Chase Africa Data – Data Repository Setup Instructions

Maja

30.1.2019

Intro:

This document contains the step-by-step instructions for how the master data repository was set up to combine existing and future clinic level data collected by Chase Africa's partners.

These instructions have been tested by Maja but will be implemented by Catherine on the actual data in order to develop and keep the knowledge in-house.

This setup will only be completed once, but the information contained here should be archived as part of the Data Management Plan.

Outline

These instructions cover the following:

1. Description of the data
2. Preparation of the data
3. Merger of the data
4. Maintenance of the data

Describing the data

To begin with we should have three tables:

1. `responses`: the responses from the Google form.
2. `migration`: the raw data that was input from the Excel files.
3. `manual`: the raw data that will be input from the partners in the future if they don't use Google forms.

(obviously you can rename them any way you see fit, alhtough I would recommend you don't use any spaces)

1. `responses` table

This table is the direct output of the Google form. This means the order of the columns and the names of the variables are defined automatically, so we'll stick with them as the baseline.

For easier reference in this document I will refer to these n variables as `var-1`:`var-n`: these are core variables that will continue to be collected from the partners.

The number of rows in this table keeps growing automatically with every new filled out form.

2. `migration` table

This table was created (as a one off) manual migration of the data contained in 6 Excel spreadsheets.

The table contains all the variables `var-1`:`var-n` as well as some additional *legacy* variables which are no longer used, these are referred here as `old-1`:`old:n`. The order of the columns and variable names were ad hoc, but they will now need to be adapted to conform with the `responses` table.

This table has 893 rows and that is it, and will never have any more.

3. `manual` table

This table will start out as an empty copy of the `responses` table and will be used to manually enter data from partners who continue to report their data using Excel instead of using Google forms.

It therefore contains variables `var-1`:`var-n` in the same order as the `responses` table.

The number of rows increases manually with every new entry by Catherine.

Preparing the data

All three tables should be placed into a single Google Sheet workbook, each on it's own sheet, named appropriately.

1. Preparing the `migration` table

Re-order the columns so they have the following order: `var-1`:`var-n` and then `old-1`:`old:n`. So the first columns must be exactly in the same order as the ones in the `responses` table.

- This means moving the other ones to the back i.e. right of the table.
- Don't forget to add the ones we discussed 21.1.: the flag for estimated data and the flag for multiple day clinics.
- This also means rethinking the `fund_round` or `fund_quarter` variable, because they don't currently exist in the `responses` table. It would probably be best to include it there, possibly with a dropdown: Q1, Q2..
- Don't worry about the variable names, you can keep them as they are, as long as you know they are paired up correctly with the ones from the `responses` table.
- You can use the Copy/Paste transposed function to easily compare the headers of both sheets side by side.

This table will also get some *protection*. The data here should never be manipulated manually. To avoid accidentally corrupting the data you need to protect the range. There are two ways you can restrict permissions:

- show a warning when someone tries to edit the range
- restrict who can edit by entering their email.

I would only suggest using the second option if you can use an email that is not normally used to access the sheet. In other words this *administrator* should be a different email, not your customary one, because otherwise it offers no protection at all, as it would essentially be saying "only the only person who has access can change the content". In that case it makes more sense to use the warning instead.

Highlight the whole sheet, right click on it, select `Protect range`, click `Set permissions` and pick your choice.

2. Preparing the `responses` table

In order for the merge to work, this table has to have the same number of columns as the `migration` table. Just empty columns, that's all, but they have to exist, so you have to manually insert them.

Then apply the same protection as above.

3. Preparing the `manual` table

To initialise this table simply copy the header row from the `responses` table into a new sheet, and make sure you have the same number of columns (including the empty ones on the right).

This table does not need any protection, but you could still add some just to be on the safe side.

Merging the data

Now open a new `master` spreadsheet that will contain all the merged data. The data will be linked from the three tables described above, and this link is *live* i.e. any changes in the tables will update automatically in the master spreadsheet.

You will link the data using the `importrange()` function - which allows you to connect to a different spreadsheet and import a range of cells from there.

This linking is straightforward if you only want to import one table. But we have three tables that change in size. That's why we also need to use the `query()` function - which is a very powerful function to conditionally select subsets of the data. But in our case we will only use it to select the correct number of non-empty rows from each table.

The first time you use `importrange()` you must explicitly grant permission to pull data from another sheet. You only have to do this once. But because you will be using a combination of query and importrange to import from three tables, this granting of permission somehow doesn't work... That's why you have to first just do a simple importrange, grant permission, and then delete it. Once the permission is granted the access stays open, so this is a one off.

Granting `master` permission to access the data

1. Find the spreadsheet key or unique identifier of the Google Sheets spreadsheet with all the data. You can find it in the url of the spreadsheet it looks similar to this: "1SANV-egy4TBY6cxrzaayGslmb9TUMEVVLdBYvj2cKO9o".
2. Now open your new `master` spreadsheet. In order to connect the spreadsheets we will use importrange to import a little subset of the responses table, just the cells A1:D5:

```
= IMPORTRANGE("1SANVegy4TBY6cxrzaayGslmb9TUMEVVLdBYvj2cKO9o", "responses!A1:D5")
```

3. As soon as you press Enter you should get a `#REF` error that also says "you need to connect these sheets" if you hover over it. Click on the blue tag that says "Allow access".
4. Now the spreadsheets are connected. You should see the cells copied over. If it has worked properly you can delete the content of the cell A1.

Importing the data from all three tables

1. Again in cell A1 you will now enter a more elaborate command that combines:

- one *query()* and
- three calls to *importrange()*

We'll deal with the range later, first the query should look like this:

= QUERY({... *RANGE*...}, "where Col1 is not null")

This takes everything in the {... *RANGE*...} and selects all the rows that are not empty (where the first column is not null).

Now the {... *RANGE*...}: This will simply be three `IMPORTRANGE()` calls separated by `;` and surrounded by `{ }`. All the imported tables have to be the same width! Because one of our tables is larger than the others–the migration table has more columns–we have to make sure that we use the widest definition, and of course we have to make sure the other two tables have that width, even if they are empty columns.

So the full formula to write into cell A1 of the master spreadsheet should look something like this:

```
=QUERY({IMPORTRANGE("1SANVegy4TBY6cxrzaayGslmb9TUMEVVLdBYvj2cKO9o", "migration!A:BD");
IMPORTRANGE("1SANVegy4TBY6cxrzaayGslmb9TUMEVVLdBYvj2cKO9o", "responses!A:BD");
IMPORTRANGE("1SANVegy4TBY6cxrzaayGslmb9TUMEVVLdBYvj2cKO9o", "manual!A:BD");},
"where Col1 is not null")
```

Of course you have to replace the correct spreadsheet key and make sure the columns are correct as well (here they go up to `BD`).

Now select all of the columns that have been imported and protect that range (only those columns, don't protect the whole worksheet).

And you're done!

Maintaining the data

Once this is complete, the ongoing maintenance of this setup should be as follows:

1. The `migration` table should never be touched.
2. The `responses` table should also never be touched. It will automatically accrue new rows as each new form is filled out. Of course there will inevitably be errors where you might want to manually go in and fix them. I have also changed the settings for the form so that respondents can "edit after they submit". However I'm not sure how this works after you've closed the tab, or started entering the next one, but still.
3. The `manual` spreadsheet is the one that you will have to manually enter the data received from Excel tables. This is the only table in this spreadsheet that should be touched. You could also move it into a separate spreadsheet, that way there is less danger of accidentally interfering with the data. You would just have to get the key for a different spreadsheet when you use `importrange()`.
4. The `master` spreadsheet will update automatically when anything changes in the first three tables. None of those columns should ever be touched. But we will add more columns later on the right with formulas to calculate various things from the data, so those columns are of course not off limits..
5. The master spreadsheet will be the main data source for the Google Data Studio dashboard. This is actually probably also the place for the funding spreadsheet.