts events

In the example below, a value from "Grantmaker Metric" in Grantmaker Metrics (Standard Section) is "Community connection > Increased sense of belonging > Number of arts program participants who reported that they have now made new friends." "Community Connection" is a domain housing the outcome "Increased sense of belonging", which in turn houses the metric "Number of arts program participants who reported that they have now made new friends."

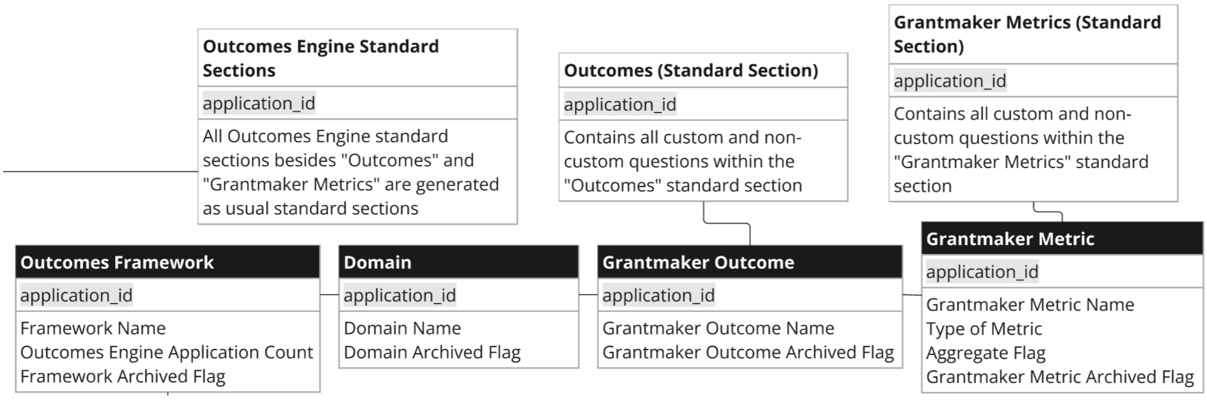


Application ID	Grantmaker Metric
HPRG0013MOCK	Community connection > Increased sense of belonging > Number of arts program participants who reported that they have now made new friends
HPRG0013MOCK	Community connection > Increased awareness of local community arts activities > Number of people who reported that they are now aware of how to find out about upcoming arts programs and/or events
HPRG0013MOCK	Community connection > Increased belief in the value of community arts > Number of people who reported that they didn't attend community arts programs in the last 5 years
HPRG0013MOCK	Community connection > Increased belief in the value of community arts > Number of people who reported they will participate in arts programs in the future
HPRG0013MOCK	Artistic events > Increased diversity in audience attendance at arts events > Number of female attendees
HPRG0013MOCK	Artistic events > Increased economic contribution to the arts sector from local community members > Number of locally-based people who reported that the event represented value for money
CHCF0002MOCK	Artistic training and development > Increased artistic skill and learning development > Number of LGBTQIA+ participants
CHCF0002MOCK	Artistic training and development > Increase practitioner learning and development through delivery of projects for community members > Number of arts practitioners who report that learning new skills through delivery of projects to untrained members of the community
CHCF0002MOCK	Artistic events > Increased diversity in audience attendance at arts events > Number of people with disability attendees
CHCF0002MOCK	Artistic events > Improved access to events for vulnerable community members > Number of people with accessibility issues who attended artistic events held
CHCF0002MOCK	Artistic events > Improved access to events for vulnerable community members > Number of people with accessibility issues who attended artistic events held
CHCF0002MOCK	Artistic events > Improved access to events for vulnerable community members > Number of people who used pension or concession cards who reported that the event was cost-effective
CHCF0017MOCK	Artistic training and development > Increased artistic skill and learning development > Number of youth participants
CHCF0017MOCK	Artistic events > Improved access to events for vulnerable community members > Number of people who used pension or concession cards and who reported interest in attending future arts events
CHCF0017MOCK	Artistic events > Increased economic contribution to the arts sector from local community members > Number of locally-based people who paid to attend an arts event
CHCF0017MOCK	Community connection > Increased belief in the value of community arts > Number of people who reported they will participate in arts programs in the future
CHCF0017MOCK	Community connection > Increased belief in the value of community arts > Number of people who reported they will participate in arts programs in the future
CHCF0017MOCK	Artistic programs > Increased interest in creative arts learning and development opportunities > Number of people who participated in artistic programs
HPRG0082MOCK	Community connection > Increased sense of belonging > Number of arts program participants who reported that they have now made new friends
HPRG0082MOCK	Artistic events > Improved access to events for vulnerable community members > Number of people with accessibility issues who reported that the venue was accessible
HPRG0082MOCK	Artistic events > Improved access to events for vulnerable community members > Number of people with accessibility issues who reported that the venue was accessible
HPRG0082MOCK	Artistic events > Improved access to events for vulnerable community members > Number of people who used pension or concession cards who reported that the event was cost-effective
HPRG0082MOCK	Artistic programs > Increased interest in promoting the benefits of artistic learning and development > Number of people participating in an arts program who reported that they didn't tell others about their involvement
HPRG0082MOCK	Artistic programs > Increased interest in promoting the benefits of artistic learning and development > Number of people who reported that they will promote arts programs to family, friends and/or colleagues
HPRG0031MOCK	Community connection > Increased belief in the value of community arts > Number of people who reported that they didn't attend community arts programs in the last 5 years

Static Outcomes Tables

If you would like to group data by domain/outcome/metrics, or access domain/outcome/metric data in a non-concatenated form, the data model contains four static tables for these purposes:

- Outcomes Framework (containing the field "Framework Name")
- Domain (containing the field "Domain Name")
- Grantmaker Outcome (containing the field "Grantmaker Outcome Name")
- Grantmaker Metric (containing the field "Grantmaker Metric Name")



Outcomes Engine Standard Sections
application_id
All Outcomes Engine standard sections besides "Outcomes" and "Grantmaker Metrics" are generated as usual standard sections

Outcomes (Standard Section)
application_id
Contains all custom and non-custom questions within the "Outcomes" standard section

Grantmaker Metrics (Standard Section)
application_id
Contains all custom and non-custom questions within the "Grantmaker Metrics" standard section

Outcomes Framework
application_id
Framework Name
Outcomes Engine Application Count
Framework Archived Flag

Domain
application_id
Domain Name
Domain Archived Flag

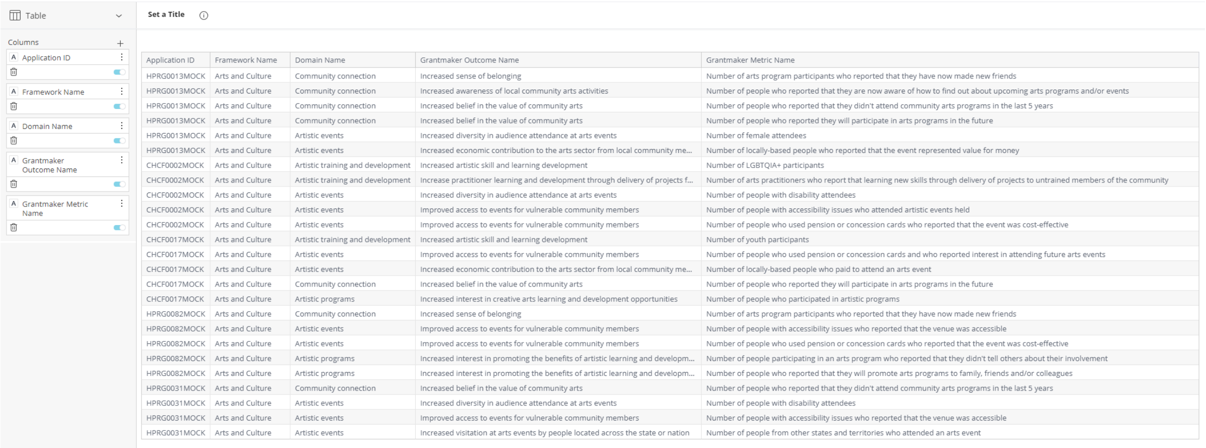
Grantmaker Outcome
application_id
Grantmaker Outcome Name
Grantmaker Outcome Archived Flag

Grantmaker Metric
application_id
Grantmaker Metric Name
Type of Metric
Aggregate Flag
Grantmaker Metric Archived Flag

Creating a widget with these table shows the difference between the storage of framework/domain/outcome/metric selections in these tables, and corresponding fields in standard section tables.

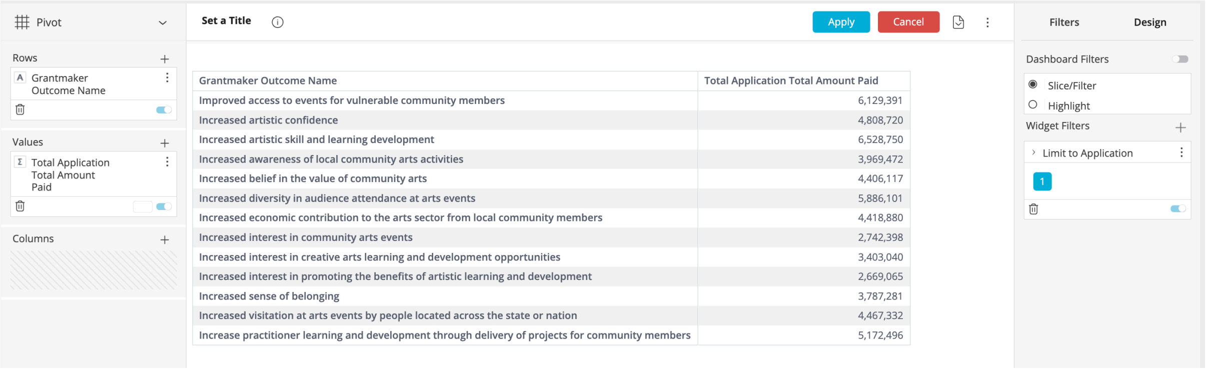
- Grantmaker Metric (Standard Section) stores a metric as "Community connection > Increased sense of belonging > Number of arts program participants who reported that they have now made new friends."
- The static tables split the domain, outcome and metric components out into "Community connection" in Domain Name, "Increased sense of Belonging" in Grantmaker Outcome Name, and "Number of arts program participants who reported that they have now made new friends" in Grantmaker Metric Name respectively.

If an outcomes framework does not make use of domains (which are optional), outcomes will be attached to a domain "Undefined." Grantseeker Outcomes in the "Outcome OE" standard section not linked to a grantmaker outcome are linked to a domain "Not selected" and a grantmaker outcome "Not selected."



Application ID	Framework Name	Domain Name	Grantmaker Outcome Name	Grantmaker Metric Name
HPRG0013MOCK	Arts and Culture	Community connection	Increased sense of belonging	Number of arts program participants who reported that they have now made new friends
HPRG0013MOCK	Arts and Culture	Community connection	Increased awareness of local community arts activities	Number of people who reported that they are now aware of how to find out about upcoming arts programs and/or events
HPRG0013MOCK	Arts and Culture	Community connection	Increased belief in the value of community arts	Number of people who reported that they didn't attend community arts programs in the last 5 years
HPRG0013MOCK	Arts and Culture	Community connection	Increased belief in the value of community arts	Number of people who reported they will participate in arts programs in the future
HPRG0013MOCK	Arts and Culture	Artistic events	Increased diversity in audience attendance at arts events	Number of female attendees
HPRG0013MOCK	Arts and Culture	Artistic events	Increased economic contribution to the arts sector from local community me...	Number of locally-based people who reported that the event represented value for money
CHCF0002MOCK	Arts and Culture	Artistic training and development	Increased artistic skill and learning development	Number of LGBTQIA+ participants
CHCF0002MOCK	Arts and Culture	Artistic training and development	Increase practitioner learning and development through delivery of projects f...	Number of arts practitioners who report that learning new skills through delivery of projects to untrained members of the community
CHCF0002MOCK	Arts and Culture	Artistic events	Increased diversity in audience attendance at arts events	Number of people with disability attendees
CHCF0002MOCK	Arts and Culture	Artistic events	Improved access to events for vulnerable community members	Number of people with accessibility issues who attended artistic events held
CHCF0002MOCK	Arts and Culture	Artistic events	Improved access to events for vulnerable community members	Number of people who used pension or concession cards who reported that the event was cost-effective
CHCF0017MOCK	Arts and Culture	Artistic training and development	Increased artistic skill and learning development	Number of youth participants
CHCF0017MOCK	Arts and Culture	Artistic events	Improved access to events for vulnerable community members	Number of people who used pension or concession cards and who reported interest in attending future arts events
CHCF0017MOCK	Arts and Culture	Artistic events	Increased economic contribution to the arts sector from local community me...	Number of locally-based people who paid to attend an arts event
CHCF0017MOCK	Arts and Culture	Community connection	Increased belief in the value of community arts	Number of people who reported they will participate in arts programs in the future
CHCF0017MOCK	Arts and Culture	Artistic programs	Increased interest in creative arts learning and development opportunities	Number of people who participated in artistic programs
HPRG0082MOCK	Arts and Culture	Community connection	Increased sense of belonging	Number of arts program participants who reported that they have now made new friends
HPRG0082MOCK	Arts and Culture	Artistic events	Improved access to events for vulnerable community members	Number of people with accessibility issues who reported that the venue was accessible
HPRG0082MOCK	Arts and Culture	Artistic events	Improved access to events for vulnerable community members	Number of people who used pension or concession cards who reported that the event was cost-effective
HPRG0082MOCK	Arts and Culture	Artistic programs	Increased interest in promoting the benefits of artistic learning and developm...	Number of people participating in an arts program who reported that they didn't tell others about their involvement
HPRG0082MOCK	Arts and Culture	Artistic programs	Increased interest in promoting the benefits of artistic learning and developm...	Number of people who reported that they will promote arts programs to family, friends and/or colleagues
HPRG0031MOCK	Arts and Culture	Community connection	Increased belief in the value of community arts	Number of people who reported that they didn't attend community arts programs in the last 5 years
HPRG0031MOCK	Arts and Culture	Artistic events	Increased diversity in audience attendance at arts events	Number of people with disability attendees
HPRG0031MOCK	Arts and Culture	Artistic events	Improved access to events for vulnerable community members	Number of people with accessibility issues who reported that the venue was accessible
HPRG0031MOCK	Arts and Culture	Artistic events	Increased visitation at arts events by people located across the state or nation	Number of people from other states and territories who attended an arts event

Use the static tables to group data, such as financial data. The widget below uses "Grantmaker Outcome Name" from the Grantmaker Outcomes table to find the total amount paid to applications that selected a given grantmaker outcome.



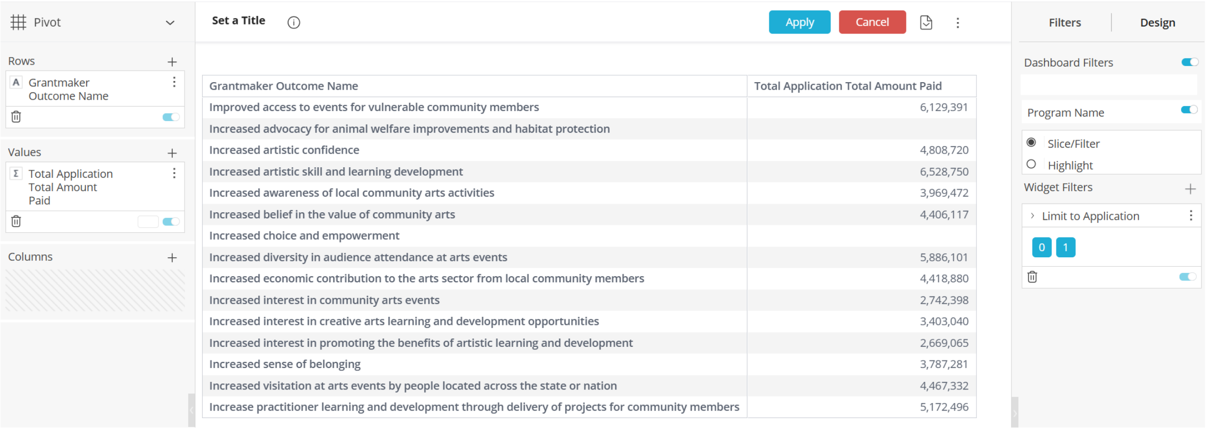
Grantmaker Outcome Name	Total Application Total Amount Paid
Improved access to events for vulnerable community members	6,129,391
Increased artistic confidence	4,808,720
Increased artistic skill and learning development	6,528,750
Increased awareness of local community arts activities	3,969,472
Increased belief in the value of community arts	4,406,117
Increased diversity in audience attendance at arts events	5,886,101
Increased economic contribution to the arts sector from local community members	4,418,880
Increased interest in community arts events	2,742,398
Increased interest in creative arts learning and development opportunities	3,403,040
Increased interest in promoting the benefits of artistic learning and development	2,669,065
Increased sense of belonging	3,787,281
Increased visitation at arts events by people located across the state or nation	4,467,332
Increase practitioner learning and development through delivery of projects for community members	5,172,496

Besides domain/outcome/metric selections, the static tables contain several other fields to facilitate outcomes-related analysis. Below contains a guide on using these field correctly.

Negative analysis (unselected domains/outcomes/metrics)

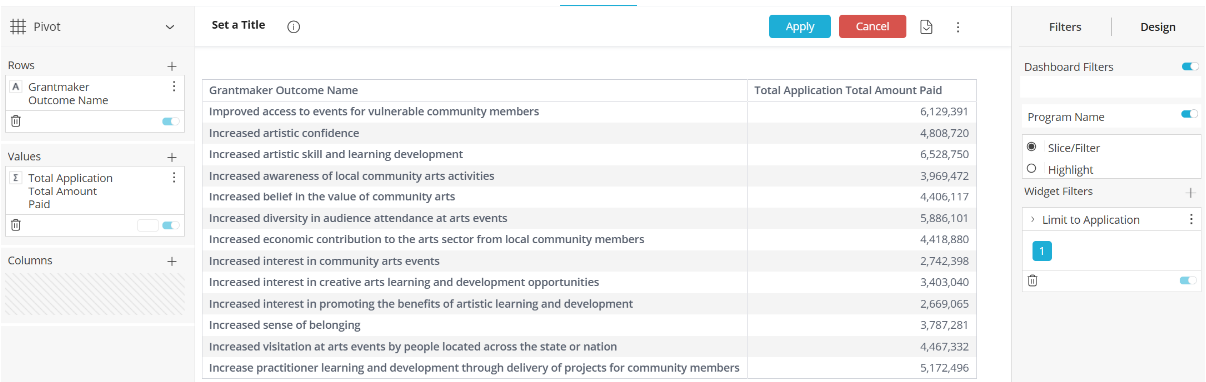
By default, the static outcomes tables will exclude frameworks/domains/outcomes/metrics which were added to a round but which have not been selected by any application. Including unselected frameworks/domains/outcomes/metrics in a widget is referred to as "negative analysis".

To include unselected frameworks/domains/outcomes/metrics in a widget, set "Limit to Application" to equal "0" or "1" (instead of the usual default "1"). In the widget below, the outcomes "Increased advocacy for animal welfare improvements and habitat protection" and "Increased choice and empowerments" are not associated with funding because no applications have selected these outcomes.



Grantmaker Outcome Name	Total Application Total Amount Paid
Improved access to events for vulnerable community members	6,129,391
Increased advocacy for animal welfare improvements and habitat protection	
Increased artistic confidence	4,808,720
Increased artistic skill and learning development	6,528,750
Increased awareness of local community arts activities	3,969,472
Increased belief in the value of community arts	4,406,117
Increased choice and empowerment	
Increased diversity in audience attendance at arts events	5,886,101
Increased economic contribution to the arts sector from local community members	4,418,880
Increased interest in community arts events	2,742,398
Increased interest in creative arts learning and development opportunities	3,403,040
Increased interest in promoting the benefits of artistic learning and development	2,669,065
Increased sense of belonging	3,787,281
Increased visitation at arts events by people located across the state or nation	4,467,332
Increase practitioner learning and development through delivery of projects for community members	5,172,496

Removing "0" from the Limit to Application filter causes "Increased advocacy for animal welfare improvements and habitat protection" and "Increased choice and empowerments" to be excluded from the widget.



Grantmaker Outcome Name	Total Application Total Amount Paid
Improved access to events for vulnerable community members	6,129,391
Increased artistic confidence	4,808,720
Increased artistic skill and learning development	6,528,750
Increased awareness of local community arts activities	3,969,472
Increased belief in the value of community arts	4,406,117
Increased diversity in audience attendance at arts events	5,886,101
Increased economic contribution to the arts sector from local community members	4,418,880
Increased interest in community arts events	2,742,398
Increased interest in creative arts learning and development opportunities	3,403,040
Increased interest in promoting the benefits of artistic learning and development	2,669,065
Increased sense of belonging	3,787,281
Increased visitation at arts events by people located across the state or nation	4,467,332
Increase practitioner learning and development through delivery of projects for community members	5,172,496

Warning: Negative analysis is only available for fields that are compatible with a Limit to Application widget filter.

In general, it is possible to use fields from the static outcomes tables to group fields from any table that is compatible following the rules for combining tables in a widget. In other words, it is possible to use the static outcomes tables to group fields from a table which either (1) contains "Application ID" in its key (2) or for which there is a Limit table whose key combination contains both the table's key and "Application ID".

However, it is not possible to show unselected frameworks/domains/outcomes/metrics when grouping fields from Funding Allocation or Payment tables because aggregating fields from those tables would require a Limit to Funding Allocation or Limit to Payments filter. See this section for a reminder on Limit to filters.

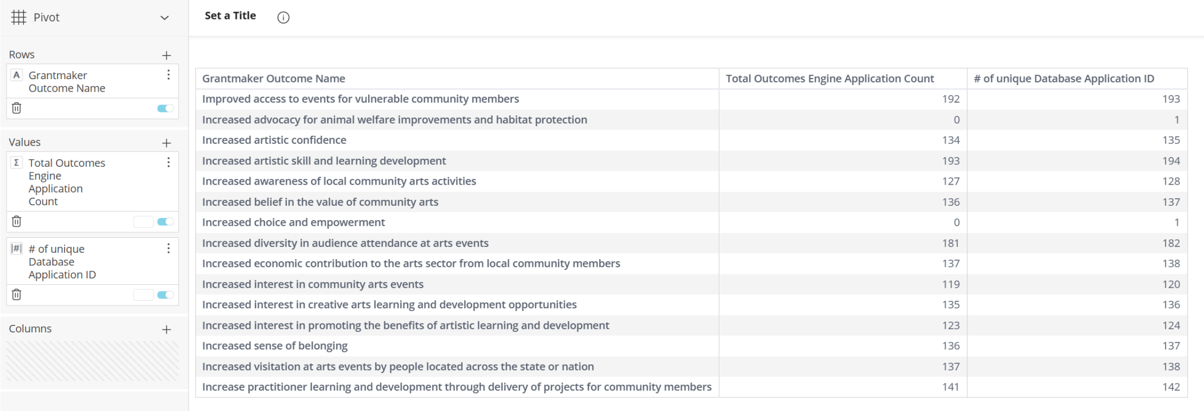
"Outcomes Engine Application Count"

Warning: if counting applications (which have selected a domain/outcome/metric) *and* implementing negative analysis, aggregate the field "Outcomes Engine Application Count" from the Outcomes Framework table.

The field “Outcomes Engine Application Count” in the Outcomes Framework table can be used in place of Database Application/Application ID when counting the number of applications using a framework/domain/outcome/metric. In general, Database Application/Application ID can still be used with outcomes-related tables. The exception is when implementing negative analysis, which is when “Outcomes Engine Application Count” must be used.

In other words, if your widget has a “Limit to Application” set to equal “0” or “1” and you are counting applications, total “Outcomes Engine Application Count” rather than count distinct Database Application IDs.

- This is because, in the background, negative analysis involves a placeholder application representing “unselected” whose id is set to “-1” (leading to a count of applications being inflated by one.)

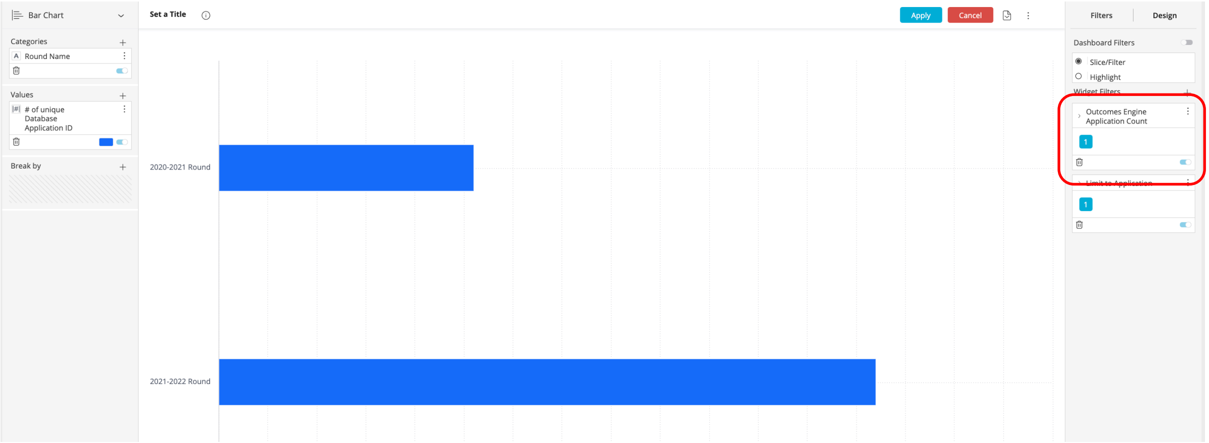

Pivot
Rows
Grantmaker Outcome Name
Values
Total Outcomes Engine Application Count
# of unique Database Application ID
Columns
Set a Title

Grantmaker Outcome Name	Total Outcomes Engine Application Count	# of unique Database Application ID
Improved access to events for vulnerable community members	192	193
Increased advocacy for animal welfare improvements and habitat protection	0	1
Increased artistic confidence	134	135
Increased artistic skill and learning development	193	194
Increased awareness of local community arts activities	127	128
Increased belief in the value of community arts	136	137
Increased choice and empowerment	0	1
Increased diversity in audience attendance at arts events	181	182
Increased economic contribution to the arts sector from local community members	137	138
Increased interest in community arts events	119	120
Increased interest in creative arts learning and development opportunities	135	136
Increased interest in promoting the benefits of artistic learning and development	123	124
Increased sense of belonging	136	137
Increased visitation at arts events by people located across the state or nation	137	138
Increase practitioner learning and development through delivery of projects for community members	141	142

Warning: “Outcomes Engine Application Count” should not be grouped by fields where an application can be associated with more than one value (e.g. fields from “Funding Allocation” since an application can have multiple funding allocations, or multi-choice question tables where an application may have selected several choices.) In these instances, it is preferable to simply use a count of distinct Application IDs (in which case negative analysis is not possible.)

This is because every application is assigned “1” for “Outcomes Engine Application Count”. Fields where an application can have multiple values (e.g. multiple funding allocations, multiple payments) would lead to duplication when totalling “Outcomes Engine Application Count”.

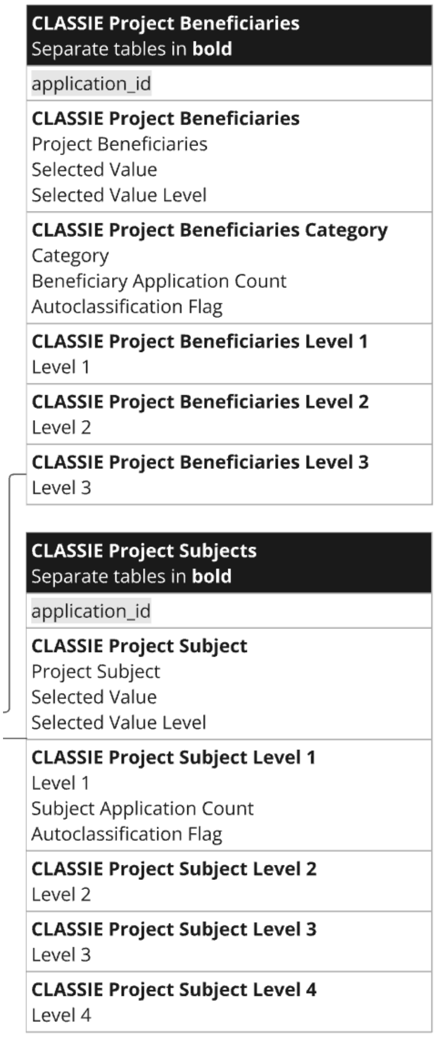
You can also use “Outcomes Engine Application Count” to filter data in widgets (including widgets using no fields from outcomes-related tables) to applications which have used the Outcomes Engine (that is, answered an Outcomes Engine standard section.) To do this, use “Outcomes Engine Application Count” as a widget filter and filter for “1”.



22 CLASSIE

If you are not familiar with CLASSIE, which allows applicants to select a project subject/beneficiary from a set list where each choice may be grouped under or belong to a parent choice, please see the Help Hub.

The CLASSIE standard fields “Project Beneficiaries” and “Project Subject” generate two separate waterfall hierarchies of five tables respectively, similar to Smart Choice Standard Fields.



CLASSIE Project Beneficiaries
Separate tables in **bold**

application_id

CLASSIE Project Beneficiaries
Project Beneficiaries
Selected Value
Selected Value Level

CLASSIE Project Beneficiaries Category
Category
Beneficiary Application Count
Autoclassification Flag

CLASSIE Project Beneficiaries Level 1
Level 1

CLASSIE Project Beneficiaries Level 2
Level 2

CLASSIE Project Beneficiaries Level 3
Level 3

CLASSIE Project Subjects
Separate tables in **bold**

application_id

CLASSIE Project Subject
Project Subject
Selected Value
Selected Value Level

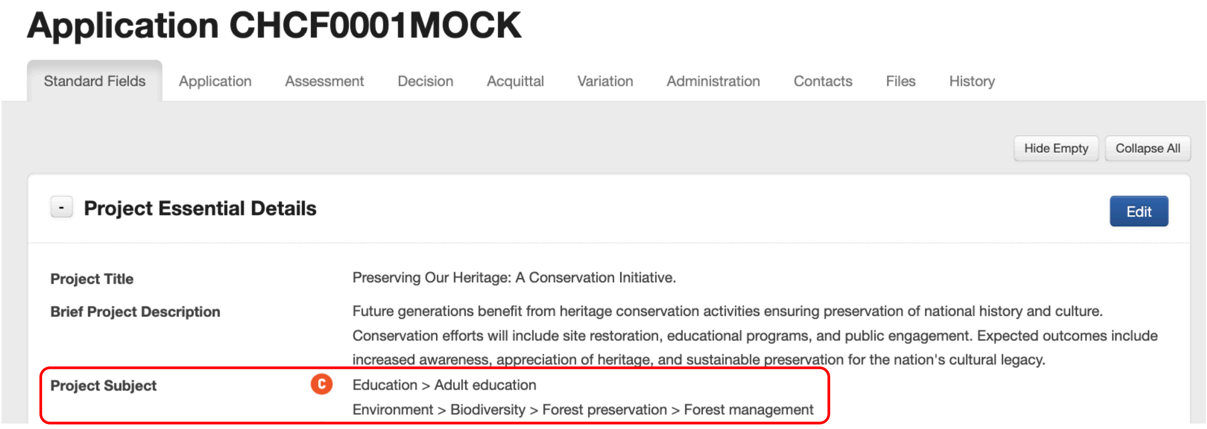
CLASSIE Project Subject Level 1
Level 1
Subject Application Count
Autoclassification Flag

CLASSIE Project Subject Level 2
Level 2

CLASSIE Project Subject Level 3
Level 3

CLASSIE Project Subject Level 4
Level 4

The easiest way to understand how CLASSIE fields are stored in the data model is through an example.



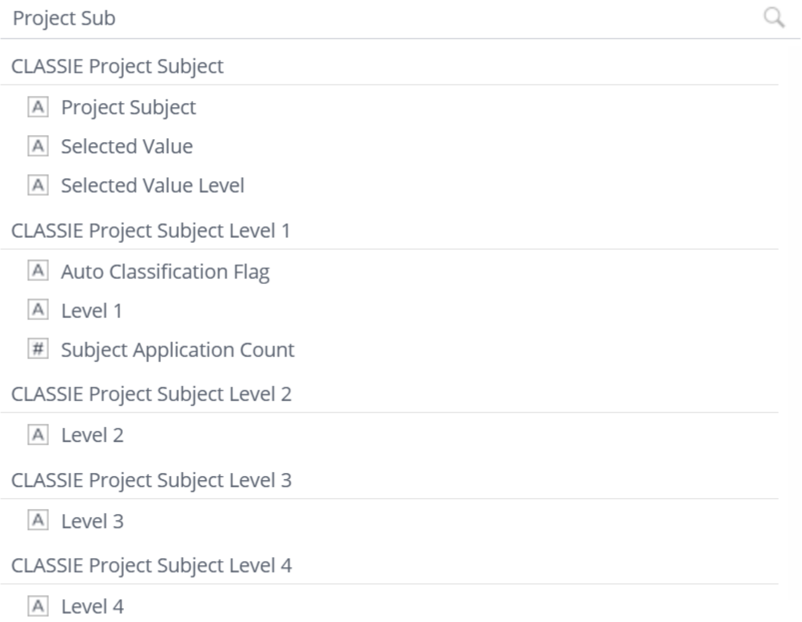
Application CHCF0001MOCK

Project Essential Details

Project Title	Preserving Our Heritage: A Conservation Initiative.
Brief Project Description	Future generations benefit from heritage conservation activities ensuring preservation of national history and culture. Conservation efforts will include site restoration, educational programs, and public engagement. Expected outcomes include increased awareness, appreciation of heritage, and sustainable preservation for the nation's cultural legacy.
Project Subject	Education > Adult education Environment > Biodiversity > Forest preservation > Forest management

If an application selects their Project Subject as “Forest management”, this is a Level 4 choice with “Forest management” nested within a hierarchy “Environment > Biodiversity > Forest preservation > Forest management”. In that hierarchy, “Forest management” is nested under the Level 3 choice “Forest preservation”, which is nested under the Level 2 choice “Biodiversity”, which is nested under the Level 1 choice “Environment”.

In the reporting tool, searching for “Project Subject” shows five different tables: one for each level in the CLASSIE hierarchy (each with the field “Level 1”, “Level 2”, “Level 3” and “Level 4” respectively) and a summary table (simply named CLASSIE Project Subject).



Project Sub

CLASSIE Project Subject

- A Project Subject
- A Selected Value
- A Selected Value Level

CLASSIE Project Subject Level 1

- A Auto Classification Flag
- A Level 1
- # Subject Application Count

CLASSIE Project Subject Level 2

- A Level 2

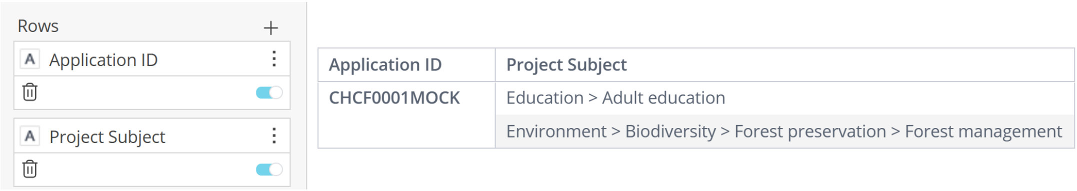
CLASSIE Project Subject Level 3

- A Level 3

CLASSIE Project Subject Level 4

- A Level 4

Below shows that selecting “Project Subject” from the CLASSIE Project Subject summary table shows selections in their layered hierarchy.



Rows +

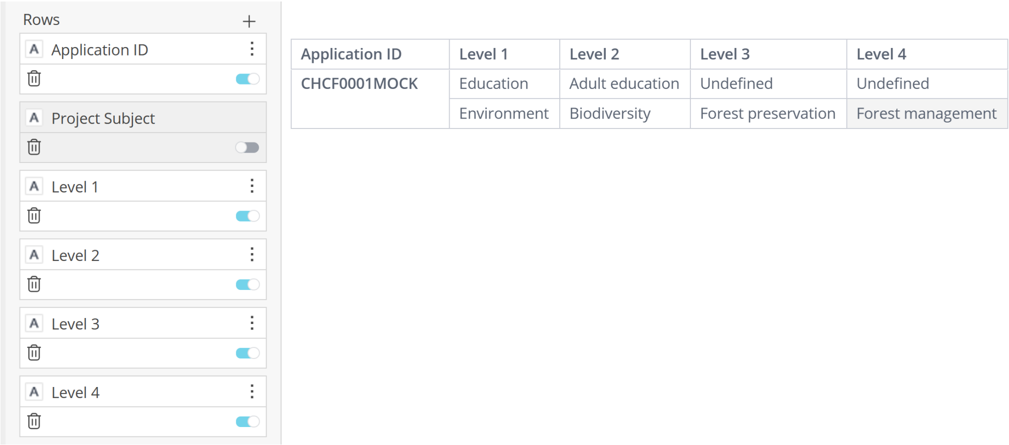
- A Application ID
- A Project Subject

Application ID	Project Subject
CHCF0001MOCK	Education > Adult education
	Environment > Biodiversity > Forest preservation > Forest management

In contrast, fields from the level tables break down the layered selection. In this selection, “Environment > Biodiversity > Forest preservation > Forest management” generates:

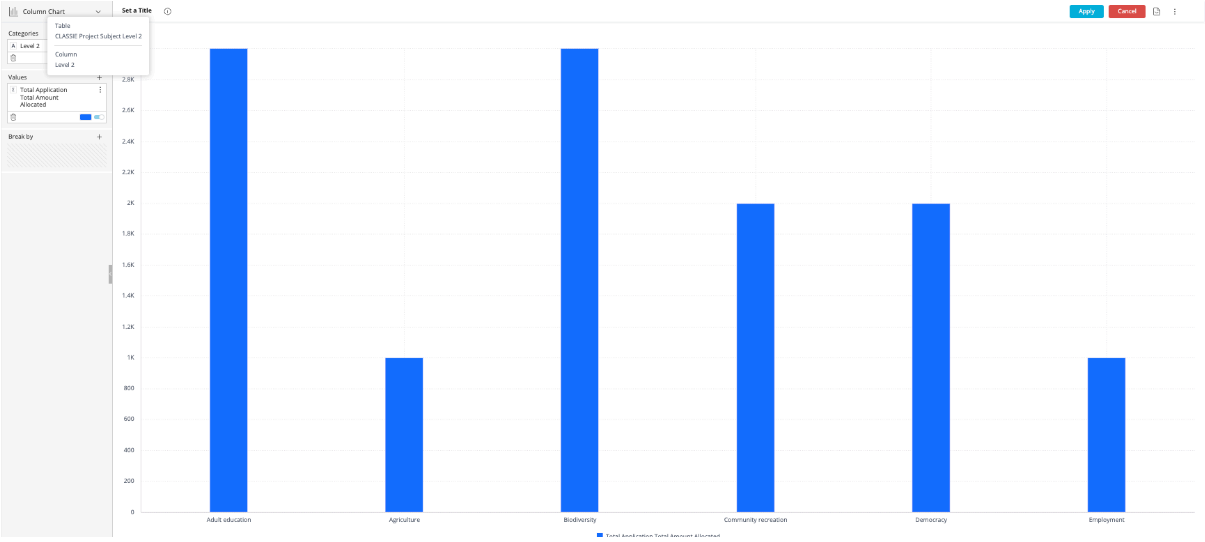
- “Forest management” for the field “Level 4” from CLASSIE Project Subject Level 4
- “Forest preservation” for the field “Level 3” from CLASSIE Project Subject Level 3
- “Biodiversity” for the field “Level 2” from CLASSIE Project Subject Level 2
- “Environment” for the field “Level 1” from CLASSIE Project Subject Level 1

The application’s second selection of “Education > Adult Education” did not drill down beyond Level 2, so is given “Undefined” for the fields “Level 3” and Level 4”.



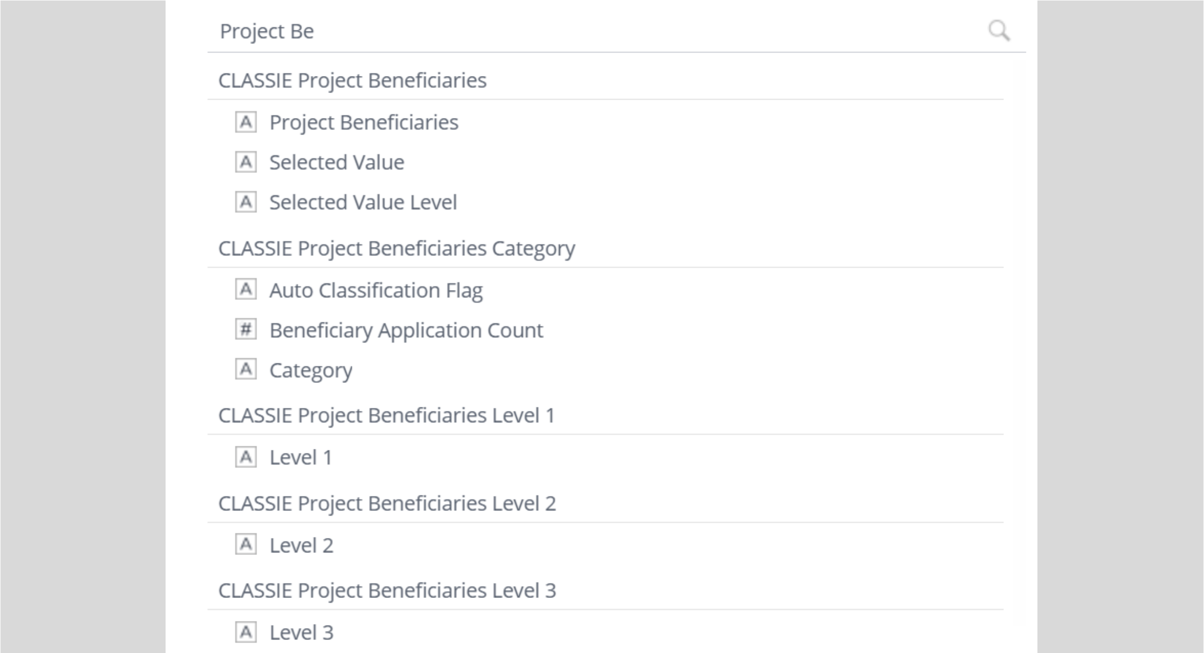
Application ID	Level 1	Level 2	Level 3	Level 4
CHCF0001MOCK	Education	Adult education	Undefined	Undefined
	Environment	Biodiversity	Forest preservation	Forest management

Use the level tables to group data by different smart choice levels. The below chart uses "Level 2" from CLASSIE Project Subject Level 2 to aggregate the amount allocated to applications that selected different Level 2 values.



While examples in this section covered Project Subject, the same information applies for Project Beneficiaries.

The only difference is that Project Beneficiaries replace Level 1 with Category, drilling down progressively to Level 1, Level 2 and Level 3.



Project Be

CLASSIE Project Beneficiaries

- A Project Beneficiaries
- A Selected Value
- A Selected Value Level

CLASSIE Project Beneficiaries Category

- A Auto Classification Flag
- # Beneficiary Application Count
- A Category

CLASSIE Project Beneficiaries Level 1

- A Level 1

CLASSIE Project Beneficiaries Level 2

- A Level 2

CLASSIE Project Beneficiaries Level 3

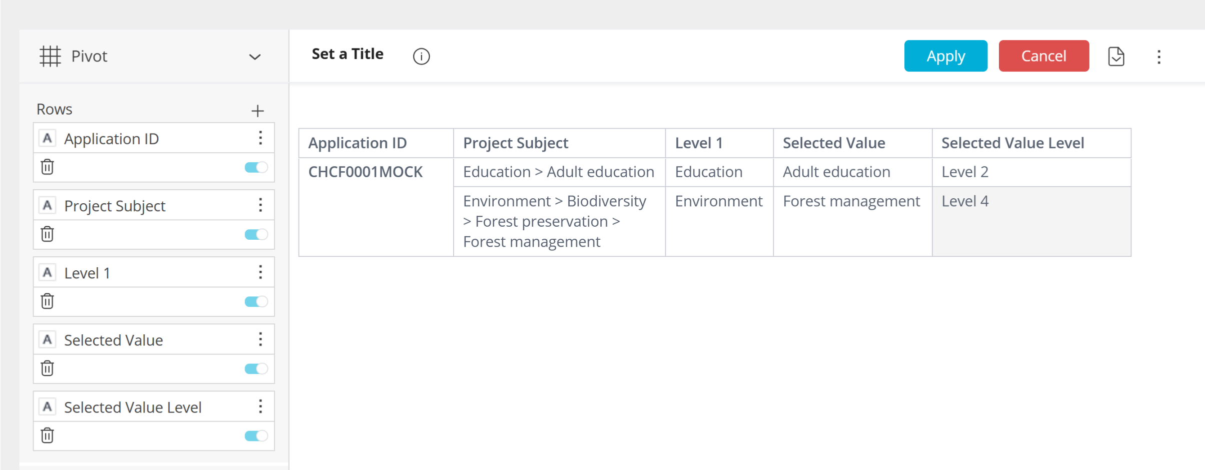
- A Level 3

The summary CLASSIE Project Subject table and the CLASSIE Subject Level 1 table contain several other fields to facilitate analysis (they are split across these two tables for data modelling purposes).

Selected Value + Selected Value Level

“Selected Value” gives the lowest level value that a CLASSIE response drilled down to. “Selected Value Level” gives the level number that a CLASSIE response drilled down to.

- For the example “Environment > Biodiversity > Forest preservation > Forest management”, “Selected Value” gives “Forest Management” and “Selected Value Level” gives “Level 4”.
- If another application selected “Environment > Biodiversity > Forest preservation”, “Selected Value” would give “Forest preservation” and “Selected Value Level” would give “Level 3.”



Rows

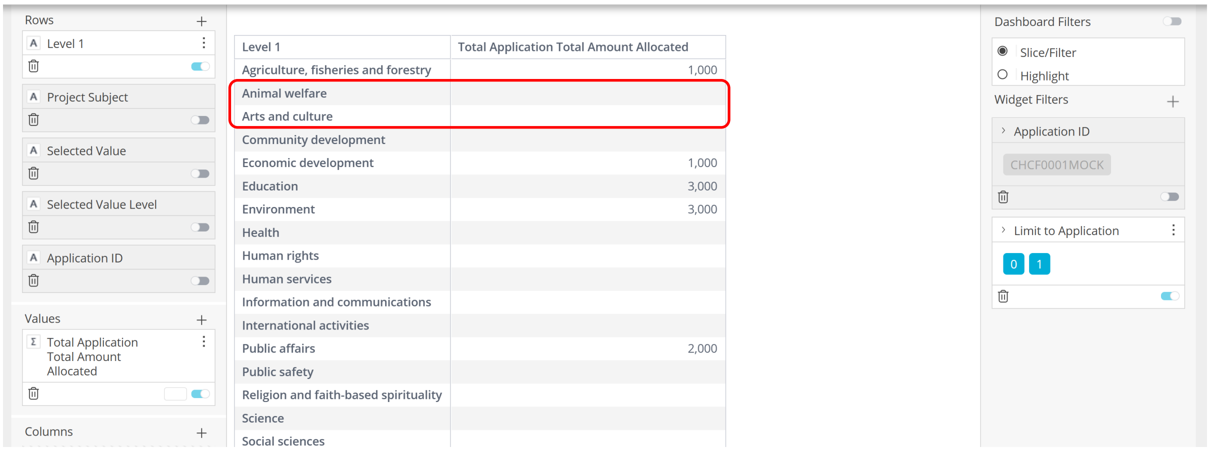
- A Application ID
- A Project Subject
- A Level 1
- A Selected Value
- A Selected Value Level

Application ID	Project Subject	Level 1	Selected Value	Selected Value Level
CHCF0001MOCK	Education > Adult education	Education	Adult education	Level 2
	Environment > Biodiversity > Forest preservation > Forest management	Environment	Forest management	Level 4

Negative Analysis (unselected subjects/beneficiaries)

By default, the reporting tool will exclude subjects/beneficiaries which were added to a round but which have not been selected by any application. To include unselected subjects/beneficiaries in a widget, set “Limit to Application” to equal “0” or “1” (instead of the usual default “1”). In the widget below, “Animal Welfare” and “Arts and Culture” do not have any associated funding because no applications selected these project subjects.

Analytics



○ Slice/Filter
○ Highlight

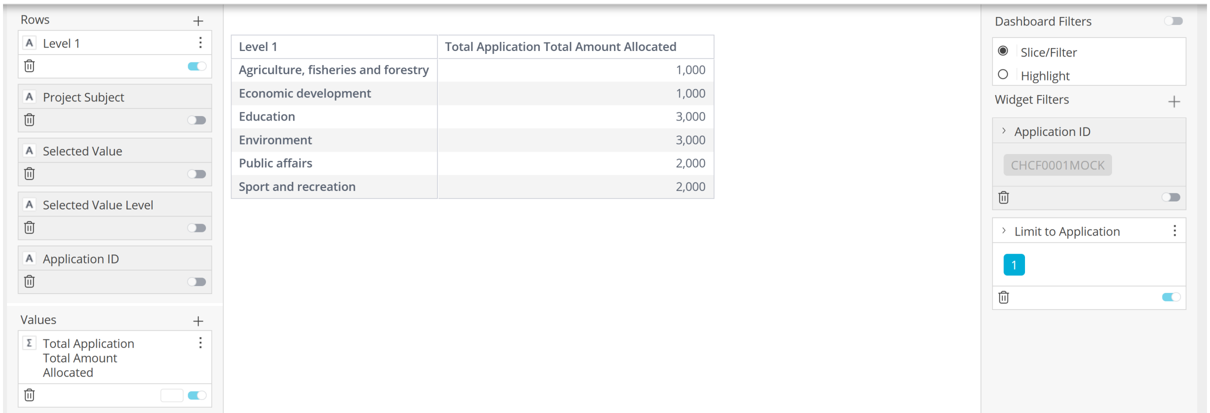
Application ID: CHCF0001MOCK

Limit to Application: 0 1

Level 1	Total Application Total Amount Allocated
Agriculture, fisheries and forestry	1,000
Animal welfare	
Arts and culture	
Community development	
Economic development	1,000
Education	3,000
Environment	3,000
Health	
Human rights	
Human services	
Information and communications	
International activities	
Public affairs	2,000
Public safety	
Religion and faith-based spirituality	
Science	
Social sciences	

Removing “0” from the Limit to Application filter causes “Animal Welfare” and “Arts and Culture” (and all other unselected subjects) to be omitted from the widget.

Analytics



○ Slice/Filter
○ Highlight

Application ID: CHCF0001MOCK

Limit to Application: 1

Level 1	Total Application Total Amount Allocated
Agriculture, fisheries and forestry	1,000
Economic development	1,000
Education	3,000
Environment	3,000
Public affairs	2,000
Sport and recreation	2,000

Warning: Negative analysis is only available with fields that are compatible with a Limit to Application widget filter.

In general, it is possible to use fields from the CLASSIE tables to group fields from any table that is compatible following the rules for combining tables in a widget. In other words, it is possible to use the CLASSIE tables to group fields from a table which either (1) contains “Application ID” in its key (2) or for which there is a Limit table whose key combination contains both the table’s key and “Application ID”.

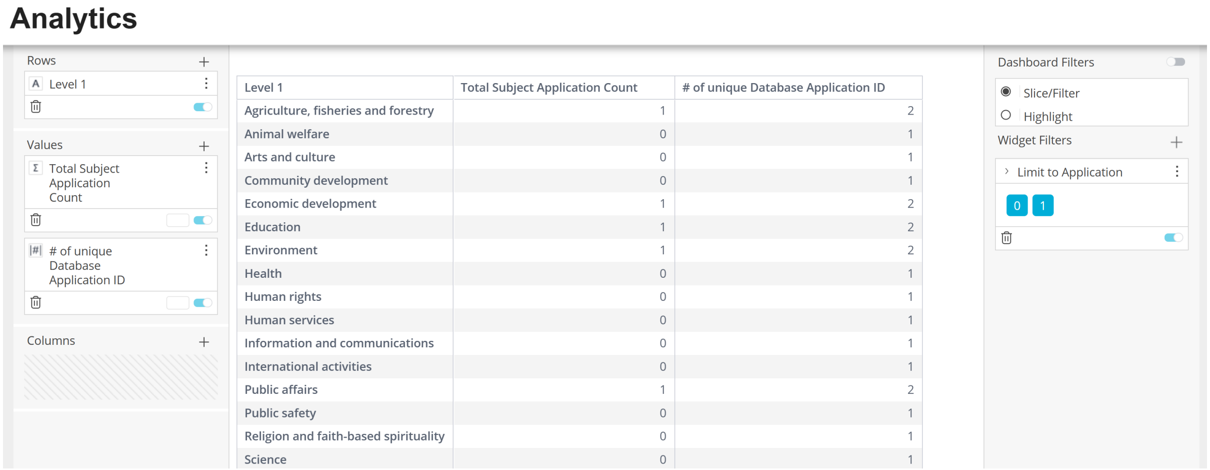
However, it is not possible to show unselected subjects/beneficiaries when grouping fields from Funding Allocation or Payment tables because aggregating fields from those tables would require a Limit to Funding Allocation or Limit to Payments filter. See this section for a reminder on Limit to filters.

“Beneficiary Application Count”/ “Subject Application Count”

Warning: if counting applications (which have selected a project/beneficiary) *and* implementing negative analysis, aggregate the fields “Subject Application Count”/ “Beneficiary Application Count”.

The fields “Beneficiary Application Count”/ “Subject Application Count” can be used in place of Database Application/Application ID when counting the number of applications which selected a beneficiary/subject. In general, Database Application/Application ID can still be used. The exception is when implementing negative analysis which is when “Beneficiary Application Count”/ “Subject Application Count” must be used.

In other words, if your widget has a “Limit to Application” set to equal “0” or “1” and you are counting applications, total “Subject Application Count”/ “Beneficiary Application Count” rather than count distinct Database Application IDs. This is because, in the background, negative analysis involves artificially counting an application whose id is set to “-1” (leading to a count of applications being inflated by one.)



Analytics

Rows: Level 1

Values: Total Subject Application Count; # of unique Database Application ID

Columns

Dashboard Filters: ● Slice/Filter ○ Highlight

Widget Filters: Limit to Application: 0 1

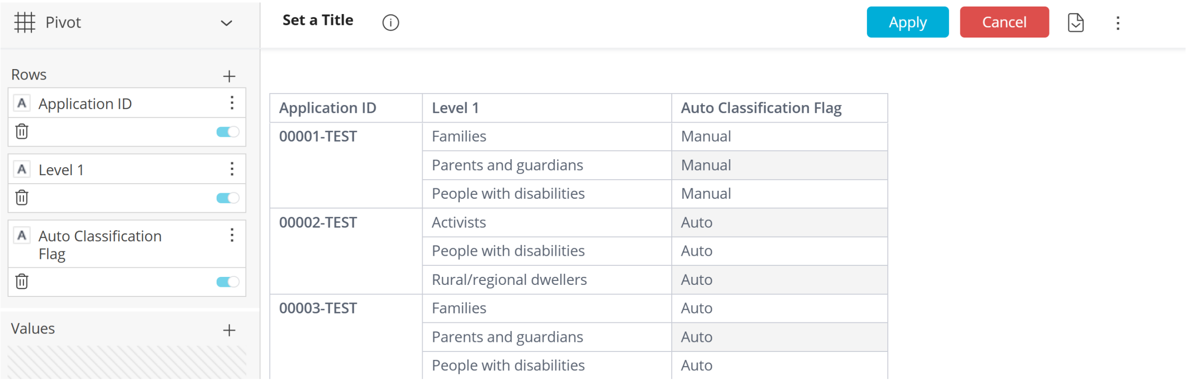
Level 1	Total Subject Application Count	# of unique Database Application ID
Agriculture, fisheries and forestry	1	2
Animal welfare	0	1
Arts and culture	0	1
Community development	0	1
Economic development	1	2
Education	1	2
Environment	1	2
Health	0	1
Human rights	0	1
Human services	0	1
Information and communications	0	1
International activities	0	1
Public affairs	1	2
Public safety	0	1
Religion and faith-based spirituality	0	1
Science	0	1

Warning: “Beneficiary Application Count”/ “Subject Application Count” should not be grouped by fields where an application can be associated with more than one value (e.g. fields from “Funding Allocation” since an application can have multiple funding allocations, or multi-choice question tables where an application may have selected several choices.) In these instances, it is preferable to simply use a count of distinct Application IDs (in which case negative analysis is not possible.)

This is because every application is assigned “1” for “Beneficiary Application Count”/ “Subject Application Count”. Fields where an application can have multiple values (e.g. multiple funding allocations, multiple payments) would lead to duplication when totalling “Beneficiary Application Count”/ “Subject Application Count”.

Autoclassification Flag

“Auto Classification Flag” can be used to identify responses that were manually input by applicants versus auto-classified by the CLASSIEfier tool.



Application ID	Level 1	Auto Classification Flag
00001-TEST	Families	Manual
	Parents and guardians	Manual
	People with disabilities	Manual
00002-TEST	Activists	Auto
	People with disabilities	Auto
	Rural/regional dwellers	Auto
00003-TEST	Families	Auto
	Parents and guardians	Auto
	People with disabilities	Auto

23 Tasks

Fields related to tasks are split across three tables: Task, Task Assignee and Task Form. As per the rules for combining tables, these tables can all be used together in widgets.

Task Form contains basic information (form names and purpose types) of forms attached to tasks. However, Task Form should not be combined with the Form Response table or dynamic form tables in widgets (the data model will join all form responses associated with an application to all tasks associated with the application; in other words the data model cannot match a task with the questions and responses to the form associated with the task.)