SmartyGrants
Software, data science & grantmaking intelligence

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

SmartyGrants Analytics Technical Documentation

Version 1.3
SmartyGrants Version 8.26.25
21 November 2024

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

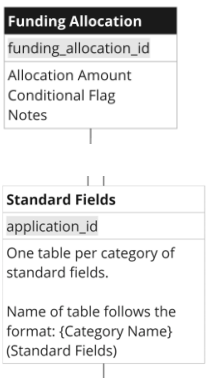
This document contains an in-depth breakdown of the data model used in SmartyGrants Analytics.

- Users seeking a quick 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

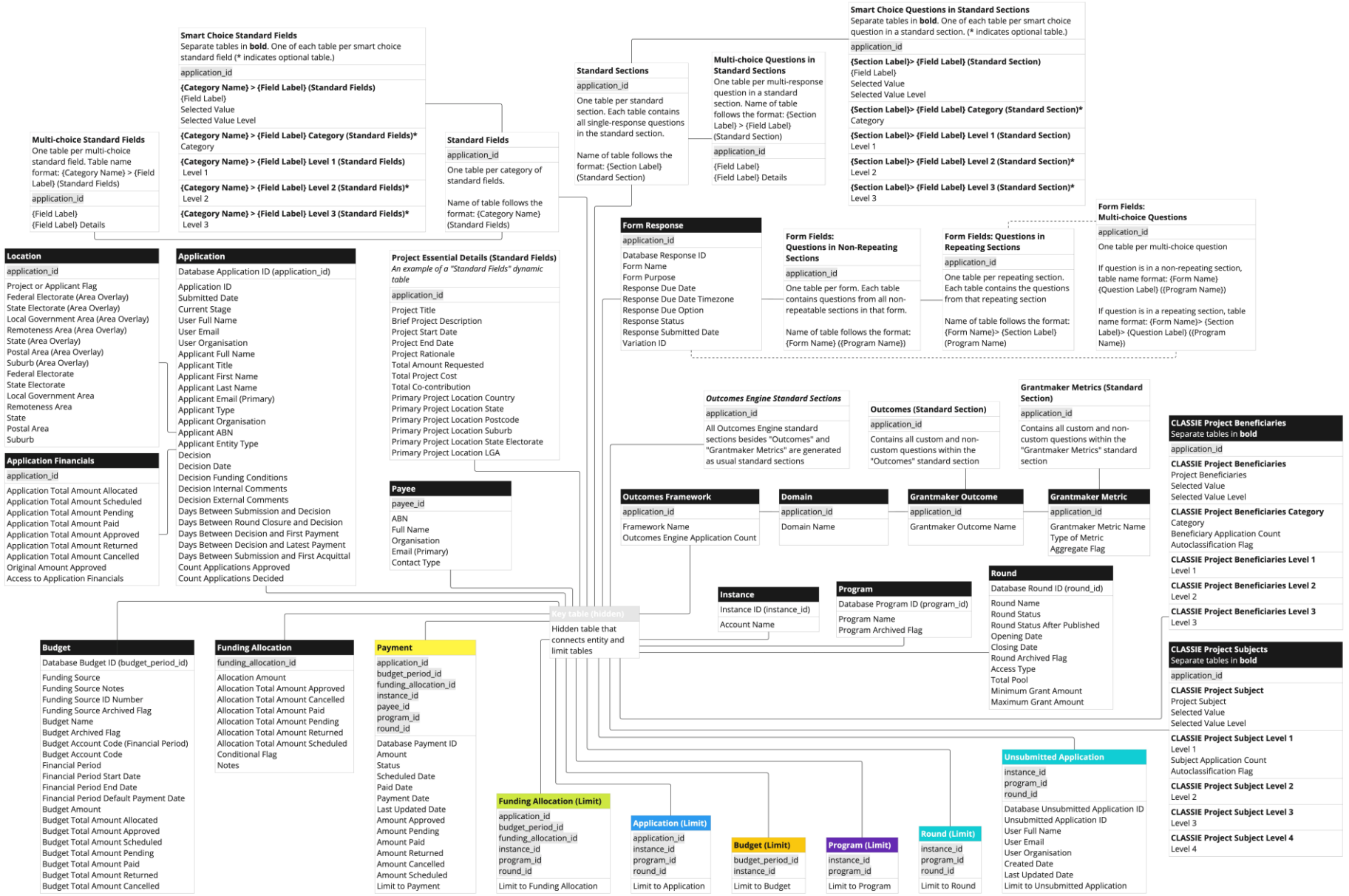
	Definition	**Examples**
Entity	An object or concept in SmartyGrants	An application
Key	A unique ID which identifies an entity.	Every application has a unique application_id
Key Combination	The combination of an entity's key with the keys of other entities that it is "nested" under or belongs to. The 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



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



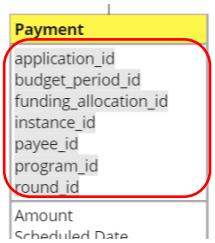
5 Designing a Widget: Steps

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.

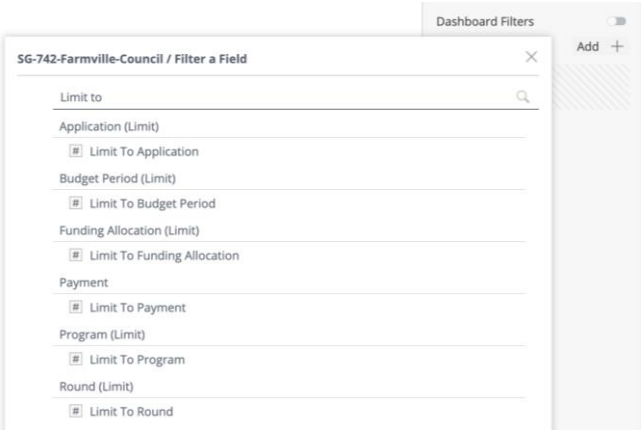
2. Check it is possible to combine these tables in a single widget



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



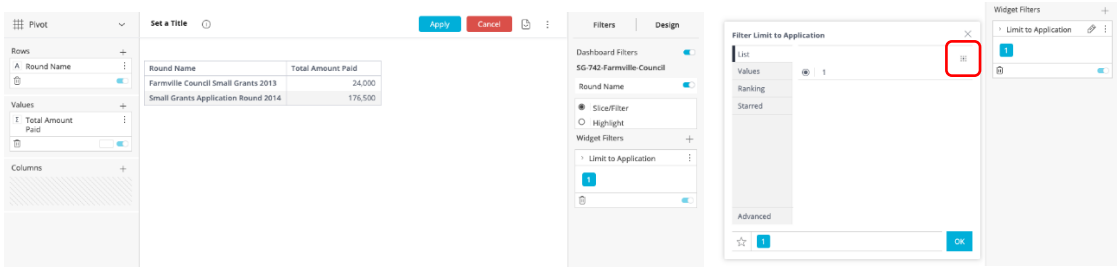
Every widget requires a "Limit to <entity>" field from a limit table applied as a widget filter. For an explanation of why this filter is needed, see the section "Limit Tables."

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 contained in its 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.

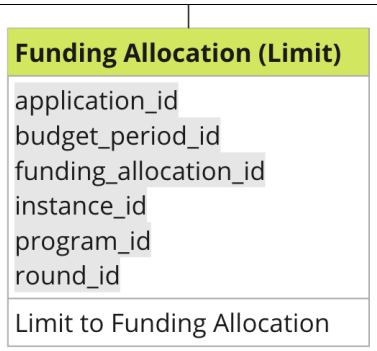
To apply a "limit to" field as a widget filter, set it to include values equalling 1. When setting a filter to equal 1, select List and click the nine-dot icon.



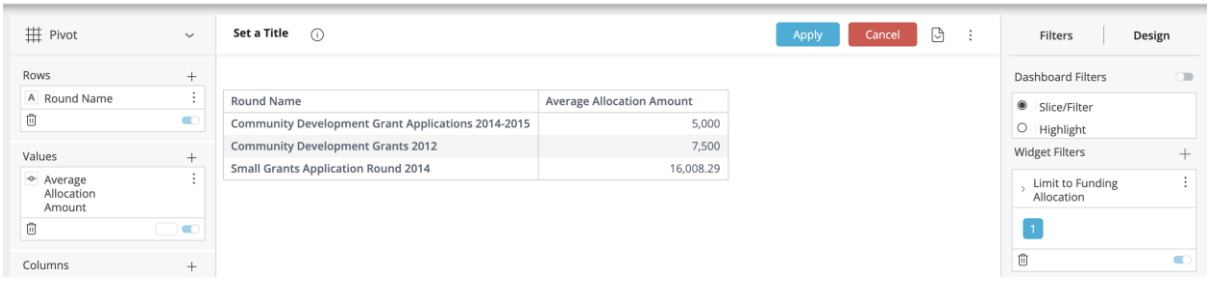
6 Designing a Widget: Worked Examples

"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	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 the key combination of 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.

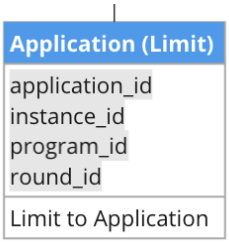
Analytics



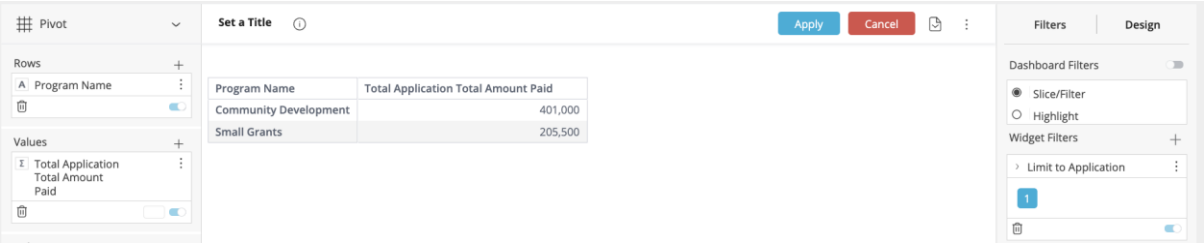
"I would like to sum the total amount paid to applications from each program."

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 that data by "Program Name" from the Program table.
Step 2: Check it is possible to combine these tables in a single widget 	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 the key combination of Application (Limit). This means Program and Application tables can be used in a single widget.

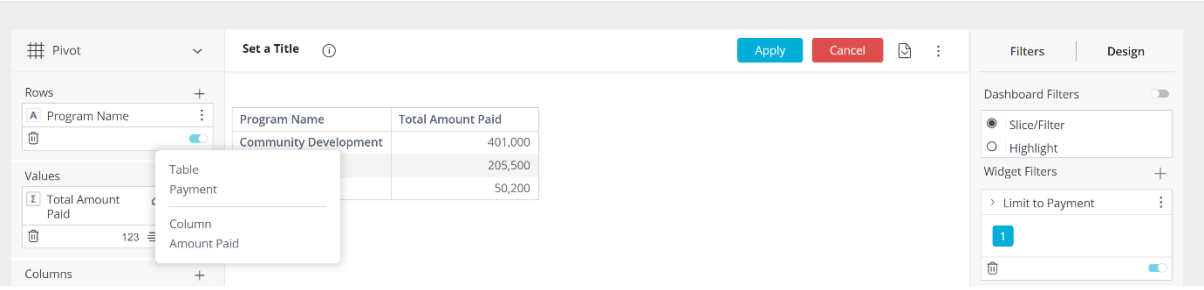
Step 3: Select a Limit To filter	This widget is aggregating “Application Total Amount Paid,” a field that gives information about applications. Apply “Limit to Application” from the Application (Limit) table as a widget filter.



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 entity 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.

Analytics



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

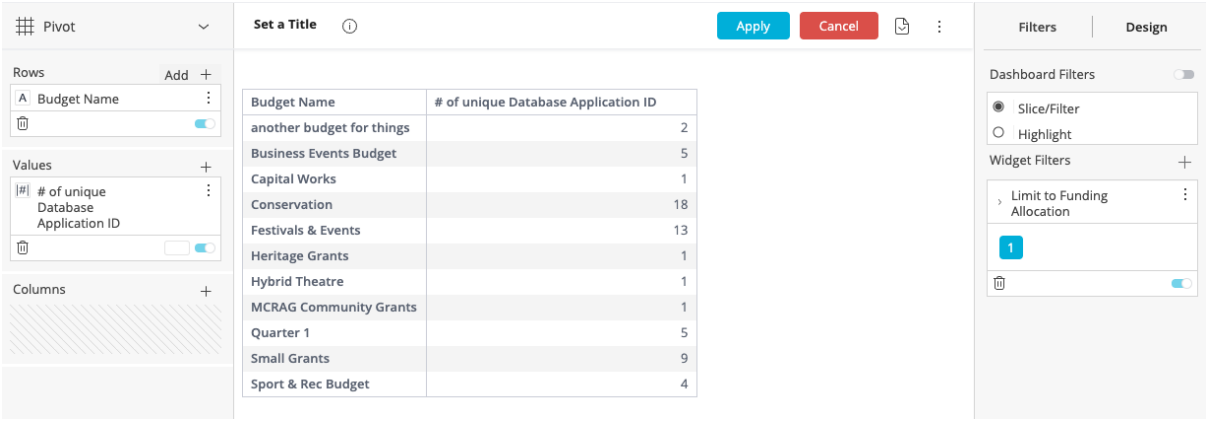
The previous example with a budget-related reporting requirement.

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 summing “Application Total Amount Paid” 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.

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

Step	Reasoning

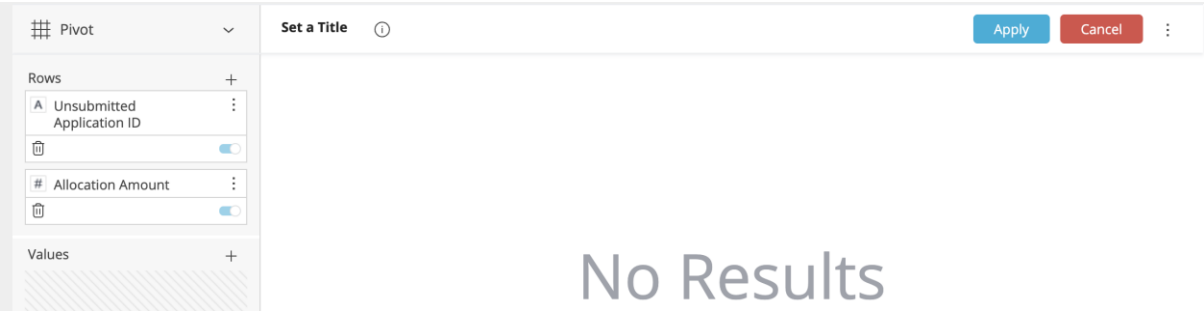
Step 1: Find the tables containing the fields you will use	This widget will provide a list of Database Application IDs, from the Application table. This list will be split by Budget Name from the Budget table.
Step 2: Check it is possible to combine these tables in a single widget **Funding Allocation (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 of 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*.



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.

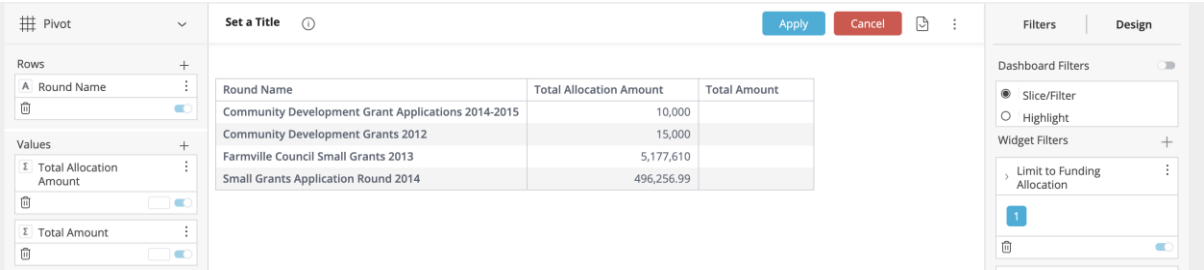


If a widget shows No Results, it is likely the combination of fields in the widget violates the rules for combining tables in a widget. In most cases, this result indicates a combination of fields that does not make logical sense.

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.

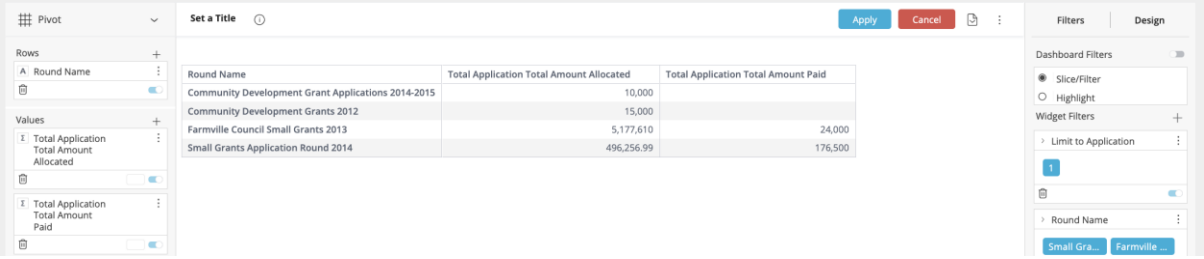
That these fields cannot be logically combined is reflected in the rules for combining tables in a widget, whereby an entity table's key is contained in the key combination of any limit table it can be combined with. Funding Allocation ID is not contained in the key combination of Unsubmitted Application.

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.



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 you have 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?

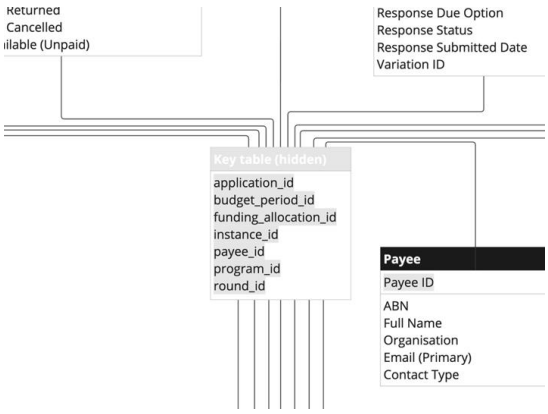
As explained in the first FAQ, if the fields you are trying to combine in a widget violate the rules for combining tables, this may indicate you have chosen a combination of fields that does not make logical sense. Otherwise, there may be a way to produce the analysis you have in mind through a combination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

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

8 The Key Table



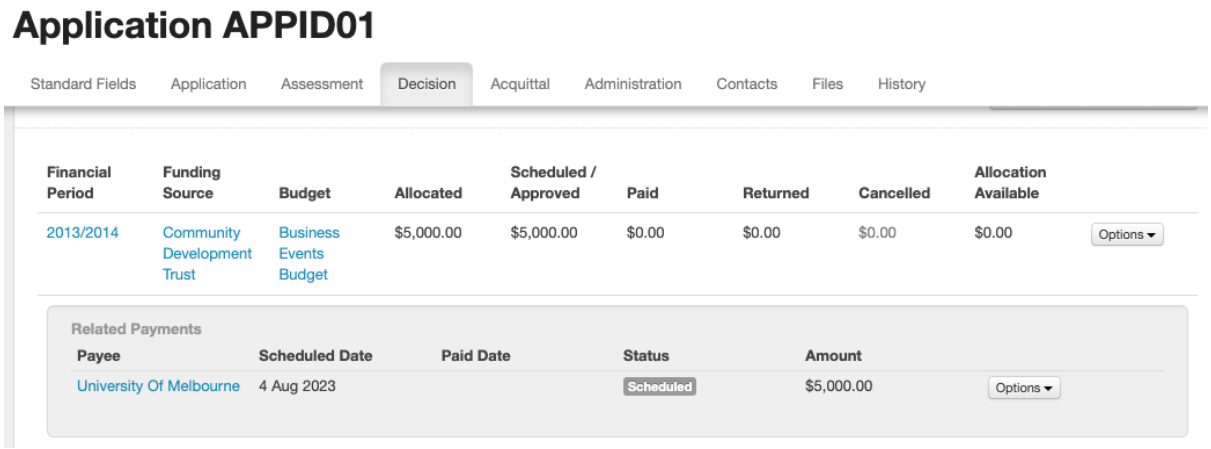
Why does the data model need the key table?

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: Application ID, Round ID, Program ID, Instance ID, Budget Period ID, Funding Allocation ID and Payee ID)

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.



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 associated with each ID, 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.

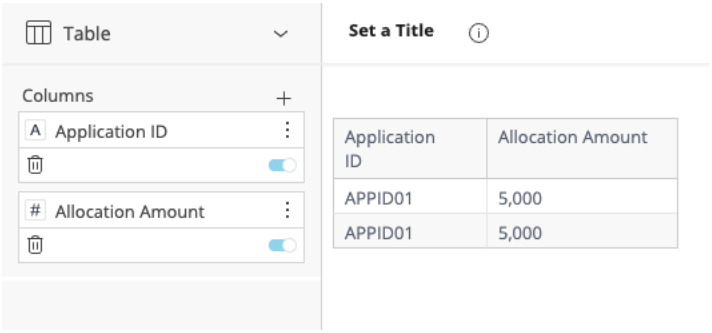
Application ID	Current Stage	Decision Date
APPID01	Evaluation	3/3/2024

The Funding Allocation table contains the fields “Allocation Amount” and “Conditional Flag”, with this data organised in rows by the key, Funding Allocation ID (some keys such as Funding Allocation ID have been hidden from the end user).

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



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 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 permutation 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, either 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.)
- The entity is a historical event (as in the case of Round Approval, which contains both past and present round approval decisions)

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.

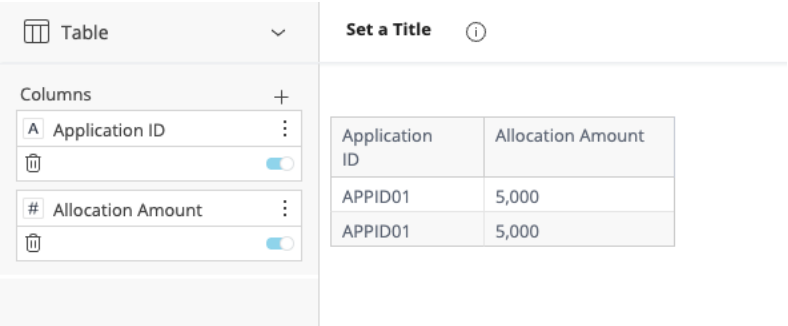
Why do we need Limit Tables?

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

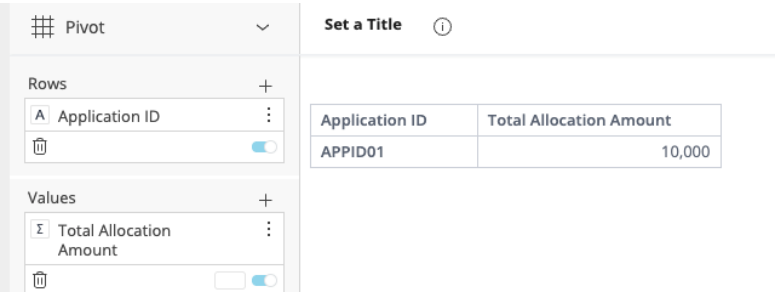
- The key table contains all actual permutations of every key combination.
- There are overlapping keys across key combinations. For example, Application ID is used both in the key combination of Payment and the key combination of Funding Allocation.

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.



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.



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.

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:

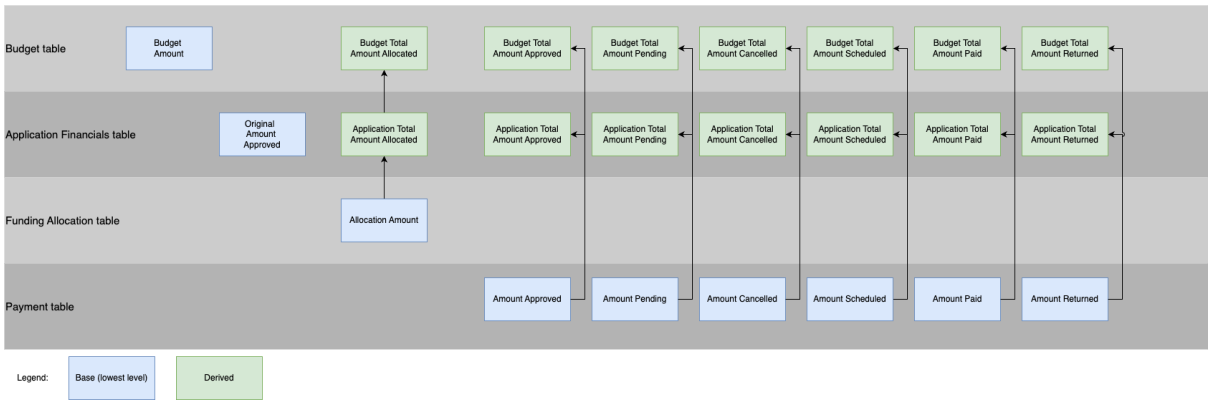
Step	Example
(1) Selection of a field from the Application table prompts widget builder to creates a list of Application IDs from that table	APPID01
(2) Widget filter of "Limit to Funding Allocation = 1" creates a join from the Funding Allocation (Limit) table to the key table on all seven ID columns	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 therefore only preserves rows with Payee ID set to "-1" (and matching the key combination permutations in Funding

	Allocation (Limit) table). 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

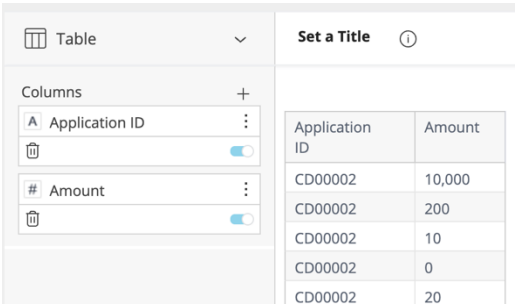
In the data model, financial data is pre-aggregated at different levels in the Application, Funding Allocation, Payment and Budget tables. For example, individual payments are recorded in the Payment table. Payments are grouped by application in the Application Financials table and grouped by budget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



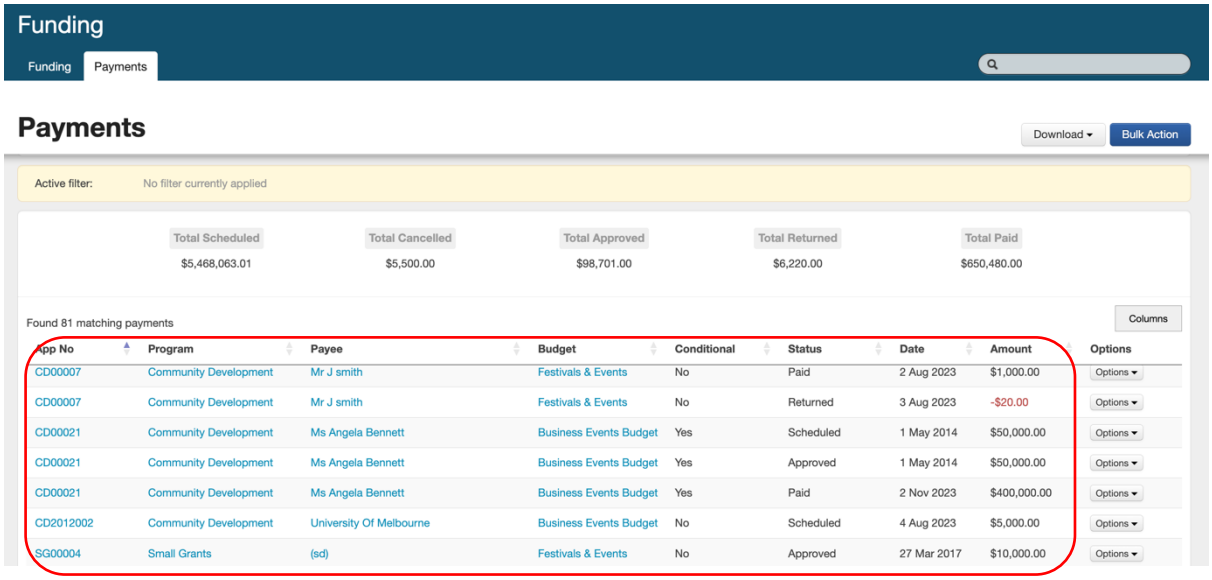
This pre-aggregation maximises flexibility for the kind of financial reporting that is possible in SmartyGrants Analytics. To use these tables effectively, you should have a firm understanding of how funding is administered in SmartyGrants (or refer to the "Funding and Payments" page on the Help Hub).

Payment table

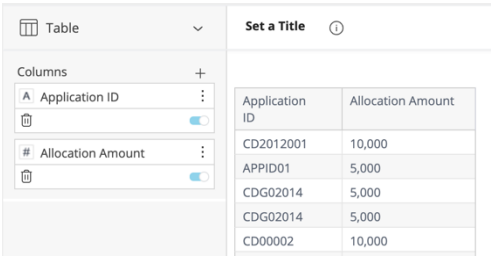


The Payments table contains data around individual payments (i.e. one row per payment), similar to the Payments page in SmartyGrants. Applications can receive multiple payments, which are separated out into different rows in this table.

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).

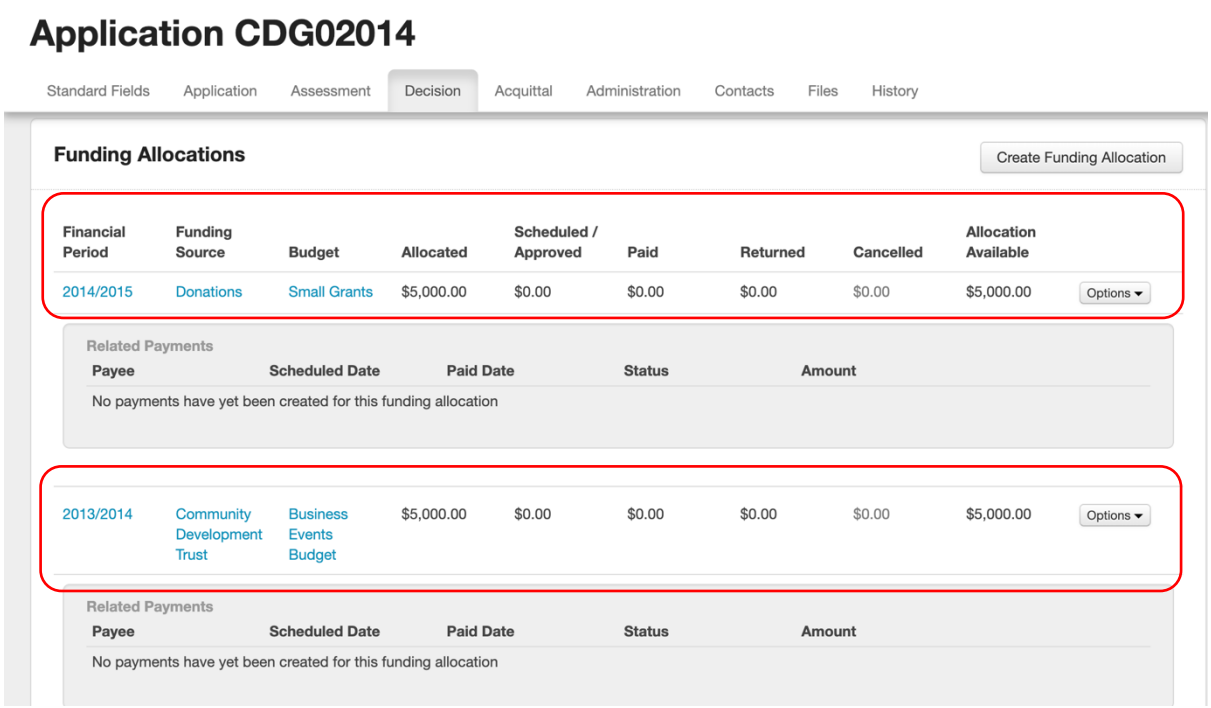


Funding Allocation table



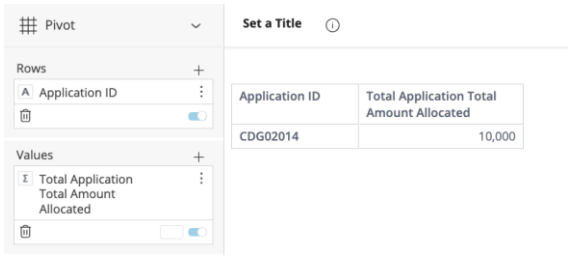
The Funding Allocation table contains data around funding allocations (i.e. one row per funding allocation). Applications can receive multiple funding allocation, which are separated out into different rows in this table.

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



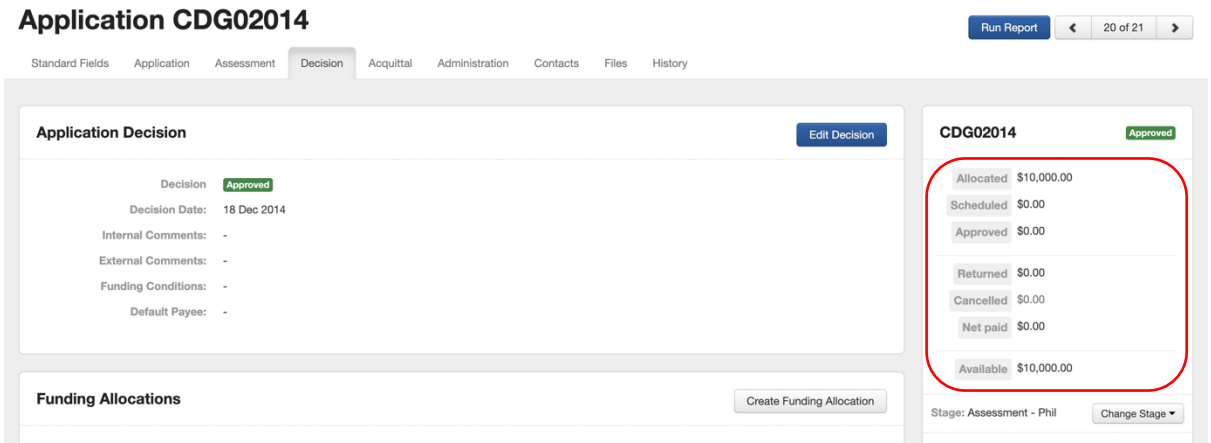
Application Financials table

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.



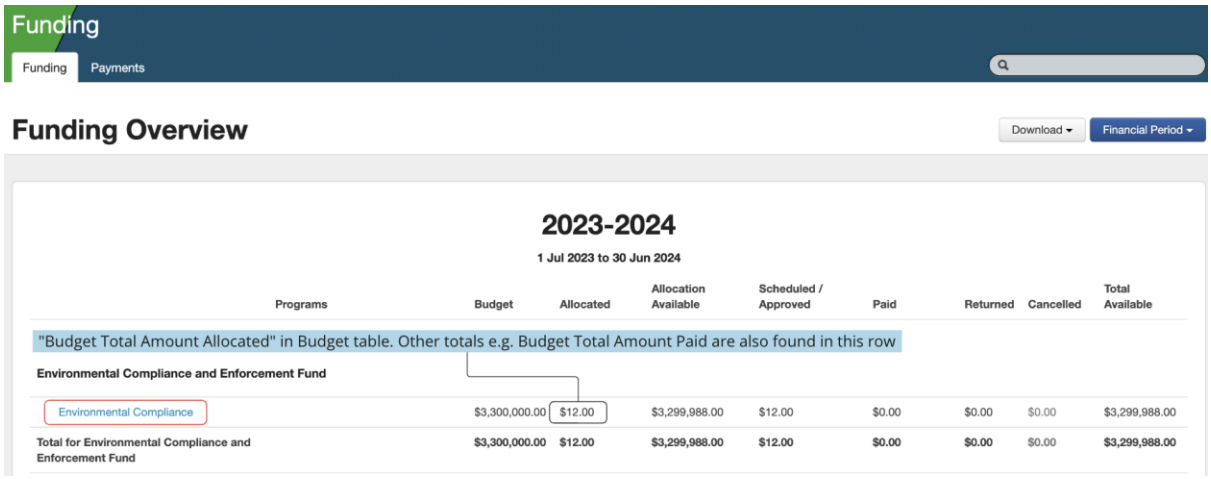
In the previous image, we saw that CDG02014 received two funding allocations of $5000 which were stored in two separate rows in the Funding Allocation table. In the Application Financials table, CDG02014 has “Application Total Amount Allocated” of $10,000.

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

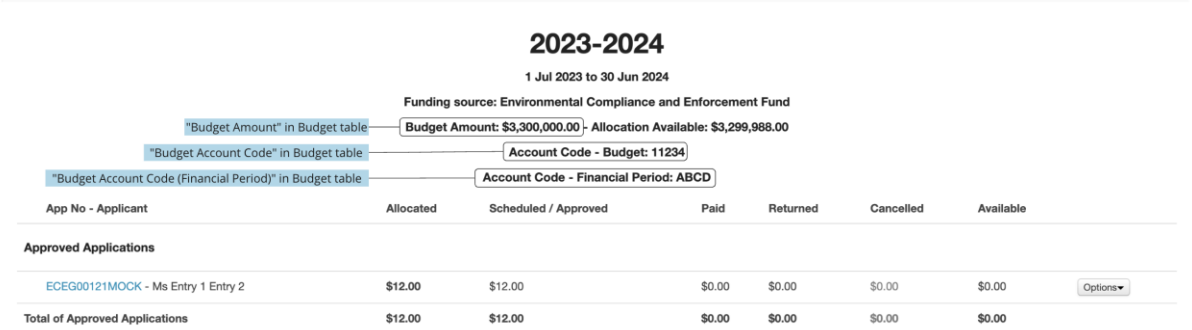


Budget table

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.



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



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 applicant."

- 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 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 may 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 (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

How 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 (hidden from end user) 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, "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 (hidden from end user) 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.

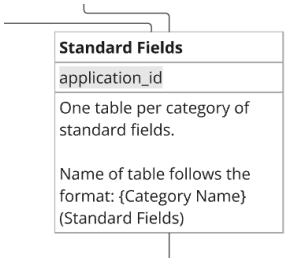
Why are the numbers in my widget using Payment table not as expected?

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 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.



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 form/standard section/ standard field are generally split into tables by question type (single-response vs various multi-choice type 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, form, 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

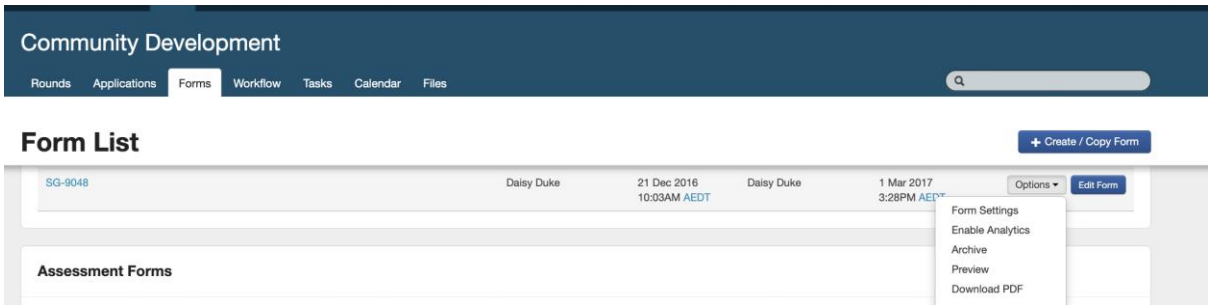
are refreshed nightly (the nightly refresh window is 1am - 6am. However accounts are enabled progressively and most accounts should complete the within an hour.) This means if you enable analytics for a form, responses to the form will appear in SmartyGrants Analytics the next day. If you make a change 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.

15 Forms

Enabling reporting on forms

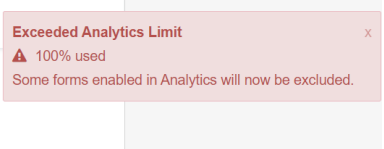
SmartyGrants Analytics facilitates reporting on responses to forms. However, data only appears in SmartyGrants Analytics for forms which have reporting enabled (responses to a form will not appear in SmartyGrants Analytics automatically.)

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**.



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.)



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.

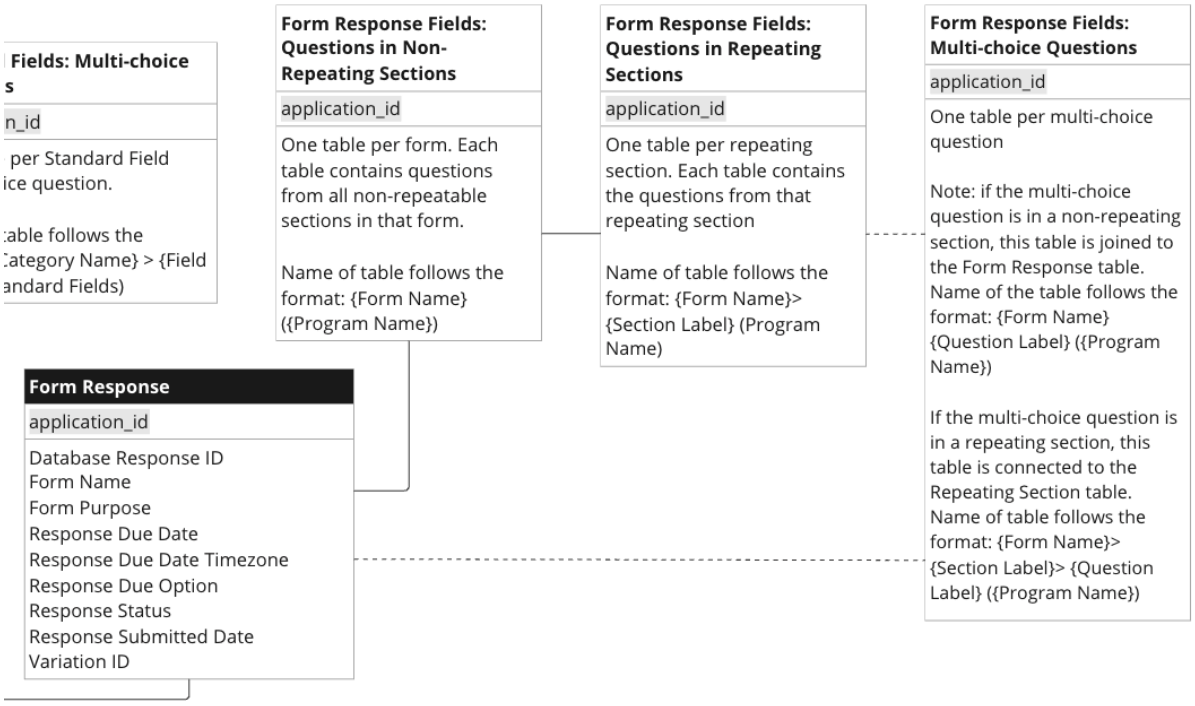
Form fields in the data model

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 "meta-data" 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



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

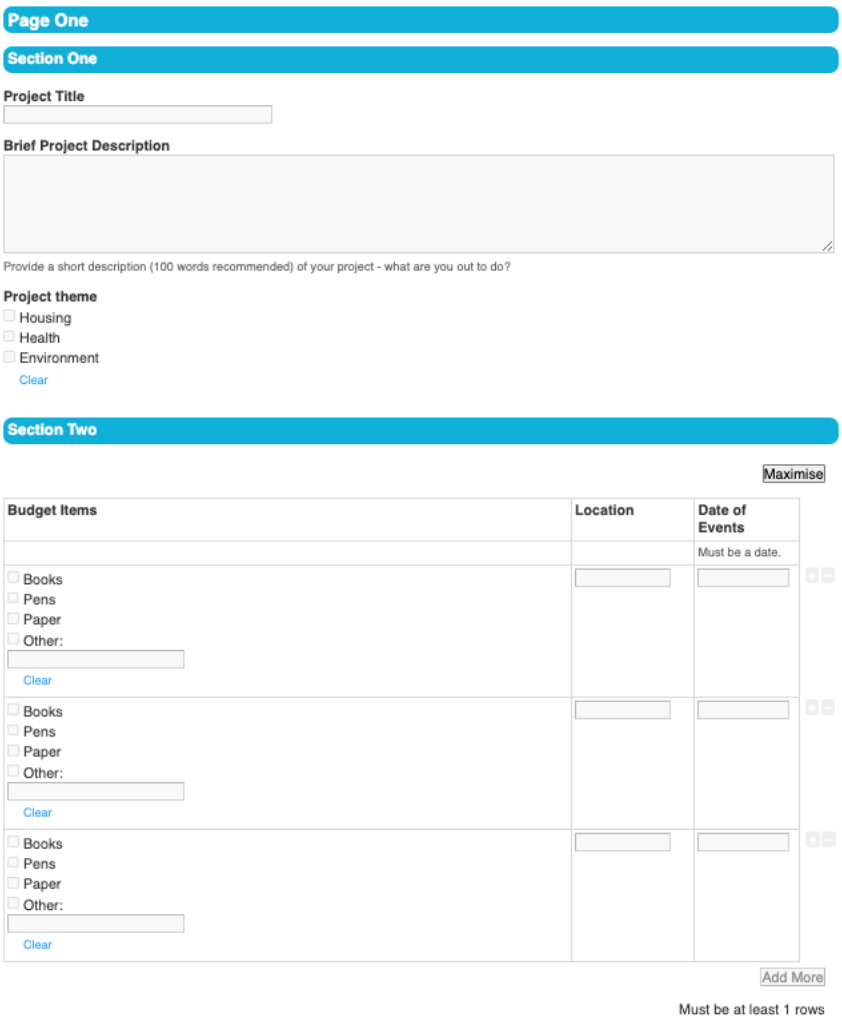
For a single form, this can generate up to four table types (with potentially several cases of each type). The naming structure these four form table types is:

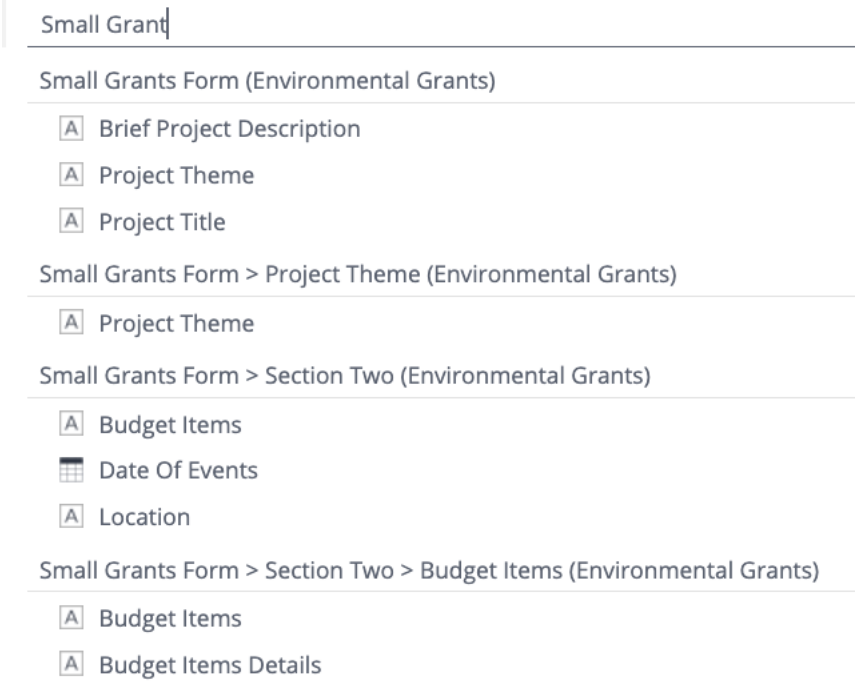
Tables for single-response questions in non-repeatable sections	Form Name (Program Name) e.g. Short Application Form (Sports Grants)
Tables for multi-choice questions in non-repeatable sections	Form Name > Question Label (Program Name) e.g. Short Application Form > Project Locations (Sports Grants)
Tables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

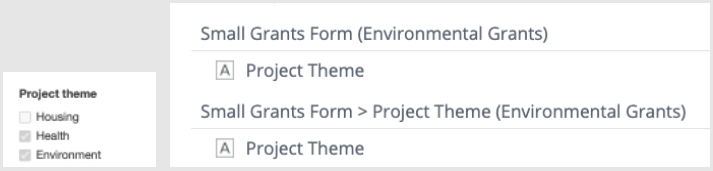

Page One
Section One
Project Title
Brief Project Description
Provide a short description (100 words recommended) of your project - what are you out to do?
Project theme
Housing
Health
Environment
Clear
Section Two
Maximise
Budget Items
Location
Date of Events
Must be a date.
Books
Pens
Paper
Other:
Clear
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

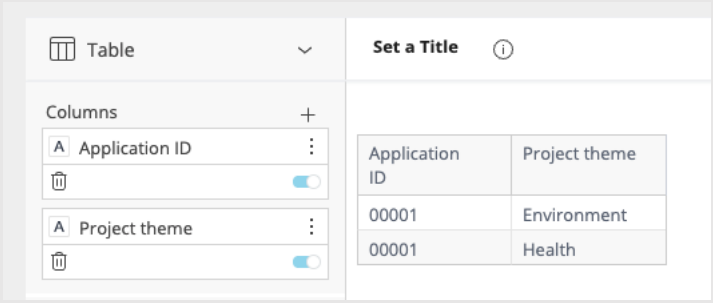
- Responses to “Project Title” and “Short Project Description” would be stored in the main table **Small Grants Form (Environmental Grants)** (the main table containing responses to single-response questions in non-repeatable sections)
- Responses to “Project Theme” would be stored in a separate table, **Small Grants Form > Project Theme (Environmental Grants)**. This is because “Project Theme” is a multi-choice question in a non-repeatable section. Note: 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?

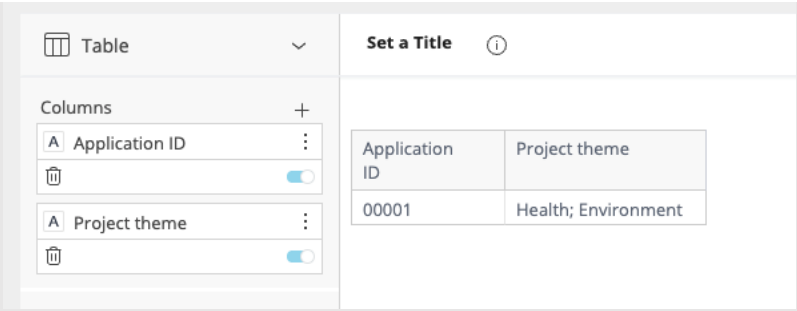
As seen in the screenshot below, the multi-choice question “Project Theme” appears both in the main Small Grants Form (Environmental Grants) table (storing single-response questions from non-repeatable sections) and in a stand-alone table.



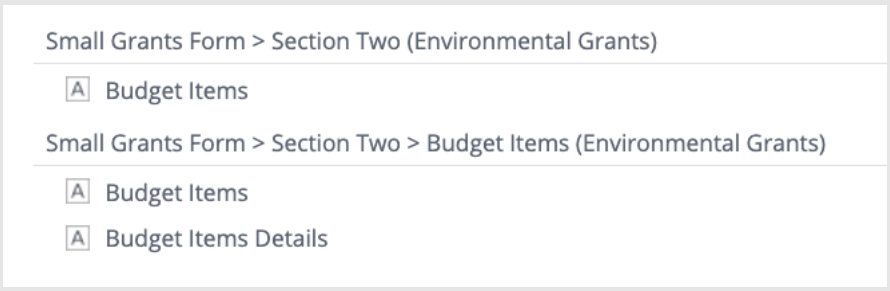
In the stand-alone table, every choice/answer is separated out into a separate row. See here for an example of how multi-choice questions where the option of “other” is enabled appear.



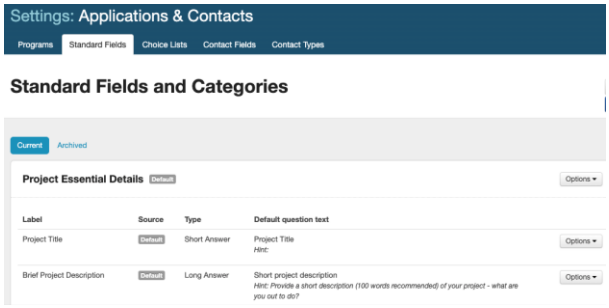
Responses are also stored in a concatenated form in the main table (storing single-response questions from non-repeating sections), where all choices are combined in a string separated by “;”.



Storing this response 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.



16 Standard Fields



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.

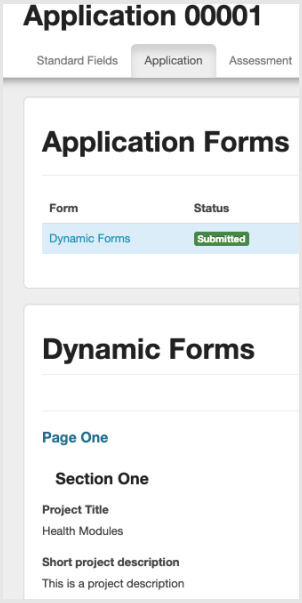
Please note archived standard fields are not included in reporting.

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?

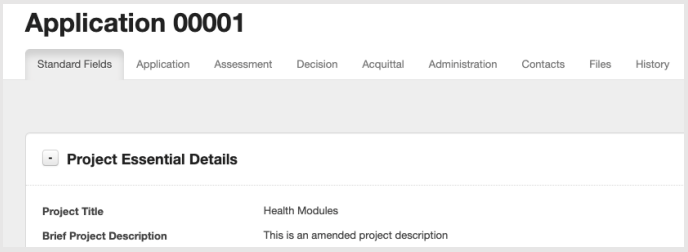
If a form contains a standard field, it will be stored both in a form table and in a standard field table. Think of the form table version as a "point in time" record of how someone filled out a standard field, while the standard field version captures the latest response to that standard field.

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

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.



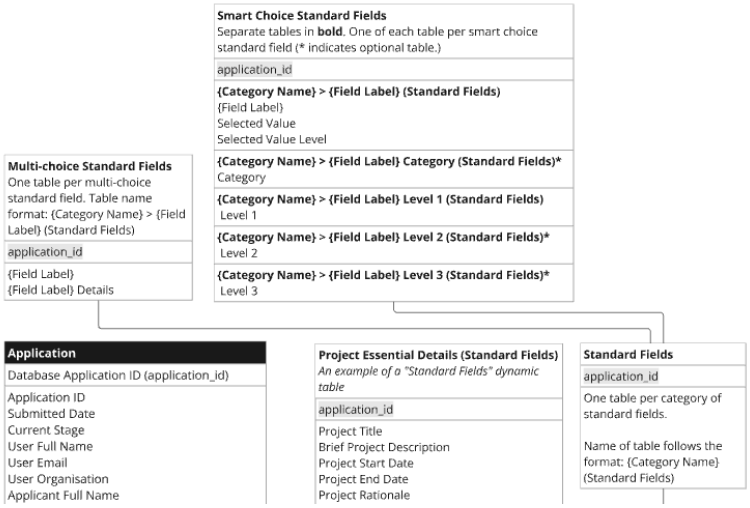
If an account administrator later enters the Applications tab of SmartyGrants and edits this applicant’s standard field response to “This is an amended project description” (or the applicant answers the same standard field differently in another form), the field in Project Essential Details (Standard 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.



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 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.)

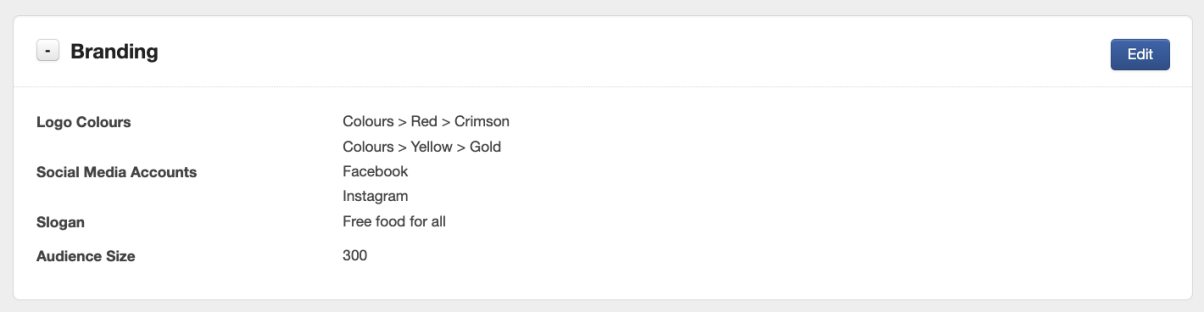


As an example, consider the following standard field category “Branding” which contains two single-response questions “Slogan” and “Audience Size”, a multi-choice question “Social Media Accounts” and a smart choice question “Logo Colours.”

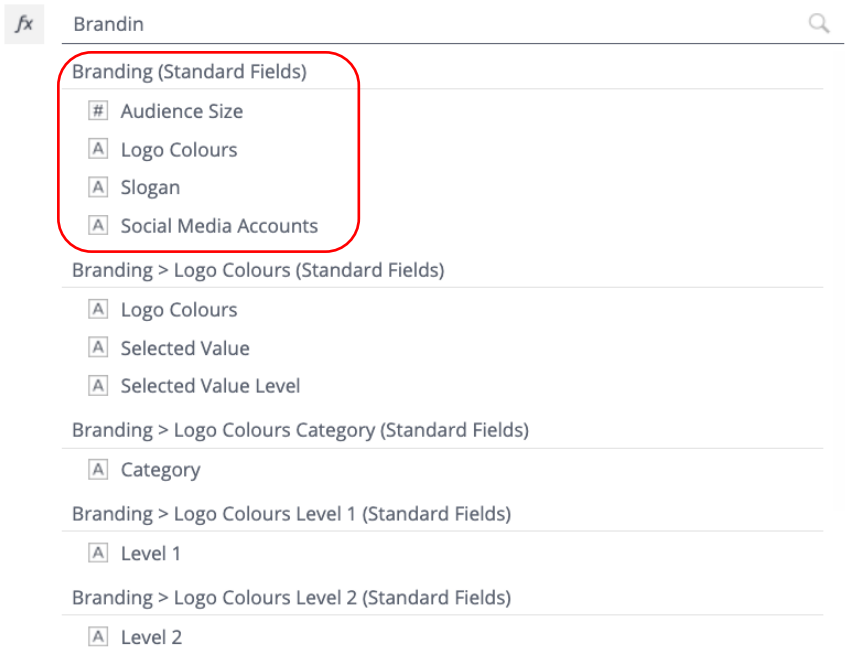
Branding Custom — Options

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

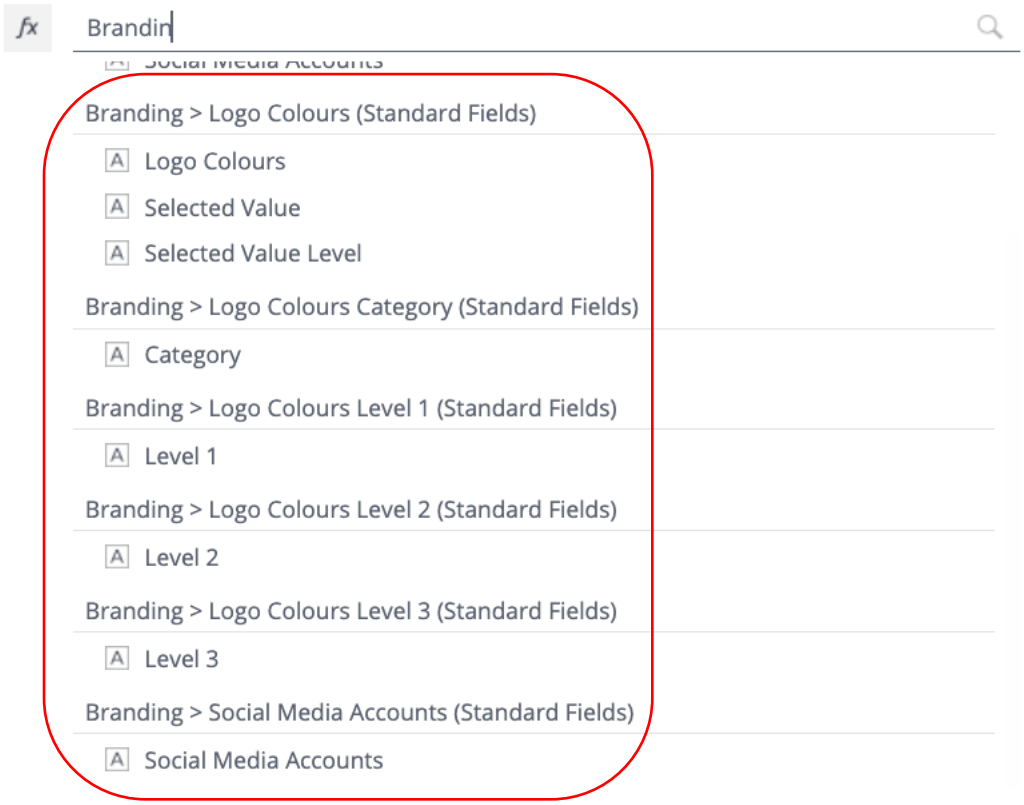
These standard fields have been filled out for the Application 0001.



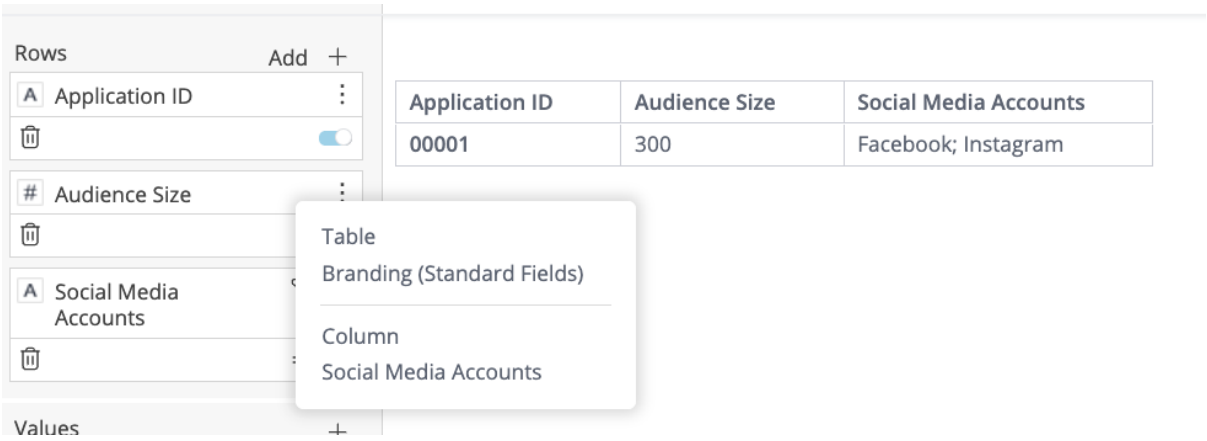
Responses to the two single-response questions “Audience Size” and “Slogan” are stored in the main table for the standard field category, Branding (Standard Fields). For every multi-choice and smart choice question in a standard field category, Branding (Standard Fields) also contains a corresponding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

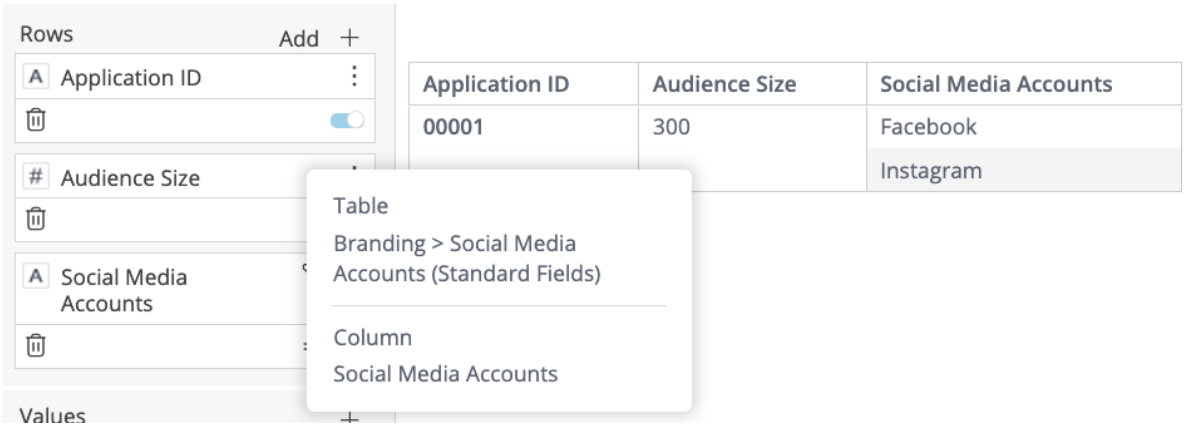


The multi-choice and smart-choice questions “Logo Colours” and Social Media Accounts” are stored in separate stand-alone tables.



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



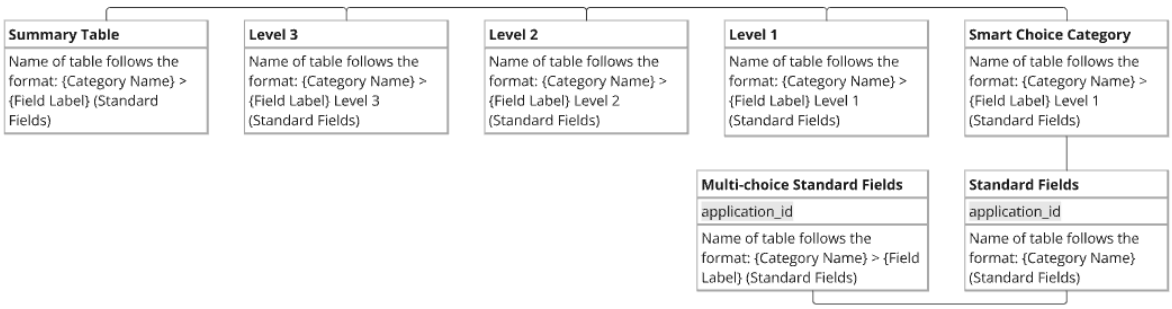


The next section goes into the various tables that are generated for a single smart-choice question.

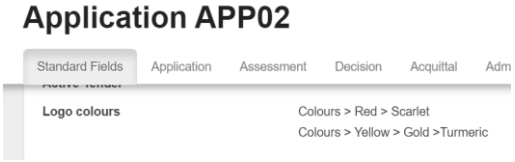
17 Smart Choice Standard Fields

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.



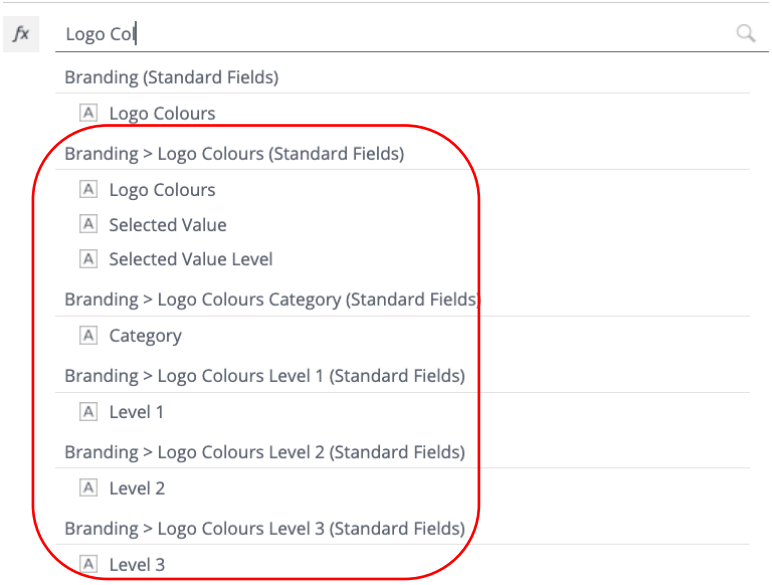
The easiest way to understand how smart choice fields are stored in the data model is through an example. An example of a smart choice response could be to the question "Logo Colours", which asks the user to select colours. Colours can be nested under umbrella colours.



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".

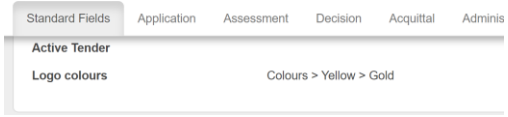
Every smart choice standard field dynamically generates a “summary” table that shows response values in their hierarchy (through a concatenated string), in addition to up to four “level” tables that split out that hierarchy.

- 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”).

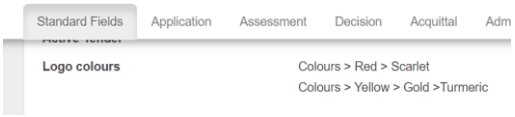


Consider two applications with the following standard field choices. APP02 selected the Level 3 choice “Turmeric” (also selecting a Level 2 choice “Scarlet”.) APP01 selected the Level 2 choice “Gold” and did not drill down to a level 3 selection.

Application APP01



Application APP02



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

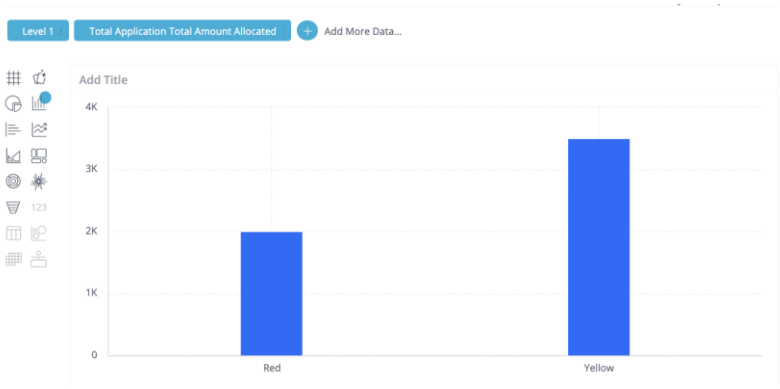
Fields from the level tables break down the layered selection. These tables have the naming structure {Standard Field Category} > {Question label} {Level} (Standard Fields). For example, APP02's selection of Colours > Yellow > Gold > Turmeric has:

- "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)

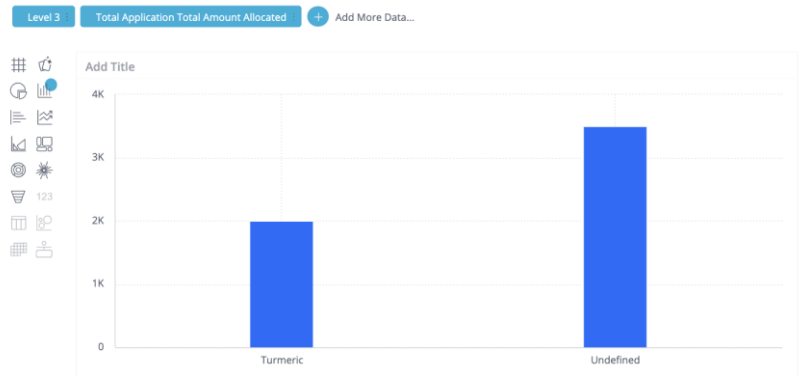
Application ID	Logo colours	Selected Value	Selected Value Level	Category	Level 1	Level 2	Level 3
APP01	Colours > Yellow > Gold	Gold	Level 2	Colours	Yellow	Gold	Undefined
APP02	Colours > Red > Scarlet	Scarlet	Level 2	Colours	Red	Scarlet	Undefined
	Colours > Yellow > Gold > Turmeric	Turmeric	Level 3	Colours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

Since APP01's selection 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.



The below chart uses "Level 3" from Branding > Logo Colours Level 3 (Standard Fields) to aggregate the amount allocated to applications that selected different Level 3 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.

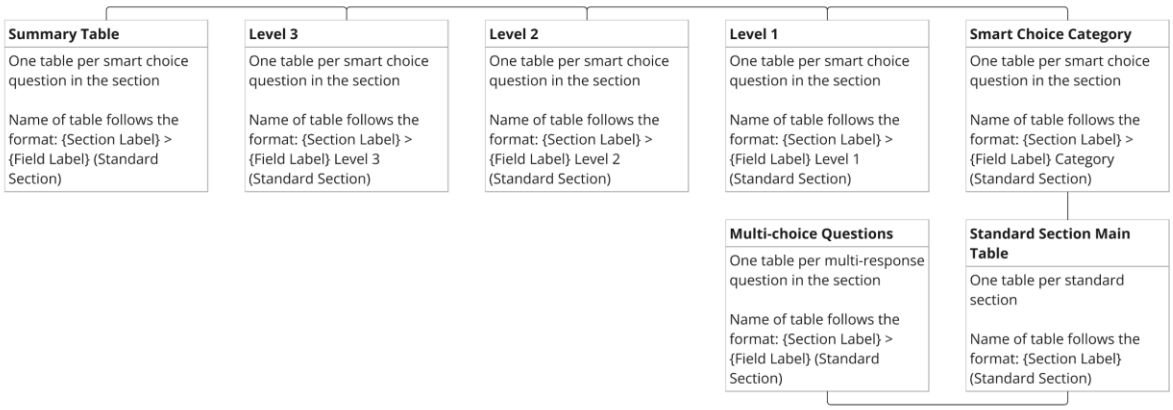
Inputting overlapping hierarchies into a smart choice template may lead to duplication (inflated figures) in the reporting tool. This duplication can be resolved through a calculated field in the widget builder, but orthodox hierarchies are encouraged to minimise unexpected results.

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.

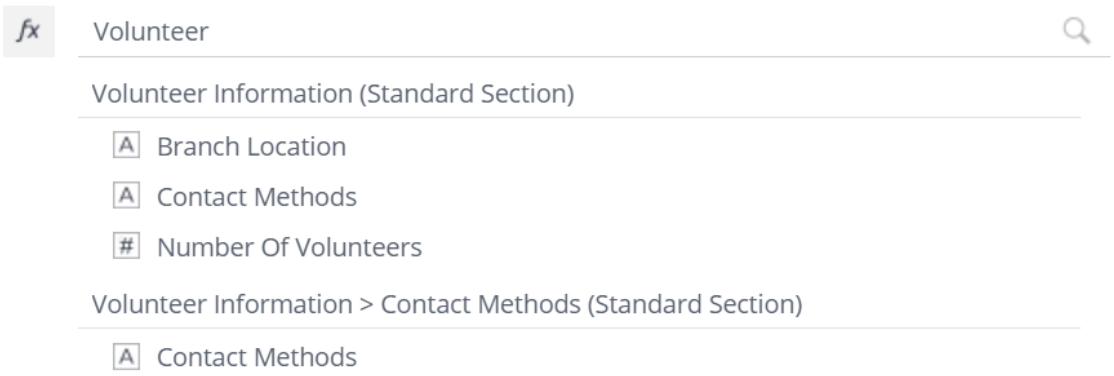
18 Standard Sections

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.



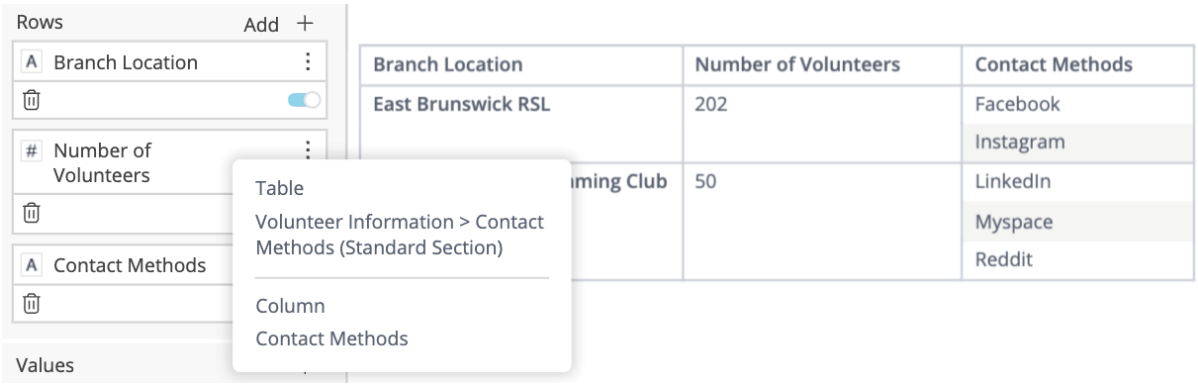
Consider the example of a standard section below “Volunteer Information,” which contains two single-response questions and one multi-choice question.



The two single-response questions in the section ("Branch Location" and "Number of Volunteers") would be 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 that row. Using the "Contact Methods" field from Volunteer Information (Standard Section) would look like:

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.



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:

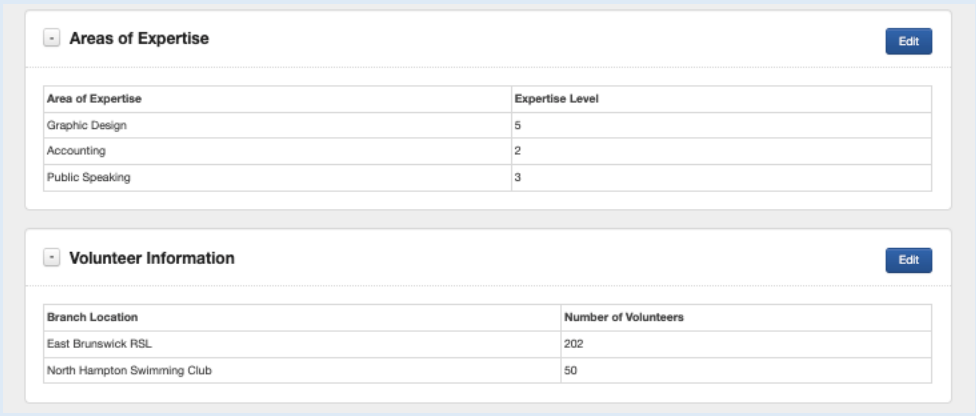
- Standard sections do not need to be turned on for reporting and are available in Analytics by default.
- Archived standard sections are not included in reporting.

Warning: be careful when combining fields from multiple sections (whether repeating 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 repeating section in a single widget.

If an application_id has multiple rows in one or all sections, the join on application_id creates all possible permutations of values for 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.)

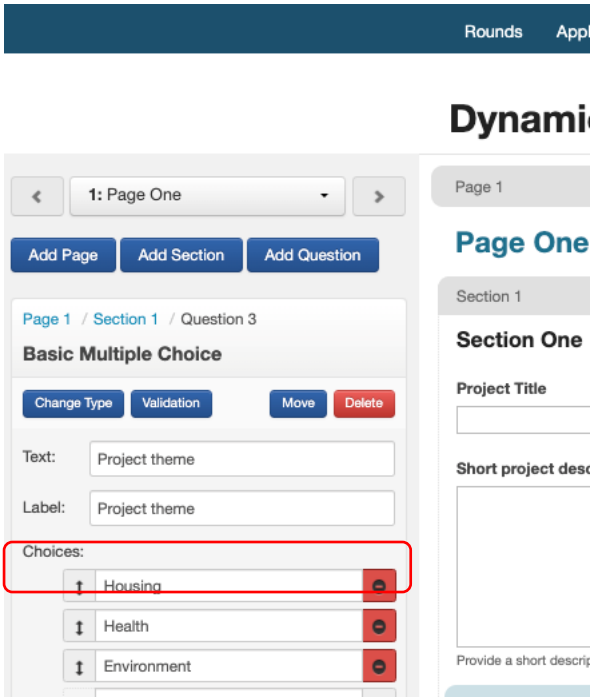
Consider the two example standard sections "Areas of Expertise" and "Volunteer Information."



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

19 Dynamic Tables: Troubleshooting/FAQs



Form FAQs

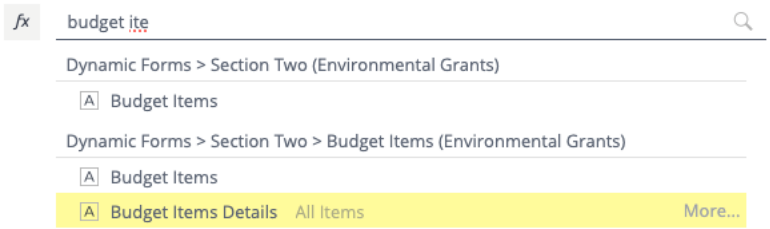
When building a widget, I can't find the field for a question in a form. What should I do?

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.

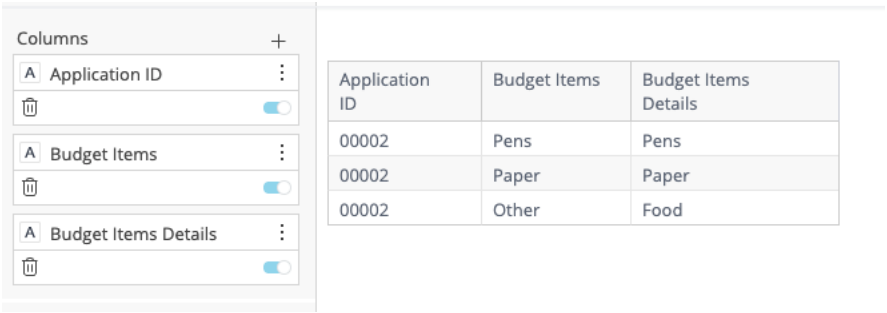
For fields from forms to be located, these labels should be intuitive to allow for easy searching. Once you have set your labels, you can simply type them into the field selector when building a widget to narrow the search.

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".



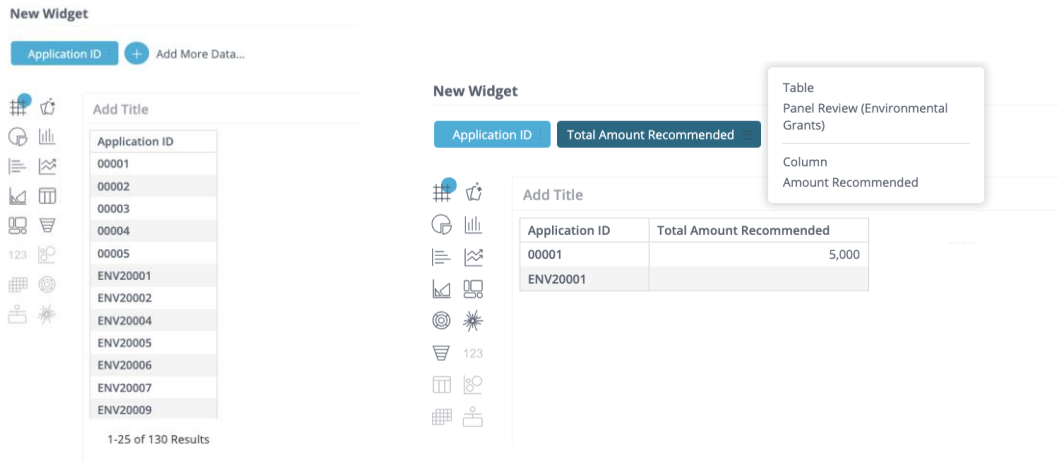
For responses where an applicant did not select "other", the choice is repeated in the "Details" column.



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.



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 field 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.

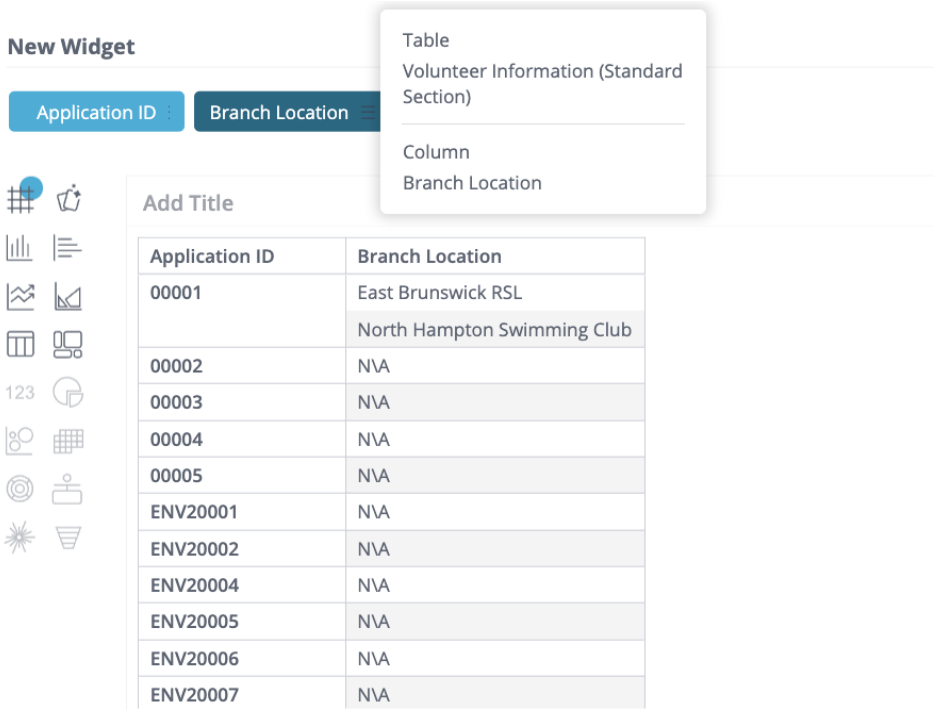
I combined fields from a standard section and a repeating section (from a form) in a widget and the widget does not look as expected (too many rows, duplicated values etc.) What happened?

It is not recommended to combine fields from a standard section and a repeating section (in a form) 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?)

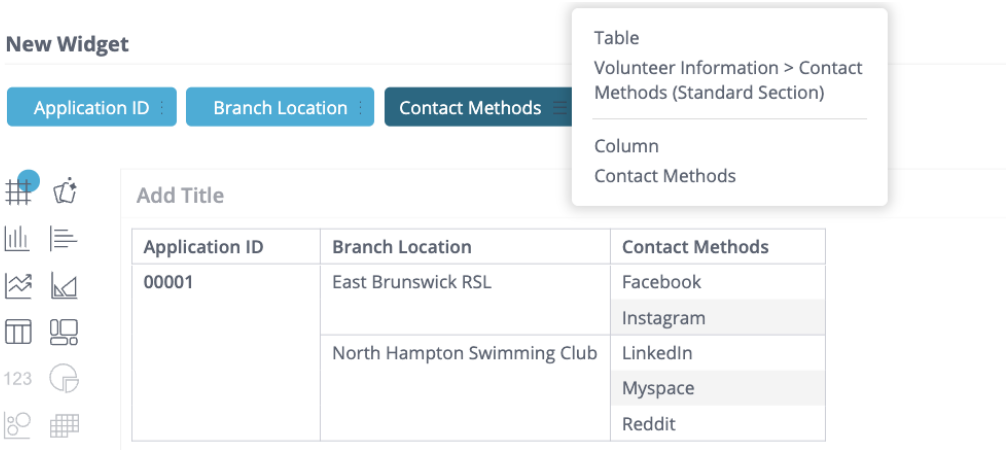
When using a field from a multi-choice or smart-choice table, the widget will automatically filter down to applications that have filled out the standard field/ filled out the standard section the question is from.

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.”



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” which is stored in a separate table is added to the widget, reducing the number of Application IDs appearing in the widget to those which have the “Volunteer Information” Standard Section filled out.



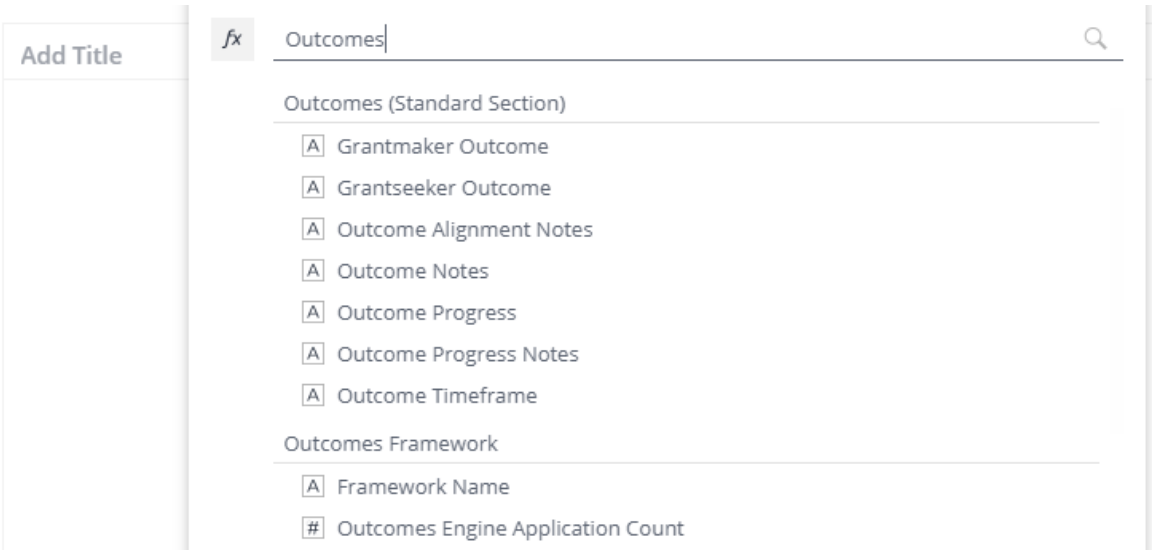
The filtering out of application IDs occurs because the multi-choice/smart-choice tables only contain data for applications with the standard field/standard section in question filled out (an inner join between these tables and other tables therefore excludes some Application IDs.)

20 Outcomes Engine Reporting

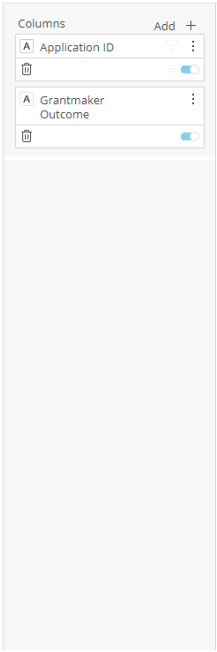
Tables related to the Outcomes Engine will only appear for instances which have the Outcomes Engine turned on. If you are not familiar with the Outcomes Engine and Outcomes Engine default standard sections, please see the Help Hub.

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.

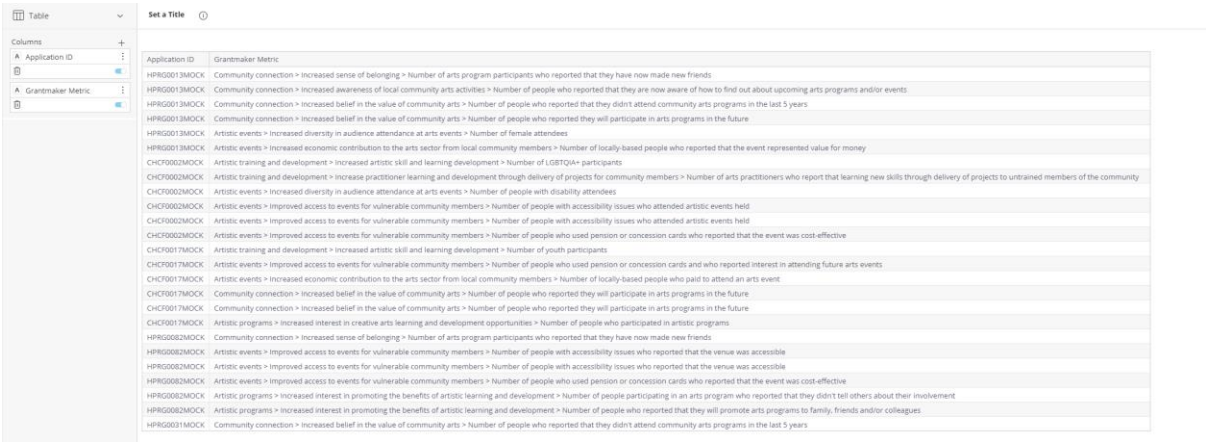


Grantmaker domain, outcome and metric selections stored in tables ending with "(Standard Section)" will be in a concatenated form following the structure domain > outcome > metric. In the example below, a value from "Grantmaker Outcome" in Outcomes (Standard Section) is "Community Connection > Increased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

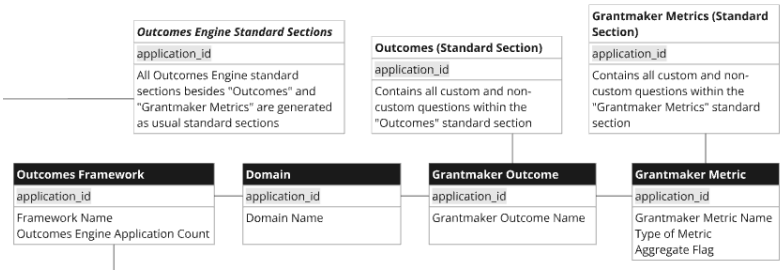
In the example below, a value from “Grantmaker Metric” in Grantmaker Metrics (Standard Section) is “Community connection > Increased sense of belonging > Number of arts program participants who reported that they have now made new friends.” “Community Connection” is a domain housing the outcome “Increased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



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:

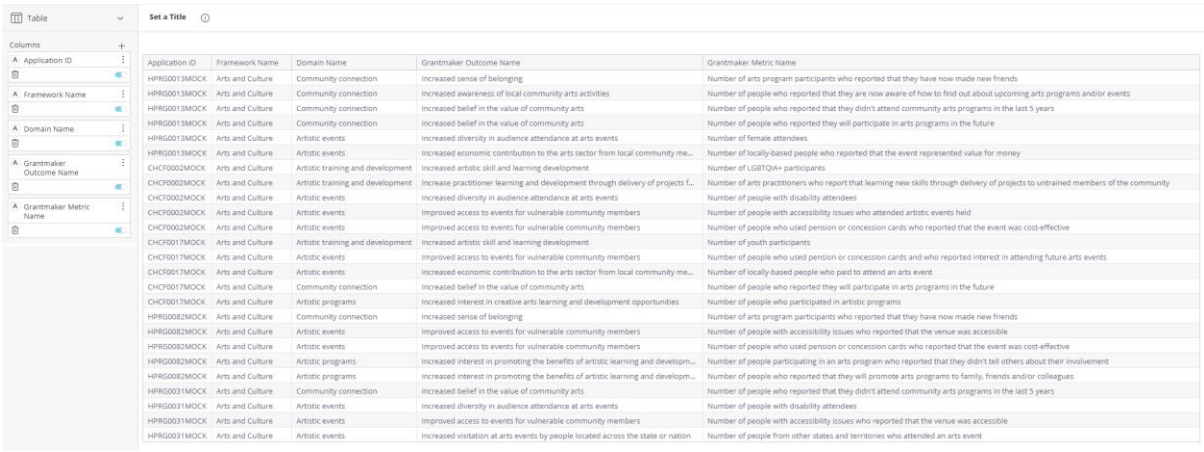
- Outcomes Framework (containing the field “Framework Name”)
- Domain (containing the field “Domain Name”)
- Grantmaker Outcome (containing the field “Grantmaker Outcome Name”)
- Grantmaker Metric (containing the field “Grantmaker Metric Name”)



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

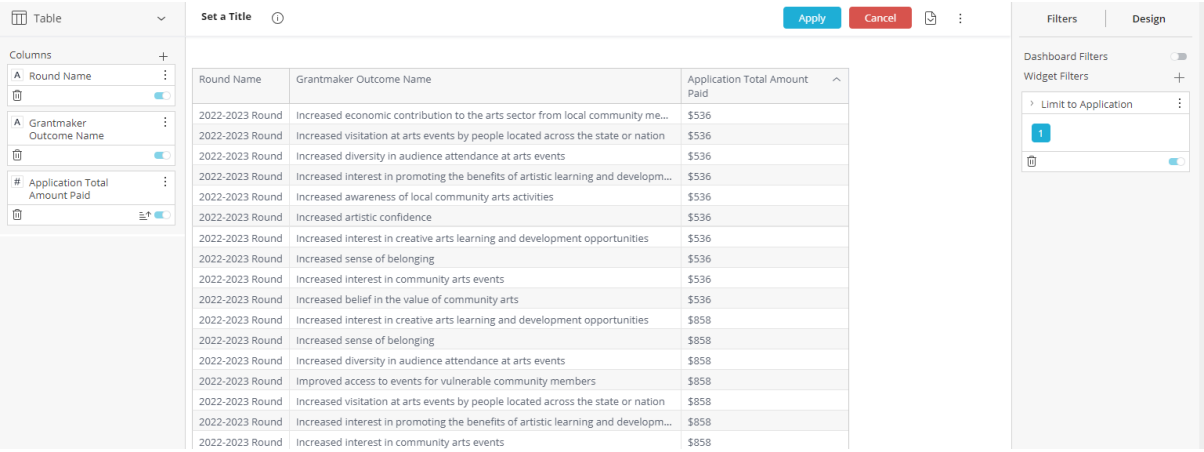
- Grantmaker Metric (Standard Section) stores a metric as “Community connection > Increased sense of belonging > Number of arts program participants who reported that they have now made new friends.”
- The static tables split the domain, outcome and metric components out into “Community connection” in 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 not linked to a grantmaker outcome in the “Outcome OE” standard section are linked to a domain “Not selected” and a grantmaker outcome “Not selected.”



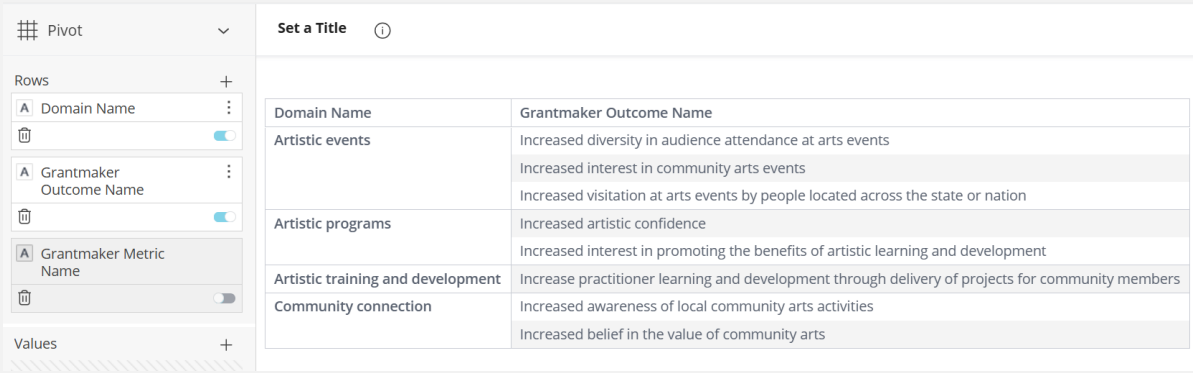
Application ID	Framework Name	Domain Name	Grantmaker Outcome Name	Grantmaker Metric Name
HPRG0013MOCK	Arts and Culture	Community connection	Increased sense of belonging	Number of arts program participants who reported that they have now made new friends
HPRG0013MOCK	Arts and Culture	Community connection	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.

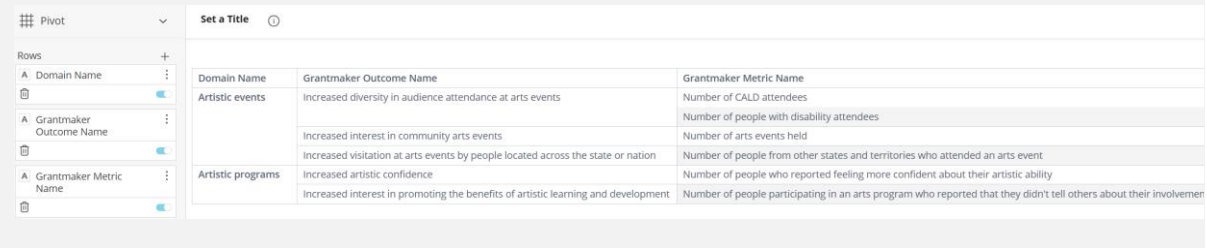


Warning: Using fields from "Grantmaker Metrics" in a widget will cause any domains/outcomes without any metrics/ whose metrics have not been selected to be omitted from the widget.

The widget below shows the domains "Artistic training and development" and "Community connection".



However, when the field "Grantmaker Metric Name" is added to the widget the outcomes from these two domains disappear. This occurs either because the outcomes framework did not include metrics for these domains/outcomes OR no applicants selected the metrics associated with these domains/outcomes.

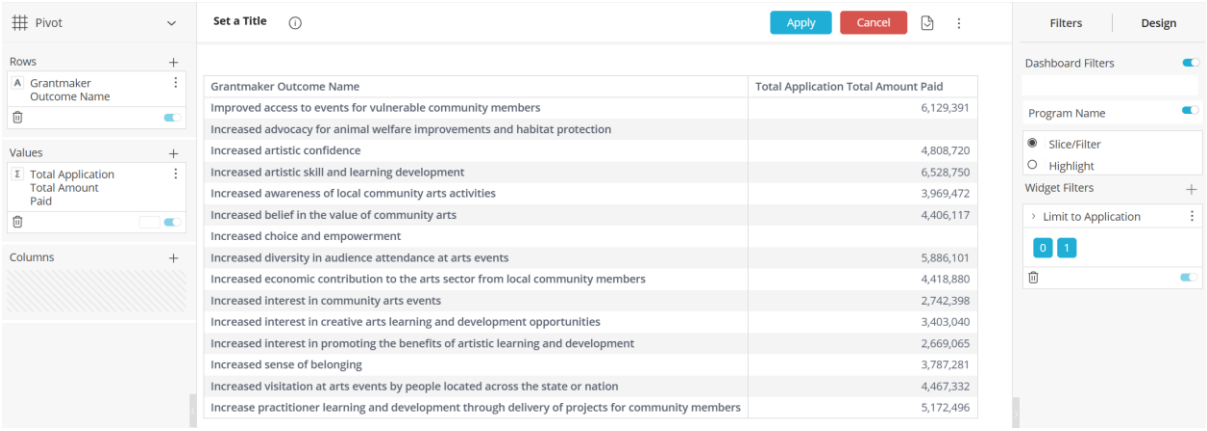


Besides domain/outcome/metric selections, the static tables contain several other fields to facilitate outcomes-related analysis. In general, it is possible to use fields from the static outcomes tables to group fields from any table which contains "Application ID" in its key or key combination. However, under certain conditions some of these fields should only be used with specific tables. Below contains a guide on using these field correctly.

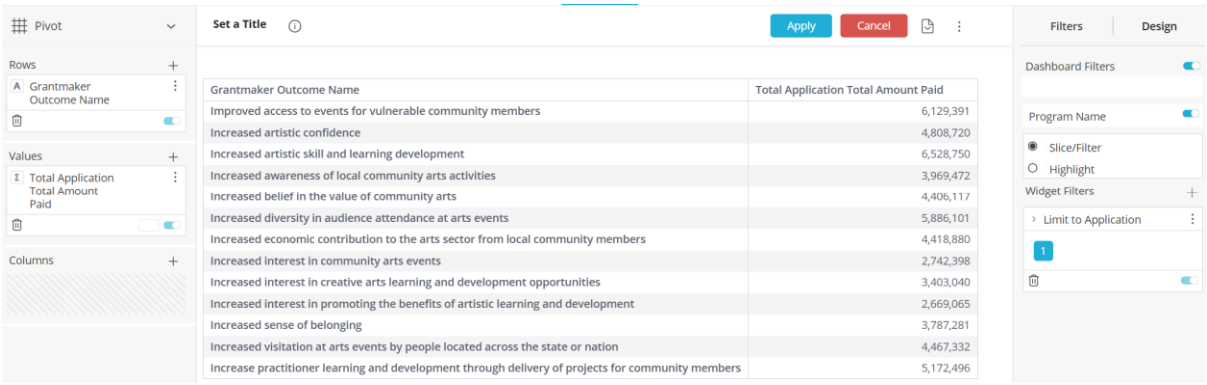
Negative analysis (unselected domains/outcomes/metrics)

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.



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.



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

In other words, it is possible to use the static outcomes tables to group fields from any table which contains “Application ID” in its key or key combination (e.g. Funding Allocation or Payments.) However, it is not possible to show unselected frameworks/domains/outcomes/metrics when grouping fields from Funding Allocation or Payments 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.

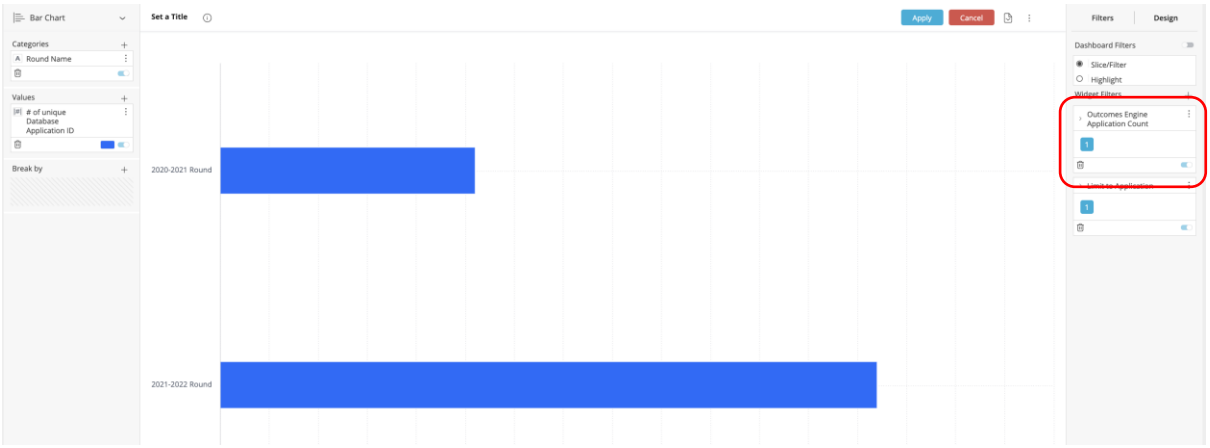
- This is because, in the background, negative analysis involves a placeholder application representing "unselected" whose id is set to "-1" (leading to a count of applications being inflated by one.)

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, multi-choice question tables, repeating section tables.) 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".



21 CLASSIE

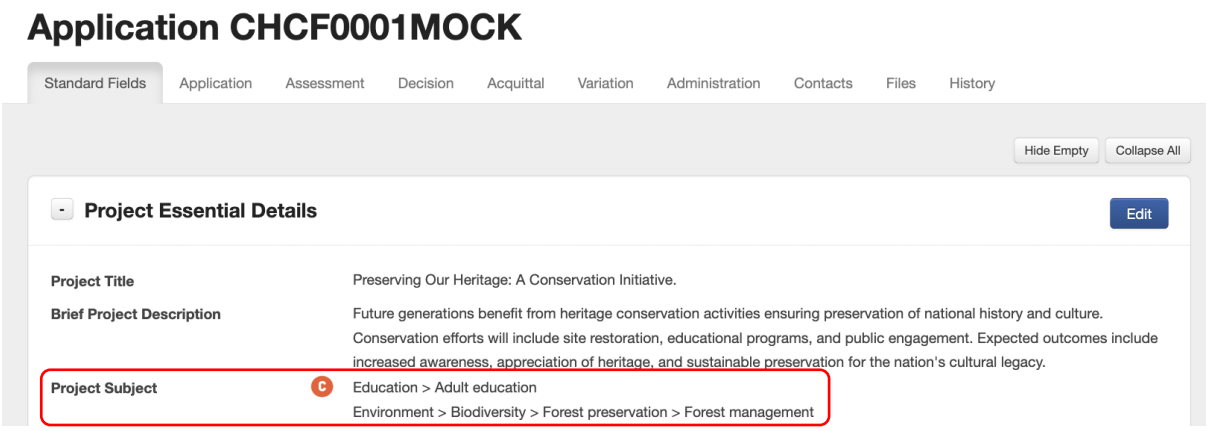
If you are not familiar with CLASSIE, which allows applicants to select a project subject/beneficiary from a set list where each choice may be grouped under or belong to a parent choice, please see the Help Hub.

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

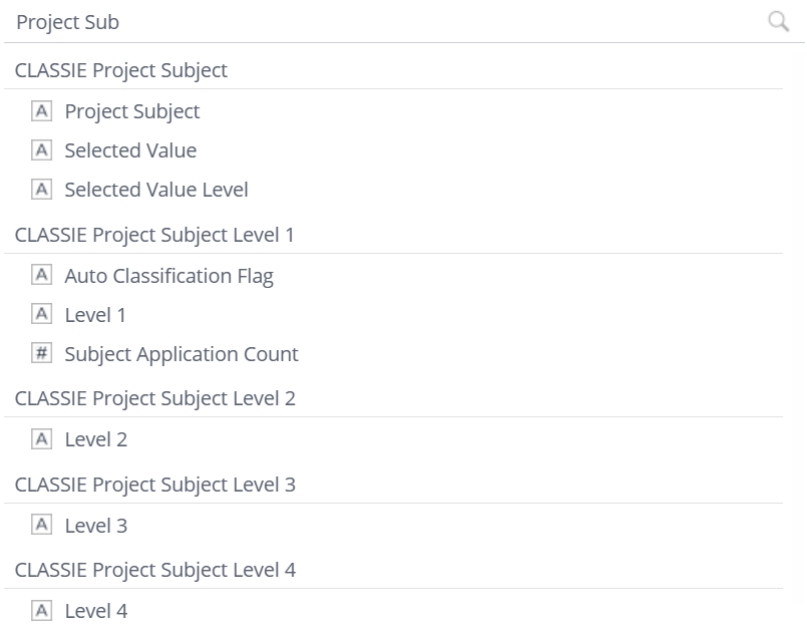
CLASSIE Project Subjects Separate tables in **bold**
application_id
CLASSIE Project Subject Project Subject Selected Value Selected Value Level
CLASSIE Project Subject Level 1 Level 1 Subject Application Count Autoclassification Flag
CLASSIE Project Subject Level 2 Level 2
CLASSIE Project Subject Level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

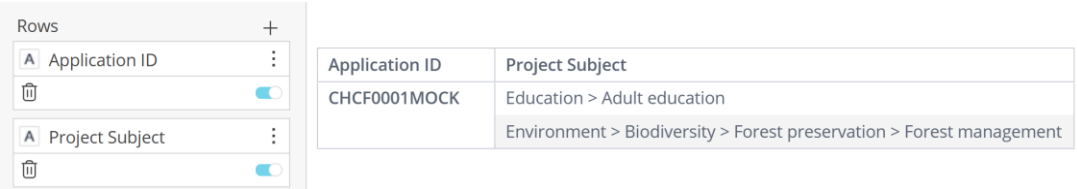


If an application selects their Project Subject as “Forest management”, this is a Level 4 choice with “Forest management” nested within a hierarchy “Environment > Biodiversity > Forest preservation > Forest management”. In that hierarchy, “Forest management” is nested under the Level 3 choice “Forest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



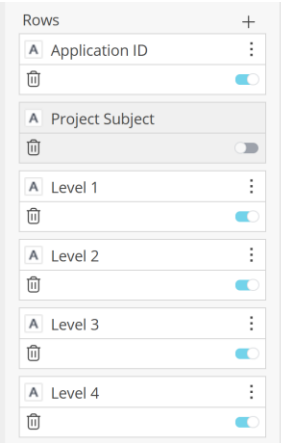
Below shows that selecting “Project Subject” from the CLASSIE Project Subject summary table shows selections in their layered hierarchy.



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

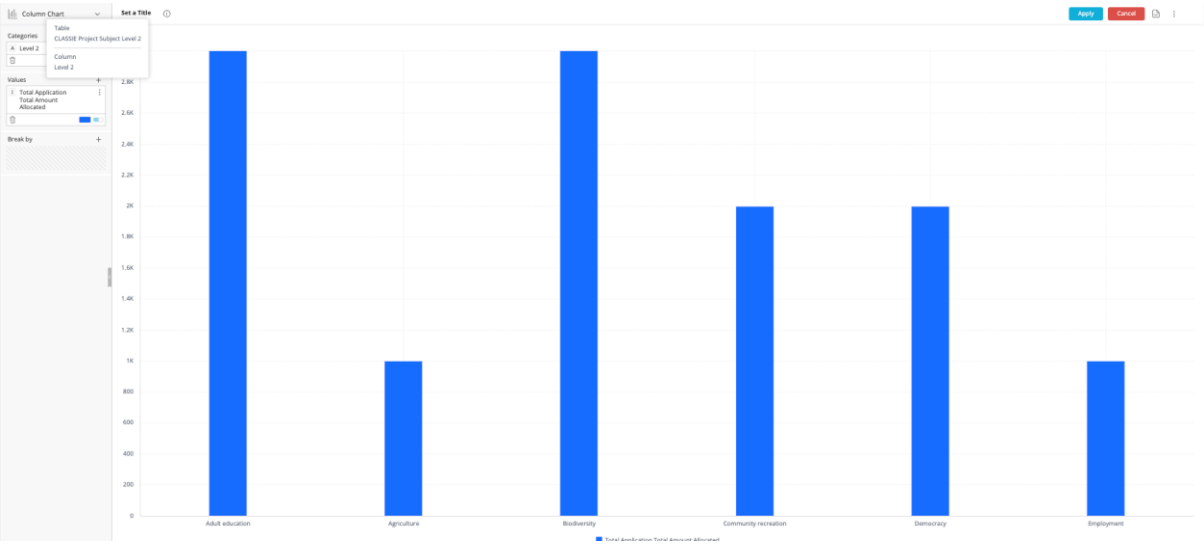
- “Forest management” for the field “Level 4” from CLASSIE Project Subject Level 4
- “Forest preservation” for the field “Level 3” from CLASSIE Project Subject Level 3
- “Biodiversity” for the field “Level 2” from CLASSIE Project Subject Level 2
- “Environment” for the field “Level 1” from CLASSIE Project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



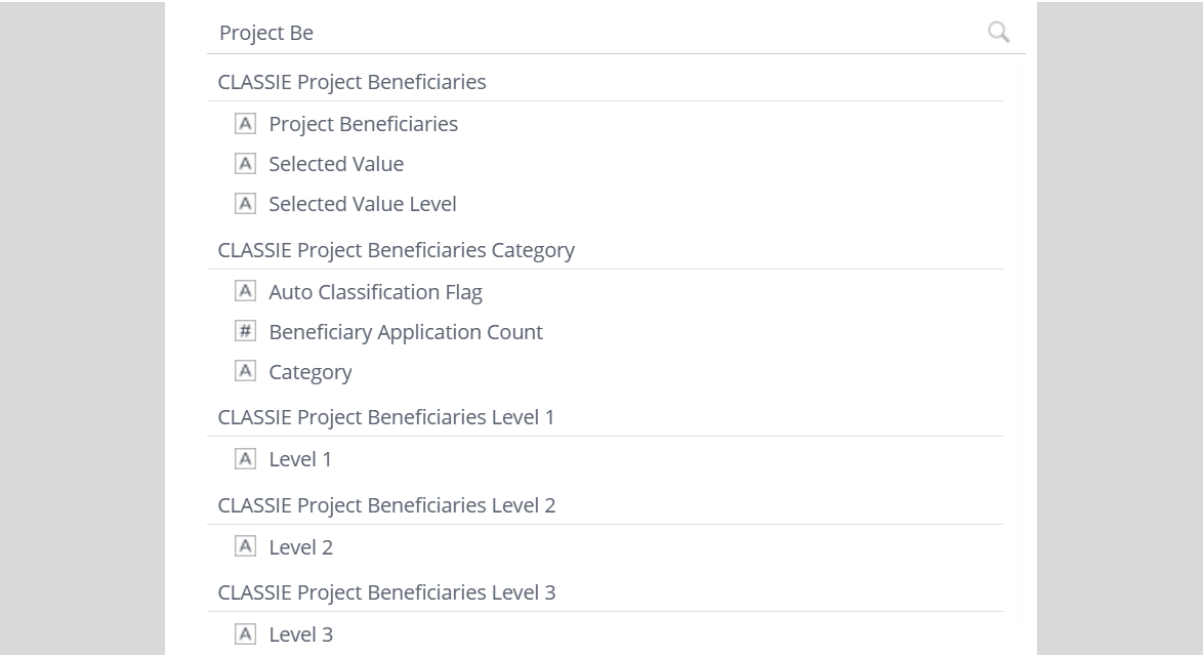
Application ID	Level 1	Level 2	Level 3	Level 4
CHCF0001MOCK	Education	Adult education	Undefined	Undefined
	Environment	Biodiversity	Forest preservation	Forest management

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.

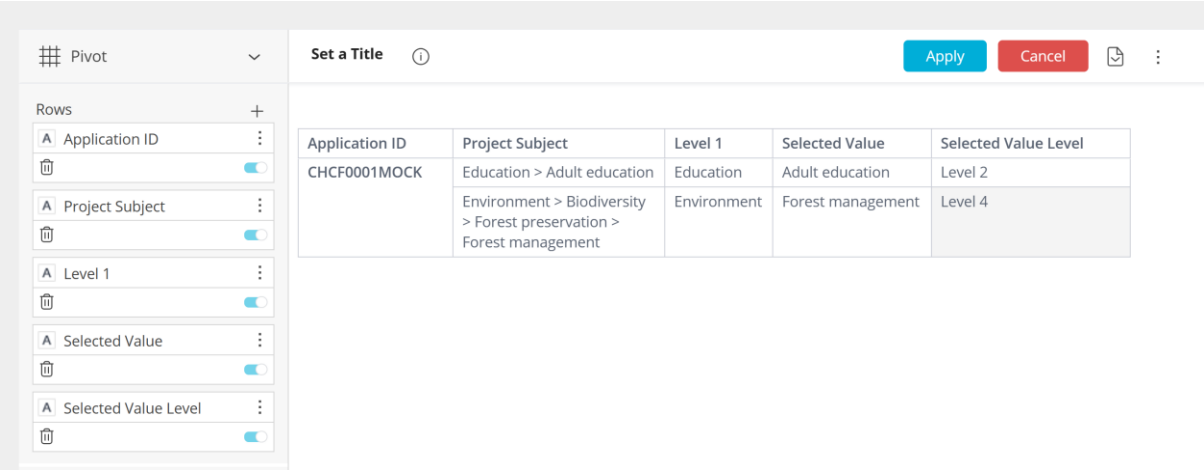


The summary CLASSIE Project Subject table and the CLASSIE Subject Level 1 table contain several other fields to facilitate analysis (they are split across these two reasons for data modelling purposes).

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.

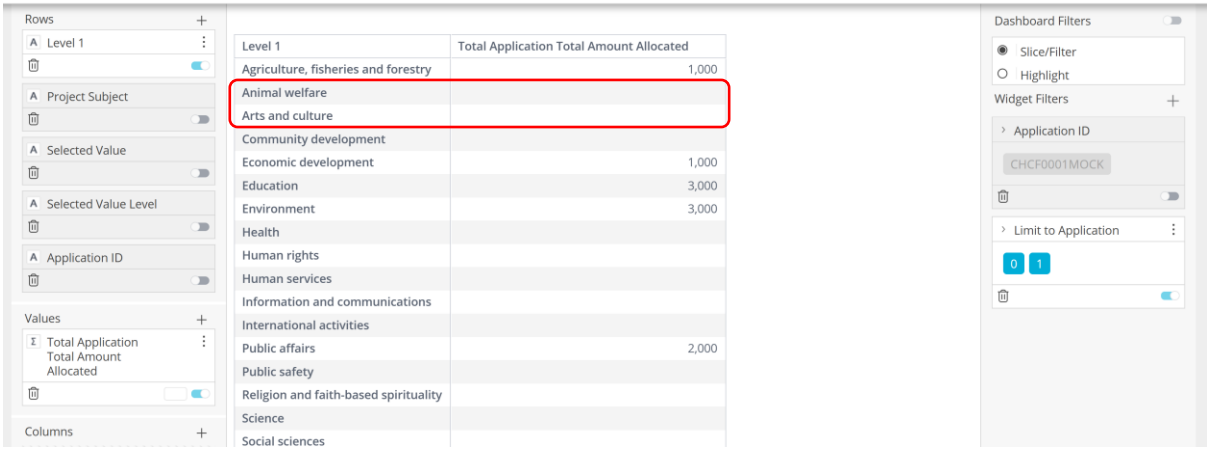
- For the example "Environment > Biodiversity > Forest preservation > Forest management", "Selected Value" gives "Forest Management" and "Selected Value Level" gives "Level 4".
- If another application selected "Environment > Biodiversity > Forest preservation", "Selected Value" would give "Forest preservation"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



Negative Analysis (unselected subjects/beneficiaries)

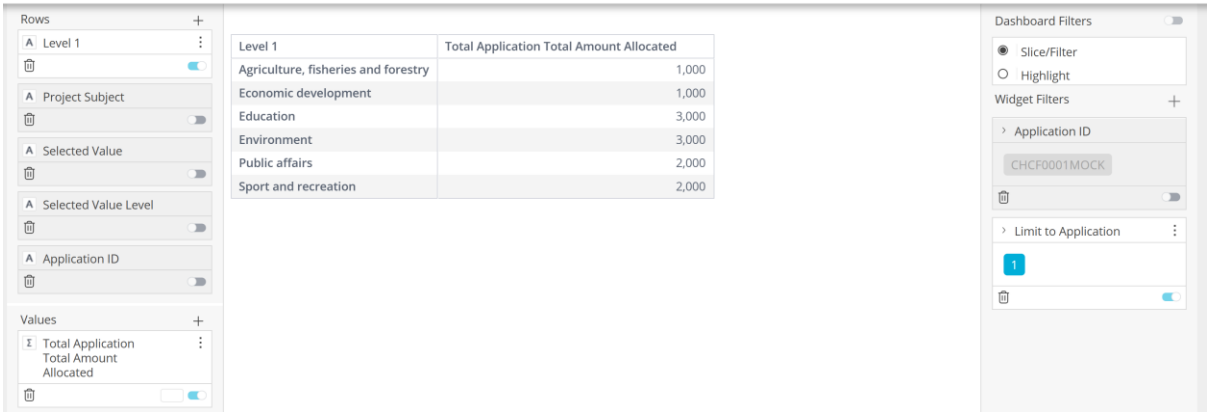
By default, the reporting tool will exclude subjects/beneficiaries which were added to a round but which have not been selected by any application. To include unselected subjects/beneficiaries in a widget, set “Limit to Application” to equal “0” or “1” (instead of the usual default “1”). In the widget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

Analytics



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



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

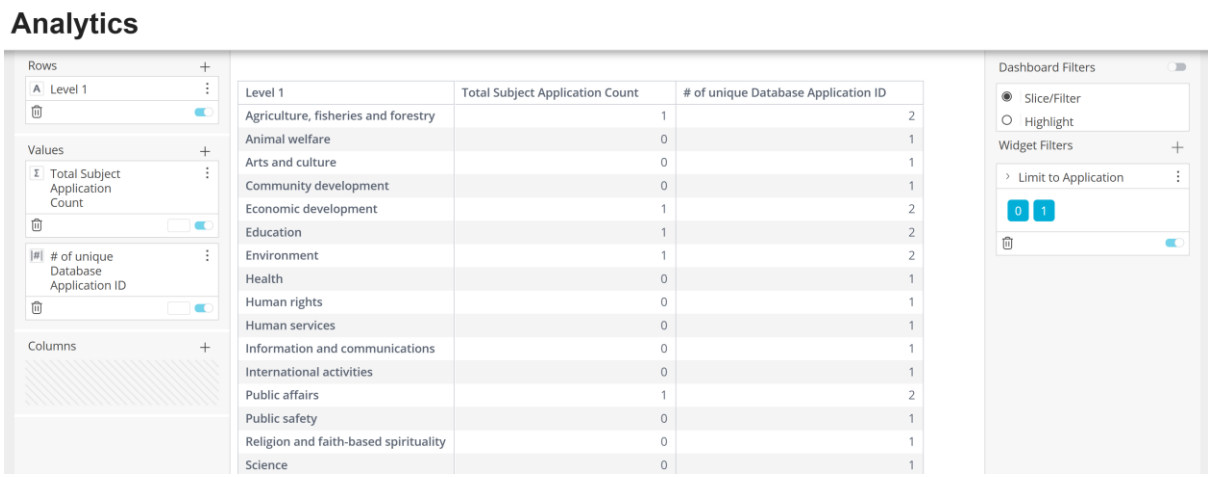
In other words, it is possible to use CLASSIE tables to group fields from any table which contains “Application ID” in its key or key combination (e.g. Funding Allocation or Payments.) However, it is not possible to show unselected projects/beneficiaries with fields from Funding Allocation or Payments 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 field "Subject Application Count"/ "Beneficiary Application Count".

The field "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, however, as described in below, 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.)



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, multi-choice question tables, repeating section tables.) 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