

Help - Designing forms - Sample forms

Below, please find a series of sample forms that demonstrate commonly-used techniques.

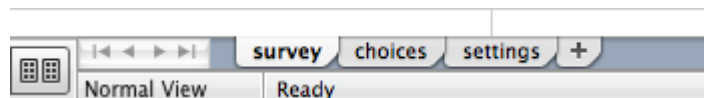
To try a sample form out – or to use it as a starting-point for one of your own forms – just go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*. You can then enable *Use a sample form as your starting point* and choose which sample to use.

Since sample forms are most helpful for those directly editing spreadsheet form definitions, the below help topics presume that you're working directly with spreadsheets. If you're using the designer, you can still learn from these samples – but everything will just be a bit easier.

The basics: Common elements in any form

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *The basics: Common elements in any form* from the list. You can also click [here](#) to download the spreadsheet form definition.

The spreadsheet form definition includes three worksheets: *survey*, *choices*, and *settings*:



Starting with the *survey* sheet, you will see that it begins with a set of hidden fields that will be automatically filled in with information from the survey device:

| type | name |
|--------------|----------------|
| start | starttime |
| end | endtime |
| deviceid | deviceid |
| subscriberid | subscriberid |
| simserial | simid |
| phonenum | devicephonenum |

These fields never show up to users filling out surveys, but they provide useful data for tracking and back-end analysis. (In the form designer, all of these fields are hidden in *Form settings*.)

The next field is a *note* field which does show up to users but does not prompt for any input:

| type | name | label |
|------|-----------|---------------------------------------------------------------|
| note | intronote | Welcome to the sample form. Please swipe forward to continue. |

This is how that note will appear to users in *SurveyCTO Collect*:



The next field is a multiple choice field, which uses the *select_one* field type:

| type | name | label |
|------------------|---------|-----------------------------|
| select_one yesno | consent | Would you like to continue? |

The answer choices to this multiple choice question are listed on the *choices* sheet:

| list_name | name | label |
|-----------|------|-------|
| yesno | 1 | Yes |
| yesno | 0 | No |

For each option in the "yesno" list, there is a *label* (which will appear to users) and a *name* (which is the associated value that will appear in the data). This is how this multiple choice question will appear to users:



If the user answers "No" to the above question, then the survey ends. This is accomplished by grouping all other survey fields into a single group and setting this group to only be *relevant* when the user answers "Yes":

| type | name | label | relevance |
|------------------|-----------|-----------------------------|---------------|
| select_one yesno | consent | Would you like to continue? | |
| begin group | consented | Sample survey module | \${consent}=1 |
| ... | | | |
| end group | consented | | |

In this case, the *relevance* column was used to create a skip pattern.

Inside the group of questions, the first field is a *text* field:

| type | name | label |
|------|------|--------------------|
| text | name | What is your name? |

This is how the question will appear on an Android device:

Collect > Sample SurveyCTO Form

What is your name?

The next field is an *integer* field:

| type | name | label |
|---------|------|-------------------|
| integer | age | What is your age? |

And this is how it will appear to users:

Collect > Sample SurveyCTO Form

How old are you?

Finally, this form integrates dynamic content into the survey. The label for the last field incorporates answers from earlier in the survey:

| type | name | label |
|------|-------------|-----------------------------------------------------------|
| note | confirmnote | Your name is \${name} and your age is \${age}. Thank you. |

In this case, the note seen by the user will integrate the survey respondent's name and age because we used the `${fieldname}` syntax to refer to those fields. (And if *SurveyCTO Collect* is configured to *hyperlink field references*, users will be able to click on the name or age to jump back and change them.)

Languages: Including translations

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Languages: Including translations* from the list. You can also click [here](#) to download the spreadsheet form definition.

When you open the spreadsheet form definition for this sample, you will notice two columns highlighted in orange: *label:spanish* and *constraint message:spanish*. Both of these columns are on the far right of the form, so you will probably need to scroll right to reach them. They have been added to the "basics" sample form in order to supply Spanish-language translations:

| type | name | label | label:spanish | constraint message:spanish |
|---------|------|--------------------|----------------------|-------------------------------------------------------|
| text | name | What is your name? | ¿Cuál es su nombre? | |
| integer | age | How old are you? | ¿Cuántos años tiene? | Por favor, introduzca una edad válida para continuar. |

You will also find translations on the *choices* worksheet. A column has been added there to provide Spanish versions of the multiple-choice labels:

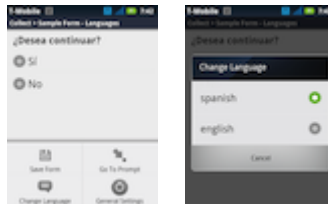
| list_name | name | label | image | label:spanish |
|-----------|------|-------|-------|---------------|
| yesno | 1 | Yes | | Sí |
| yesno | 0 | No | | No |

Finally, on the *settings* worksheet, you will notice a column for *default_language*:

| form_title | form_id | version | default_language |
|-------------------------|-----------------|------------|------------------|
| Sample Form - Languages | samplelanguages | 2013030512 | english |

Because the text in the default *label* columns is in English, the default language is set to "english".

From any page of the survey (including the opening screen), users filling out this form will be able to click *Change language* and then choose whether they want to see English or Spanish. (On Android, they click their device's menu button to find the *Change language* option; on the web, it's in the *Options* menu.)



Randomization: Randomizing form elements

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Randomization: Randomizing form elements* from the list. You can also click [here](#) to download the spreadsheet form definition.

When you open the spreadsheet form definition, you will notice two columns highlighted in orange: *relevance* and *calculation*. Both of these columns are to the right of the form, so you may need to scroll right to reach them.

This sample uses two independent random draws to independently randomize the phrasing of two separate questions. The first looks like this:

| type | name | label | relevance | calculation |
|------|------|-------|-----------|-------------|
|------|------|-------|-----------|-------------|

| | | | | |
|---------------------|-------------|-----------------------------------|--------------------------------------------------------------------------------|----------------|
| calculate | randomdraw1 | | | once(random()) |
| select_one yesno | likesblue1 | Do you like the color blue? | $\text{\$}\{\text{consent}\}=1$ and $\text{\$}\{\text{randomdraw1}\} \leq 0.5$ | |
| select_one yesno | likesblue2 | Do you truly love the color blue? | $\text{\$}\{\text{consent}\}=1$ and $\text{\$}\{\text{randomdraw1}\} > 0.5$ | |

This gives an equal (50%) chance that each phrasing is used. The second question, about the color red, uses a similar method, drawing a second random number so that the second question's phrasing is chosen independently of the first's.

Note that the questions are individually *relevant* only if consent is given. As in the "basics" sample form above, the questions could also have been put inside a group so that consent relevancy could be handled at the group level.

Images: Adding pictures and other graphics

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Images: Adding pictures and other graphics* from the list. You can also click here to download the spreadsheet form definition (this is actually a .zip file that contains the .xlsx file as well as three image files referenced in the form definition; whenever you upload this form to your SurveyCTO server, you must be sure to also attach the image files as part of the upload).


The first *text* field in the spreadsheet form definition is configured to show the (old) SurveyCTO logo. This is accomplished by including the logo's filename in the *media:image* column (which you may need to scroll right to see):

| type | name | label | media:image |
|------|---------|-----------------------------------------|------------------------|
| text | anytext | Please enter some text here (any text). | Sample-Images-Logo.png |

This is how the question will appear to users:

Collect > Sample SurveyCTO Form - Image

Please enter some text here (any text).



The next field is a multiple-choice question. The possible answers are listed under the *choices* tab, and these also have images associated with them:


| list_name | name | label | image |
|-----------|------|----------|----------------------------|
| animal | 1 | Elephant | Sample-Images-Elephant.jpg |
| animal | 2 | Tiger | Sample-Images-Tiger.jpg |

This is how that question will appear to users:


Collect > Sample SurveyCTO Form - Image

What is your favorite animal?

☒ Elephant



☐ Tiger



Field lists: Multiple questions on a single screen

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Field lists: Multiple questions on a single screen* from the list. You can also click [here](#) to download the spreadsheet form definition.

In this sample, you will see two ways to display several questions on a single screen. In both cases, the questions to be displayed together have been grouped using *begin group* and *end group* rows. Also in both cases, "field-list" is specified in the *appearance* column, to indicate that these questions should appear on the same screen.

Following are the key elements of the first group in the spreadsheet form definition:

| type | name | label | appearance |
|-------------------|-----------|--------------------------|------------|
| begin group | page1 | Basic respondent details | field-list |
| text | name | What is your name? | |
| select_one region | hh_region | Which region are you in? | minimal |
| end group | page1 | | |

Above, "minimal" is also specified in the multiple-choice question's *appearance* column, in order to display the options as a drop-down menu rather than a list of radio buttons. This is how the questions will appear to users:

Collect > Sample SurveyCTO Form - Appearances

Sample survey module > Basic respondent details

What is your name?

Which region are you in?

Select One Answer ▼

In the second group, the "label" and "list-nolabel" appearances are used so that the series of multiple-choice questions will appear in a single table:

| type | name | label | appearance |
|------------------|---------------|------------------|--------------|
| begin group | page2 | Household assets | field-list |
| select_one yesno | labels | HH has a..? | label |
| select_one yesno | has_radio | Radio? | list-nolabel |
| select_one yesno | has_tv | TV? | list-nolabel |
| select_one yesno | has_motorbike | Motorbike? | list-nolabel |
| select_one yesno | has_car | Car? | list-nolabel |
| end group | page2 | | |

And this is how these questions will appear to users:

Collect > Sample SurveyCTO Form - Appearances

Sample survey module > Household assets

| HH has a..? | Yes | No |
|-------------|-----------------------|-----------------------|
| Radio? | <input type="radio"/> | <input type="radio"/> |
| TV? | <input type="radio"/> | <input type="radio"/> |
| Motorbike? | <input type="radio"/> | <input type="radio"/> |
| Car? | <input type="radio"/> | <input type="radio"/> |

Cascading selects: Filtering multiple-choice option lists

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Cascading selects: Filtering multiple-choice option lists* from the list. You can also click [here](#) to download the spreadsheet form definition.

When you open the spreadsheet form definition, you will see a series of three multiple-choice questions in the middle of the form:

| type | name | label | choice_filter |
|--------------------|----------------|--------------------------------------------------|---------------------------|
| select_one region | survey_region | In which region are you filling out this survey? | |
| select_one country | survey_country | In which country? | filter=\${survey_region} |
| select_one city | survey_city | In which city? | filter=\${survey_country} |

You may need to scroll right to find the *choice_filter* column, which is highlighted in orange. For any *select_one* or *select_multiple* field, the expression in this column specifies how to filter the available choices based on earlier responses (see *Dynamically filtering lists of multiple-choice options* for a complete discussion).

In this form, the *choices* sheet, which lists all possible answer options for multiple-choice questions, has an additional *filter* column:

| list_name | name | label | filter |
|-----------|------------|--------------|--------|
| region | AF | Africa | |
| region | SA | South Asia | |
| country | ZW | Zimbabwe | AF |
| country | ZA | South Africa | AF |
| country | IN | India | SA |
| country | BD | Bangladesh | SA |
| city | Harare | Harare | ZW |
| city | Bulawayo | Bulawayo | ZW |
| city | Pretoria | Pretoria | ZA |
| city | Cape_Town | Cape Town | ZA |
| city | New_Delhi | New Delhi | IN |
| city | Chennai | Chennai | IN |
| city | Dhaka | Dhaka | BD |
| city | Chittagong | Chittagong | BD |

The values in that *filter* column are referenced in the *choice_filter* expressions on the *survey* sheet, so the expressions and the values work hand-in-hand. So, for example, the *choice_filter* expression for the "In which country (are you filling out this survey)?" question is "filter=\${survey_region}", and the *filter* values for the country options are "AF" and "SA", which match the values for the region choice options.

If the user selects "South Asia" for region, for example, then this is the next question that will appear:

Collect > Sample SurveyCTO Form - Cascading select

In which country?

☐ India

☐ Bangladesh

And if the user then selects "Bangladesh", this will be the next question:

Collect > Sample SurveyCTO Form - Cascading select

In which city?

☐ Dhaka

☐ Chittagong

If the user instead selects "Africa" as the region, this is how the next question will appear:

Collect > Sample SurveyCTO Form - Cascading select

In which country?

☐ Zimbabwe

☐ South Africa

And if the user then selects "South Africa", this will be the next question:

Collect > Sample SurveyCTO Form - Cascading select

In which city?

☐ Pretoria

☐ Cape Town

You can see that the questions remain constant; the only things that change are the answer options, which are filtered based on previous responses.

Rosters: Two methods for repeated questions

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Rosters: Two methods for repeated questions* from the list. You can also click [here](#) to download the spreadsheet form definition.

This sample demonstrates two different methods for collecting household roster information, where the same questions are asked repeatedly in order to collect information about multiple household members. One method simply repeats the same questions using a series of fields, showing only as many questions as are

appropriate for the number of household members. The other method uses a *repeat group* to ask the same questions repeatedly, until the user ends the process. See *Repeating fields* for a full discussion of the different approaches to repeating questions.

The multiple-choice question in row 14 of the spreadsheet form definition allows the user to choose which of these two methods to use:

| type | name | label | hint |
|---------------------|-----------|-------------------------------------------------------------------|----------------------------------------------------------|
| select_one yesno | userepeat | Would you like to use a repeating group for the household roster? | A repeating group has a different UI and data structure. |

If the user selects "No", then the first method of gathering data will be used; if the user selects "Yes", the second method will be used. The field's hint reminds users that repeat-group answers will be organized differently in exported data. This is how the first question, including its hint, will look:

Collect > Sample SurveyCTO Form - Rosters

Would you like to use a repeating-group for the household roster?
A repeating group has a different UI and data structure.

☒ Yes

☐ No

If the user selects "No", then rows 16-34 describe how the survey will proceed. In this case, the user is first prompted to enter the number of family members in the respondent's household, then the survey proceeds to ask questions for exactly that number of people:

| type | name | label | relevance |
|----------------------|-------------|------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| integer | numfamily | How many family members live with you? | $\${\text{consent}}=1$ and $\${\text{userepeat}}=0$ |
| text | fam_name1 | What is the name of the first other family member who lives with you? | $\${\text{consent}}=1$ and $\${\text{userepeat}}=0$ and $\${\text{numfamily}} \geq 1$ |
| text | fam_name2 | What is the name of the second other family member who lives with you? | $\${\text{consent}}=1$ and $\${\text{userepeat}}=0$ and $\${\text{numfamily}} \geq 2$ |
| select_one gender | fam_gender1 | What is $\${\text{fam_name1}}$'s gender? | $\${\text{consent}}=1$ and $\${\text{userepeat}}=0$ and $\${\text{numfamily}} \geq 1$ |
| select_one gender | fam_gender2 | What is $\${\text{fam_name2}}$'s gender? | $\${\text{consent}}=1$ and $\${\text{userepeat}}=0$ and $\${\text{numfamily}} \geq 2$ |

Note the expressions in the *relevance* column for all of these fields: they all require "Yes" (which has a value of 1) to an earlier consent question and "No" (a value of 0) to the earlier "use repeat groups?" question. Additionally, some include conditions like $\${\text{numfamily}} \geq 2$ in order to only appear when appropriate; for example, if the respondent said that there was only one household member, then there should be no questions about a second household member.

If the user selects "Yes" to the multiple-choice question about using a *repeating group*, then rows 35-40 describe the questions that will follow:

| type | name | label | relevance |
|-------------------|---------------|------------------------------------------------------------------------------|-----------------------------------|
| note | repeatintro | Now I will ask you a few questions about each family member living with you. | \${consent}=1 and \${userepeat}=1 |
| begin repeat | fam_group | Family member | \${consent}=1 and \${userepeat}=1 |
| text | fam_rg_name | What is his or her name? | \${consent}=1 and \${userepeat}=1 |
| select_one gender | fam_rg_gender | What is his or her gender? | \${consent}=1 and \${userepeat}=1 |
| integer | fam_rg_age | What is his or her age? | \${consent}=1 and \${userepeat}=1 |
| end repeat | fam_group | | |

Before this set of questions is asked – or asked again – a pop-up window will ask if the user wishes to add (another) family member, using the label given in the *begin repeat* row:

As long as the user clicks *Add Group*, he or she can keep entering details for additional household members. As soon as he or she clicks *Do Not Add*, the survey continues (or, in this case, ends, since there are no additional questions).

See the sample below for an example of a third, hybrid method in which a repeat group is used, but the group is repeated a fixed number of times. And do see the full discussion in *Repeating fields* if you haven't already.

Rosters: A third, hybrid method for repeated questions (repeat_count)

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Rosters: A third, hybrid method for repeated questions (repeat_count)* from the list. You can also click [here](#) to download the spreadsheet form definition.

This sample demonstrates a method for handling rosters (repeated questions) that is a hybrid between the two methods demonstrated in the previous rosters sample. It asks the user how many family members there are, then uses a repeat group to ask questions about exactly that number of family members.

In the spreadsheet form definition, the crux of this method is the *repeat_count* column in the *begin repeat* row, which you may need to scroll to the right to see:

| type | name | label | repeat_count |
|--------------|-----------|----------------------------------------|---------------|
| integer | numfamily | How many family members live with you? | |
| begin repeat | fam_group | Family member | \${numfamily} |

In this case, the *repeat_count* column is used to repeat the group once for each family member, according to the count given by the *numfamily* field.

Rosters: Choosing among earlier entries

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Rosters: Choosing among earlier entries (variation 1)* or *Rosters: Choosing among earlier entries (variation 2)* from the list. You can also click [here](#) (variation 1) or click [here](#) (variation 2) to download the spreadsheet form definition.

This sample has two variations, both of which populate the possible answers to a multiple-choice question with the household member names entered earlier in the survey. In both, the choice lists are on the *choices* sheet of the spreadsheet form definition as follows:

| list_name | name | label | filter |
|-----------|------|---------------|--------|
| hhmember | 1 | \${fam_name1} | 1 |
| hhmember | 2 | \${fam_name2} | 2 |
| hhmember | 3 | \${fam_name3} | 3 |
| hhmember | 4 | \${fam_name4} | 4 |
| hhmember | 5 | \${fam_name5} | 5 |

The labels for the choices come from the names entered earlier, and the *filter* column is used to filter the list so that only the appropriate number of names are shown (i.e., so that blank names don't appear when there are fewer than five household members).

The choices are shown (and filtered) for the "worker_hhmember" field on the *survey* sheet:

| type | name | label | choice_filter |
|---------------------|-----------------|-------------------------|-----------------------|
| select_one hhmember | worker_hhmember | Which household member? | \${numfamily}>=filter |

The difference in the two variations of this sample form is as follows: variation one prompts for household member names in a series of fields named *fam_name1*, *fam_name2*, etc.; variation two uses a repeat group to prompt for household member names, then has to pull the names out of the repeat group using a series of *calculate* fields named *fam_name1*, *fam_name2*, etc. These calculate fields are defined as follows:

| type | name | calculation |
|-----------|-----------|---------------------------------------------------|
| calculate | fam_name1 | indexed-repeat(\${fam_rg_name}, \${fam_group}, 1) |
| calculate | fam_name2 | indexed-repeat(\${fam_rg_name}, \${fam_group}, 2) |
| calculate | fam_name3 | indexed-repeat(\${fam_rg_name}, \${fam_group}, 3) |
| calculate | fam_name4 | indexed-repeat(\${fam_rg_name}, \${fam_group}, 4) |
| calculate | fam_name5 | indexed-repeat(\${fam_rg_name}, \${fam_group}, 5) |

These *calculate* fields use the indexed-repeat() function (see *Using expressions in your forms*) to pull the names from the earlier repeat group and put them into individual fields that can be referenced in the labels on the *choices* sheet.

Rosters: Collecting repeated information with multiple repeats

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Rosters: Collecting repeated information with multiple repeats* from the list. You can also click [here](#) to download the spreadsheet form definition.

In the spreadsheet form definition, you will find two groups of repeated questions, with the second group referring to answers from the first. The first group asks for all family members' names, and the second follows up to ask, later, about all family members' ages.

After consenting to participate in the survey, the user will be prompted whether they want to add a family member:

If the user chooses to add a group, the following question will appear:

Collect > Sample - Repeat Group Referring to Earlier Repeat Group

Sample survey module > HH member names (1)

What is the name of household member #1?

After entering the name of one household member, the user will then be prompted whether or not another household member should be added:

Collect > Sample - Repeat Group Referring to Earlier Repeat Group

Sample survey module > HH member names (1)

What is the name of household member #1?

John

i Add One More Group?

Add another "HH member names" group?

Add Group Do Not Add

And if the user chooses to add another household member, the following question will appear:

Collect > Sample - Repeat Group Referring to Earlier Repeat Group

Sample survey module > HH member names (2)

What is the name of household member #2?

Following is how the workbook appears for this portion of the survey. You will notice that the label in the *begin repeat* row is the same as the text within the quotation marks in the "Add New Group?" pop-up above.

| type | name | label | calculation |
|--------------|------------|-------------------------------------------------------|-------------|
| begin repeat | names | HH member names | |
| calculate | namenumber | | index() |
| text | name | What is the name of household member #\${namenumber}? | |
| end repeat | names | | |

You can see that *index()* has been used as the calculation expression for the *calculate* field (see discussion of the *index()* function in *Using expressions in your forms*). This function returns the index or repetition number for the current repeat group; in this case, it is used to identify the number of the current family member, which is used in the label for the *name* field.

Returning to the survey, when the user chooses not to add any more family members, the following question will appear to ask about the first household member's age:

How old is John?

This question will repeat the same number of times as the previous one, and it will dynamically integrate the name for each family member that was entered earlier. This is how the workbook appears for this repeated question:

| type | name | label | calculation | repeat_count |
|--------------|-----------------|------------------------------------|----------------------------------------------|------------------|
| begin repeat | ages | HH member ages | | count(\${names}) |
| calculate | namefromearlier | | indexed-repeat(\${name}, \${names}, index()) | |
| integer | age | How old is \${namefromearlier}? | | |
| end repeat | ages | | | |

Note that you may need to scroll right in the workbook in order to see the *calculation* and *repeat_count* columns.

In the *repeat_count* column, the *count()* function is used so that the second group repeats exactly the same number of times as the earlier *names* group repeated (i.e., so that it collects one age to go with every name entered earlier).

In the *calculation* column, the *indexed-repeat()* function is used to pull the earlier-entered name to go with each age. This allows the names gathered earlier to be integrated into the labels for the later questions about family member ages.

To read more about these and other functions available for calculations, see *Using expressions in your forms*.

An example with nested repeat groups

Click [here](#) to download a more complex, nested version of this sample form.

The more complex version of this sample nests repeat groups three levels deep, asking about families, then, within each family, about household members, and then, within each family member within each family, about each pet. Finally, ages are collected in a second set of nested repeat groups. (This is a much more complex example, but it might be helpful if you are designing, for example, a form that collects information on agricultural plots and then, within each plot, on the crops planted on that plot.)

Pre-loading: Referencing pre-loaded .csv data

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Pre-loading: Referencing pre-loaded .csv data* from the list. You can also click [here](#) to download the spreadsheet

form definition (this is a .zip file that contains a .xlsx sample form as well as a .csv file containing sample data for pre-loading).

If you upload the spreadsheet form definition to your SurveyCTO server manually, be sure to attach the .csv file – or upload the .csv file as a dataset named *hhplotdetails* and then attach that dataset – otherwise the form won't be able to find the data when users are filling it out.

When you open the .csv file, you will see that the first column is titled *hhid_key*. Here, "hhid" stands for "household identification number"; the suffix "_key" indicates that the values in this column will be used to uniquely identify each row, so SurveyCTO should index the column's contents for fast look-up. (See *Pre-loading data into a form* for a full discussion of the techniques employed here.)

| hhid_key | nplots | plot1description | ... |
|----------|--------|------------------------------|-----|
| 1001 | 1 | HHID 1001 Plot 1 Description | ... |
| 1002 | 2 | HHID 1002 Plot 1 Description | ... |
| ... | | | |

This .csv file represents data that the survey team already has about the number of plots that each household owns (in the *nplots* column), as well as short descriptions and sizes for each of those plots (in the *plot#description* and *plot#size* columns). (Further to the right, there are other columns as well, but you can ignore those for now as they are only used in the next sample.)

Turning to the spreadsheet form definition for this form, you will see that the user is first asked to double-enter the household identification number of the respondent. He or she will be provided the hint that "Valid IDs are between 1001 and 1060" because those are the households covered in the accompanying .csv file.

| type | name | label | constraint |
|---------|--------------|-------------------------------------|---------------------|
| integer | hhid | Please enter the household's ID. | .>=1001 and .<=1061 |
| integer | confirmentry | Please re-enter the household's ID. | .= \${hhid} |

Note that the *constraint* for the *hhid* field intentionally allows 1061 to be entered, even though that is not a valid ID in the .csv file. This is allowed so that you can see how the form behaves when an entered ID cannot be found in the .csv data.

Next in the workbook, you will see a block of twenty *calculate* fields. These are internal, hidden fields, so nothing you see in this block will appear to the user filling out the form. Each of these pulls a different piece of data from the .csv file and assigns it a name that can be referenced later in the survey. The expression in the *calculation* column of each calls the *pulldata()* function, passing the following as parameters: which .csv file to pull from, which column to pull, and which column and value to use in identifying the correct row to load.

Finally, you will see a block of actual questions that will be visible to the user. The labels for most of them make reference to the data values that were earlier pulled from the .csv file.

| type | name | label | relevance | choice_filter |
|------|------|-------|-----------|---------------|
|------|------|-------|-----------|---------------|

| | | | | |
|---------------------|-----------------|----------------------------------------------------------------------------------------------------|---------------|--------------------|
| select_one plot | bestplot | Please select your best-performing plot in the most recent agricultural season. | | filter<=\${nplots} |
| note | plotsintro | I will now ask you a series of questions about each of your \${nplots} agricultural plots. | \${nplots}>=1 | |
| select_one yesno | plot1cultivated | Did you cultivate your "\${plot1description}" (size \${plot1size}) plot in the most recent season? | \${nplots}>=1 | |
| select_one yesno | plot2cultivated | Did you cultivate your "\${plot2description}" (size \${plot2size}) plot in the most recent season? | \${nplots}>=2 | |

All of the values inside the $\${}$'s (for example, $\${nplots}$ and $\${plot1size}$) refer back to the names of *calculate* fields from earlier, in order to pull in those field values – which, in turn, were pulled in from the pre-loaded .csv data.

You will also notice that respondents will only be asked questions about plots they own. In the first question in this block, which asks respondents to identify their best plot, the expression *filter<=\${nplots}* in the *choice_filter* column – along with the accompanying answer options on the *choices* tab – ensure that respondents will be presented with a list of only those plots that they own. In later fields, the expressions *\${nplots}>=1* or *\${nplots}>=2* in the *relevance* column indicate that these questions should only be asked if the number of plots the respondent owns is high enough.

Pre-loading: Searching and selecting from pre-loaded data

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Pre-loading: Searching and selecting from pre-loaded data* from the list. You can also click here to download the spreadsheet form definition (this is a .zip file that contains a .xlsx sample form as well as a .csv file containing sample data for pre-loading).

If you upload the spreadsheet form definition to your SurveyCTO server manually, be sure to attach the .csv file – or upload the .csv file as a dataset named *hhplotdetails* and then attach that dataset – otherwise the form won't be able to find the data when users are filling it out.

This sample is precisely the same as the previous pre-loading sample form, except that instead of prompting for the user to double-enter the household ID, this sample allows the user to select among households in the .csv file. The new *hhidentrychoice* field allows the user to choose how to proceed:

Collect > Sample Form - Preloading - Search and select

Sample preload module

Please select how you would like to select the household ID:

☐ Multiple-choice selections

☒ Search region for household

Regardless of the user's choice, the next question will ask the user to choose a region:

Collect > Sample Form - Preloading - Search and select

Sample preload module

Please select the household's region.

☒ Region 1

☐ Region 2

The list of available regions comes from the .csv file. This is because the *select_one* field in the spreadsheet form definition calls the *search()* function in its *appearance* column and the *choices* sheet specifies the .csv columns to use for the choice labels and values:

| type | name | label | appearance |
|-------------------|------------|---------------------------------------|-------------------------|
| select_one region | hhidregion | Please select the household's region. | search('hhplotdetails') |

| list_name | name | label |
|-----------|----------|--------|
| region | regionid | region |

The "search('hhplotdetails')" *appearance* indicates that the choice options should include all unique values in the hhplotdetails.csv file. The *name* and *label* columns of the *choices* sheet indicate the .csv columns to use for the choice values and labels (*regionid* and *region* respectively).

If the user chose "Multiple-choice selections" at the first prompt, then the region choice will be followed by a choice of available districts within that region:

Collect > Sample Form - Preloading - Search and select

Sample preload module

Please select the household's district.

☐ District 1

☐ District 2

☐ District 3

Here, the list of districts will be filtered to only include those districts within the selected region. This is accomplished with additional parameters to the *search()* *appearance*:

| type | name | label | appearance |
|---------------------|--------------|-----------------------------------------|-----------------------------------------------------------------|
| select_one district | hhiddistrict | Please select the household's district. | search('hhplotdetails', 'matches', 'regionid', '\${hhidregion}) |

| list_name | name | label |
|-----------|------------|----------|
| district | districtid | district |

Next, the district choice will be followed by a choice of available villages within that district:

Collect > Sample Form - Preloading - Search and select

Sample preload module

Please select the household's village.

☐ Village 3 (district 2)

☐ Village 4 (district 2)

This is again accomplished via an appropriate *search()* *appearance* and appropriate column names in the *choices* sheet:

| type | name | label | appearance |
|------|------|-------|------------|
|------|------|-------|------------|

| | | | |
|-----------------------|-------------|-------------------------------------------|-----------------------------------------------------------------------|
| select_one village | hhidvillage | Please select the household's village. | search('hhplotdetails', 'matches', 'districtid', \${hhiddistrict}) |
|-----------------------|-------------|-------------------------------------------|-----------------------------------------------------------------------|

| list_name | name | label |
|-----------|-----------|---------|
| village | villageid | village |

Finally, the village choice will be followed by a choice of available households within that village:

Collect > Sample Form - Preloading - Search and select

Sample preload module

Please select the household.

☐ Household 1011 (village 3)

☐ Household 1012 (village 3)

☐ Household 1013 (village 3)

☐ Household 1014 (village 3)

☐ Household 1015 (village 3)

This is yet again accomplished via an appropriate `search()` *appearance* and appropriate column names in the *choices* sheet:

| type | name | label | appearance |
|--------------------|--------------|---------------------------------|---------------------------------------------------------------------|
| select_one hhid | selectedhhid | Please select the household. | search('hhplotdetails', 'matches', 'villageid', \${hhidvillage}) |

| list_name | name | label |
|-----------|----------|-----------|
| hhid | hhid_key | household |

If the user had instead chosen "Search region for household" at the first prompt, then the region choice would be followed by a choice of options for searching within the chosen region:

Collect > Sample Form - Preloading - Search and select

Sample preload module > HH search options

Search type:

Contains

Search text:

The workbook side of this looks like this:

| type | name | label | appearance |
|-------------------------|------------|-------------------|------------|
| begin group | searchopt | HH search options | field-list |
| select_one searchoption | searchtype | Search type: | minimal |
| text | searchtext | Search text: | |

| | | | |
|-----------|-----------|--|--|
| end group | searchopt | | |
|-----------|-----------|--|--|

| list_name | name | label |
|--------------|------------|-------------|
| searchoption | contains | Contains |
| searchoption | matches | Matches |
| searchoption | startswith | Starts with |
| searchoption | endswith | Ends with |

This includes the "field-list" *appearance* on the *begin group* row, in order to show multiple fields on a single screen; the *searchtype* field to allow the user to choose among the search options supported by the *search()* function ("contains", "matches", and so on); and the *searchtext* field to allow the user to enter the search text itself.

Then on the following page of the survey, the user will be able to select among all households identified in their search:

This is accomplished with particular parameters to the *search()* function:

| type | name | label | appearance |
|--------------------|-----------|------------------------------|--------------------------------------------------------------------------------------------------|
| select_one hhid | foundhhid | Choose a matching household: | search('hhplotdetails', \${searchtype}, 'household', \${searchtext}, 'regionid', \${hhidregion}) |

| list_name | name | label |
|-----------|----------|-----------|
| hhid | hhid_key | household |

Here, the *search()* function specifies the search type, the column to search, the value for which to search, the column by which to filter, and the text by which to filter. The *hhid* row on the *choices* sheet specifies which columns from the .csv to use for choice values and labels (*hhid_key* and *household* respectively).

Because every row of the .csv file has a similar description in the *household* column (like "Household 1009 (village 2)"), certain searches will find no households (e.g., *contains* "z") and other searches will locate all 60 (e.g., *starts with* "H").

Note: given that there are two different methods of selecting the household (either multiple-choice selections or search), a different variable is used for each option's results: (1) *selectedhhid*, which is the household result when the multiple-choice option is chosen, and (2) *foundhhid*, which is the household result when

the search option is chosen. It is easier for later fields in the form to be able to reference just a single *hhid* field. Therefore, a *calculate* field is used to record the appropriate household ID for use later in the form:

| type | name | calculation |
|-----------|------|------------------------------------------------------------|
| calculate | hhid | if(\${hhidentrychoice}=1, \${selectedhhid}, \${foundhhid}) |

The if() function is used to store the appropriate value into the *hhid* field, depending on the case.

Follow-ups: Asking follow-up questions for a list of selected items

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Follow-ups: Asking follow-up questions for a list of selected items* from the list. You can also click [here](#) to download the spreadsheet form definition (this is a .zip file that contains a .xlsx sample form, a .csv file containing sample data for pre-loading, and the original .xlsx source for the .csv file).

If you upload the spreadsheet form definition to your SurveyCTO server manually, be sure to attach the .csv file – or upload the .csv file as a dataset named *sampleq4list_options* and then attach that dataset – otherwise the form won't be able to find the data when users are filling it out.

This sample is somewhat advanced. It combines and extends techniques illustrated by the earlier *Rosters: Collecting repeated information with multiple repeats* and *Pre-loading: Searching and selecting from pre-loaded data* sample forms.

The focus here is on five methods you might use to ask follow-up questions for a selected list of options. For example, you might want to ask a farmer which of many possible crops she has cultivated, then follow up with a list of specific questions for each of the crops she chose; or, you might ask which of many types of assets she acquired over the past year, then follow up with additional questions about those assets. This sample itself is framed around crops, but the demonstrated techniques are entirely general: you can use them for any case in which you want to ask follow-up questions for just those items selected from a list.

Method 1 uses a simple *select_multiple* field to ask the user to check off one or more crops, then uses a *repeat group* to iterate through each selected crop. (This method is not recommended for reasons described below.)

| type | name | label | repeat_count |
|-----------------------------|-----------------|-----------------------------------------------------------|-----------------------------------------|
| select_multiple croplist | selected_crops1 | Please choose zero or more crops from the following list. | |
| begin repeat | crop_repeat1 | Follow-up questions (\${crop_name1}) | count- selected(\${selected_crops1}) |
| ... | | | |
| end repeat | crop_repeat1 | | |

Within the repeat group, *calculate* fields are used to pull the appropriate crop ID and label from the earlier *select_multiple* field, which in turn informs the open-ended follow-up question follows:

| type | name | label | calculation |
|-----------|----------------|-----------------------------------------------------------------------------------|--------------------------------------------------------|
| calculate | crop_id1 | | selected-at(\${selected_crops1}, index()-1) |
| calculate | crop_name1 | | jr:choice-name(\${crop_id1}, '\${selected_crops1}') |
| text | crop_question1 | This is an example follow-up question about the following crop: \${crop_name1} | |

See the earlier "rosters" samples above for the basics of asking repeated questions, or see the help on using expressions for more about the functions used in this form.

Method 2 is precisely the same, except the list of options comes from a pre-loaded .csv file (or attached dataset) rather than coming from the *choices* sheet. This method might be preferred when the list of options is extremely long and/or you would like to be able to update the list of options without having to update the form itself. See the sample on searching and selecting from pre-loaded data for more on pulling multiple-choice options from a pre-loaded .csv file. (This method is also not recommended for reasons described below.)

To the user, Method 3 looks identical to Method 2, but behind the scenes the implementation is importantly different – and it avoids one serious drawback present in both Method 1 and Method 2. In the earlier methods, everything would work fine so long as the user proceeded linearly through the form, selecting crops and entering follow-up responses, but it would present some difficulties if the user were to go back and change the crop selections after entering follow-up responses.

For example, say the user selects crops 1, 2, and 4, answers the follow-up questions, and then goes back and selects also crop 3. The third set of follow-up questions would now suddenly be associated with crop 3, whereas it had earlier been associated with crop 4 (since crop 4 had been the user's third selection). The user would need to go through and carefully adjust his or her responses accordingly.

The solution used by Method 3 is to always repeat the follow-up questions *x* times, where *x* is the total number of crops (in contrast, Methods 1 and 2 used the *number of selected crops* as *x*, rather than the total). In order to avoid asking questions about crops that weren't selected, Method 3 then employs the standard *relevance* capability, to simply hide the follow-up questions associated with crops that weren't selected.

Method 4 uses this same method, but using options from the *choices* sheet rather than using options pre-loaded from a .csv file. Thus, Method 3 is the preferred alternative to Method 2 and Method 4 is the preferred alternative to Method 1.

Both Methods 3 and 4 do come at a cost: by including follow-up questions for every possible choice – albeit, with the non-selected ones hidden – the form might perform a bit slower (particularly when there are a large number of options and/or follow-up questions and when the Android devices are older). More blank data will also be exported for all of the hidden follow-up questions, though this might be viewed as an advantage since the data structure might be simpler to use in practice.

Finally, Method 5 demonstrates an alternative questioning sequence to Method 3: instead of asking the user to select all relevant crops before moving to follow-up questions, Method 5 simply iterates through all crops asking a yes/no question first and then asking follow-up questions after each yes.

Upload the sample form – along with its pre-loaded option data, `sampleq4list_options.csv`, as an attachment (or upload the .csv file as a dataset named *sampleq4list_options* and then attach that dataset) – and try the form for yourself. It presents a simple enough interface for users but uses a few relatively advanced techniques to make that possible.

Auditing: Including text and audio audits

To open or use this sample form, go to the Design tab, scroll down to the *Your forms and datasets* section, and click + then *Start new form*; then enable *Use a sample form as your starting point* and choose *Auditing: Including text and audio audits* from the list. You can also click [here](#) to download the spreadsheet form definition.

This sample form takes the "basics" sample form and adds both text and audio audits (for more about these kinds of audits, see the help topic on monitoring for quality). The key fields added to the workbook are:

| type | name | appearance |
|-------------|-------------|-------------------|
| text audit | text_audit | p=100 |
| audio audit | audio_audit | p=100; s=0; d=900 |

Because the *text audit* field has *p=100* in its *appearance* column, the form will save text audits for 100% of filled-out forms. When the form's data has been downloaded and exported, these text audits will be saved as individual .csv files in the *media* subdirectory. For example:

| Field name | Total duration (seconds) | First appeared (seconds into survey) |
|--------------------------|--------------------------|--------------------------------------|
| intronote | 3 | 0 |
| consent | 5 | 3 |
| consented[1]/name | 3 | 8 |
| consented[1]/age | 3 | 10 |
| consented[1]/confirmnote | 1 | 13 |

Because the *audio audit* field also includes *p=100* in its *appearance* column, the form will also save audio recordings for 100% of filled-out forms. On download and export, these will also be included as individual audio files in the *media* subdirectory. And because *s=0; d=900* was also specified in the *appearance* column, the audio recording will begin immediately (0 seconds into the survey) and last for up to 15 minutes (900 seconds of duration).

Dataset basics: Using listing survey details in a household survey

Click [here](#) to download the dataset basics sample form. This .zip file contains two .xlsx files with form definitions plus a .xml file with a dataset definition.

You will upload this particular sample to your server in three parts: as two forms and one dataset that links them. First, go to the *Your forms and datasets* section of the Design tab and upload each of the two .xlsx files, one at a time, using + followed by *Upload form definition*. Then, click + again and *Add server dataset* to add a new dataset, choose the *New dataset from definition* tab, and upload the .xml file.

This sample demonstrates how datasets can be used to link multiple survey forms into a single workflow. In this case, the workflow is this:

- Household listing -> household sample -> household surveys

Accordingly, the sample includes two survey forms and a dataset to link them: a household listing survey, a household sample dataset, and a household survey. The survey forms are short; they do not include the litany of questions you would normally include in a household listing or in a full household survey. These forms are meant only to demonstrate the logic of connecting surveys via datasets.

Look first at the household listing form included with this sample. It includes fields for the household address and the head of household's name. It also includes several calculated fields:

| type | name | calculation |
|-----------|--------------|---------------------------------------|
| calculate | hhid | once(int(random()*900000)+100000) |
| calculate | random_draw1 | once(random()) |
| calculate | in_sample | if(\${random_draw1} < 0.60, 1, 0) |
| calculate | random_draw2 | once(random()) |
| calculate | survey_team | if(\${random_draw2} < 0.50, 'A', 'B') |

The *hhid* field automatically generates a random six-digit household ID for each listed household. The *in_sample* field uses a random draw to decide whether or not to include this household in the sample: 60% of the time *in_sample* will be given a 1, and 40% of the time it will be given a 0. Finally, *survey_team* uses a second random draw to assign 50% of households to survey team A and 50% to team B.

In this case, the listing form itself assigns household IDs, chooses a sample, and makes random assignments. You might prefer to reserve some or all of those tasks for your back-office team; while the goal of this sample is to demonstrate a fully-automated workflow, in practice you might well build a semi-automated workflow instead.

This household listing form publishes data directly into the dataset included with this sample. Specifically, the *hhid* field publishes into the dataset's *hhid_key* field, the address and head-of-household fields publish to fields by the same names, and the *survey_team* field publishes to the *team* field. Importantly, publishing is set so that only submissions for which *in_sample* is equal to 1 are published into the dataset; that's because we want this dataset to hold just our randomly chosen household sample, not all listed households.

This dataset is then attached – as pre-loaded data – to the second form included with this sample, the household survey form.

This second form begins with a simple *select_multiple* field that asks for which of two teams to list households.

List assigned households for which survey team?

☒

Team A

☐

Team B

The next field asks the enumerator to select one of the sample households assigned to the selected survey team.

Which household are you surveying?

☐ Address 2 (hoh_name: Name 2)

☐ Address 3 (hoh_name: Name 3)

☐ Address 5 (hoh_name: Name 5)

☒ Address 6 (hoh_name: Name 6)

☐ Address 10 (hoh_name: Name 10)

☐ Address 12 (hoh_name: Name 12)

☐ Address 13 (hoh_name: Name 13)

☐ Address 15 (hoh_name: Name 15)

☐ Address 16 (hoh_name: Name 16)

This field dynamically loads the list of options from pre-loaded data – in this case, data pre-loaded from the attached dataset (see *Loading multiple-choice options from pre-loaded data*). Following are the relevant rows from the *survey* and *choices* worksheets of the form definition.

| type | name | label | appearance |
|----------------------|------|------------------------------------|--------------------------------------------------|
| select_one household | hhid | Which household are you surveying? | search('hh_sample', 'matches', 'team', \${team}) |

| list_name | name | label |
|-----------|----------|-------------------|
| household | hhid_key | address, hoh_name |

The `search()` appearance indicates that the choice options should include unique values from the *hh_sample* dataset, from all rows for which the *team* column matches the selected survey team ("A" or "B"). The *name* and *label* columns of the *choices* sheet indicate the dataset columns to use for the choice values and labels (*hhid_key* for the values and both *address* and *hoh_name* for the labels).

Next, once the user has selected a specific household, the `pulldata()` function is used to read the *address* and *hoh_name* fields from the pre-loaded data (i.e., from the dataset).

| type | name | calculation |
|-----------|----------|---------------------------------------------------------|
| calculate | address | pulldata('hh_sample', 'address', 'hhid_key', \${hhid}) |
| calculate | hoh_name | pulldata('hh_sample', 'hoh_name', 'hhid_key', \${hhid}) |

Now part of the household survey form, these fields can be referenced to, for example, solicit confirmation that the enumerator is at the correct household.

Are you at the correct house?

Address: Address 6
Head of HH: Name 6

☒ Yes

☐ No

As mentioned before, in this sample everything is fully automated: whenever new household listing forms are submitted to the server, the household sample dataset – and the household survey form to which it is attached – is automatically updated within 10 minutes. An alternative workflow might be:

1. The listing form publishes key household details into a household dataset – but the form does not include drawing a random household ID, drawing a random sample, or assigning surveys to survey teams.
2. A member of the field or back-office team downloads the household dataset and updates it to include household ID's, an indicator of which households to include in the sample, and survey team assignments, all in additional columns of the downloaded dataset.
3. The team member uploads the updated dataset to the server, allowing the server to merge the revised data with any additional data that might have been published in the mean time (if, i.e., new household listing forms were submitted since the team member downloaded the dataset).
4. The household survey form filters the list of possible households to include only those in the sample (since now the dataset includes both households in and out of the sample).

There are an infinite number of possible workflows. This sample and accompanying discussion are just meant to give you an idea of what is possible. The next sample continues by extending this one to include an additional back-check survey.

Dataset back-checks: Adding random back-checks to a household survey

Click [here](#) to download the dataset back-checks sample form. This .zip file contains three .xlsx files with form definitions plus two .xml files with dataset definitions.

This sample extends the dataset basics sample just above, so you may want to review that sample first. One of the two datasets used here and two of the three survey forms are shared with that sample (so this sample only includes one extra dataset and one extra survey form).

You will upload this sample to your server in five parts: as three forms and two datasets that link them. First, go to the *Your forms and datasets* section of the Design tab and upload each of the three .xlsx files, one at a time, using + followed by *Upload form definition*. Then, click + again and *Add server dataset* to add a new dataset, choose the *New dataset from definition* tab, and upload the .xml file; then do the same for the second .xml file. (If you've already uploaded the one dataset for the above sample, you don't need to re-upload that.)

This sample demonstrates how datasets can be used to link multiple survey forms into a single workflow to perform back-checks. Expanding upon the prior sample, the new two-part workflow is as follows:

- A: Household listing -> household sample -> household surveys
- B: Household surveys -> back-check sample -> back-check surveys

Accordingly, the sample includes three survey forms and two datasets to link them: a household listing survey, a household sample dataset, a household survey, a back-check sample dataset, and a back-check survey.

To understand the first part of the workflow (labeled *A* above), please see the prior sample. Here, we will focus on the second part (labeled *B*).

Look first at the household survey form, *Sample-Dataset-Basics-HHSurvey.xlsx*. In addition to the fields discussed in the sample above, it includes three *calculate* fields at the very end:

| type | name | calculation |
|-----------|---------------|-----------------------------------|
| calculate | randomdraw_bc | once(random()) |
| calculate | bc_selected | if({randomdraw_bc} <= 0.15, 1, 0) |
| calculate | survey_date | once(today()) |

These fields use a random draw to select 15% of surveys for back-checking (i.e., for auditing via independent re-visits), then record the date on which each survey was conducted. (Building the random selection into the survey itself automates the entire process, but you could also organize a workflow that allows *you* to randomly select surveys for back-checking; in that case, you'd manually update a server dataset by uploading your own back-check selections.)

This household survey form publishes data directly into the back-check sample dataset included with this sample. Specifically, the *hhid* field publishes into the dataset's *hhid_key* field, and the address, survey date, and head-of-household fields publish to fields by the same names. In particular, the age recorded for the head of household publishes to the dataset because we want to use the back-check survey to verify that this age was captured correctly. Also, the publishing is set so that only submissions for which *bc_selected* is equal to 1 are published into the dataset because we want this dataset to hold just our randomly-chosen back-check sample (not all surveyed households).

This dataset is then attached – as pre-loaded data – to the third form included with this sample, the back-check survey form.

This third form begins with a field that asks the enumerator to select one of the households randomly selected for back-checks.

Which household are you surveying?

- ☐ Address 2 (hoh_name: Name 2)
- ☐ Address 3 (hoh_name: Name 3)
- ☐ Address 5 (hoh_name: Name 5)
- ☒ Address 6 (hoh_name: Name 6)
- ☐ Address 10 (hoh_name: Name 10)
- ☐ Address 12 (hoh_name: Name 12)
- ☐ Address 13 (hoh_name: Name 13)
- ☐ Address 15 (hoh_name: Name 15)
- ☐ Address 16 (hoh_name: Name 16)

This field dynamically loads the list of options from pre-loaded data (in this case, data pre-loaded from the attached back-check sample dataset). Following are the relevant rows from the *survey* and *choices* worksheets of the form definition.

| type | name | label | appearance |
|----------------------|------|------------------------------------|---------------------|
| select_one household | hhid | Which household are you surveying? | search('bc_sample') |

| list_name | name | label |
|-----------|----------|-------------------|
| household | hhid_key | address, hoh_name |

The `search()` *appearance* indicates that the choice options should include all unique values from the *bc_sample* dataset. The *name* and *label* columns of the *choices* sheet indicate the dataset columns to use for the choice values and labels (*hhid_key* for the values and both *address* and *hoh_name* for the labels).

Next, once a specific back-check household has been chosen, the `pulldata()` function is used to read the *address*, *hoh_name*, *hoh_age*, and *survey_date* fields from the pre-loaded data (i.e., from the dataset).

| type | name | calculation |
|-----------|--------------|-------------------------------------------------------------------------|
| calculate | address | <code>pulldata('bc_sample', 'address', 'hhid_key', \${hhid})</code> |
| calculate | hoh_name | <code>pulldata('bc_sample', 'hoh_name', 'hhid_key', \${hhid})</code> |
| calculate | original_age | <code>pulldata('bc_sample', 'hoh_age', 'hhid_key', \${hhid})</code> |
| calculate | survey_date | <code>pulldata('bc_sample', 'survey_date', 'hhid_key', \${hhid})</code> |

Now part of the back-check survey form, these fields can be referenced to, for example, solicit confirmation that the enumerator is at the correct household.

Are you at the correct house?

Address: Address 6
Head of HH: Name 6

☒ Yes
☐ No

Fields can also be used to internally check whether responses recorded during the back-check match responses recorded during the original survey. In this sample, the *age_different* field is used to determine whether the household head's age recorded during back-check is different from what was recorded during the original survey.

| type | name | calculation |
|-----------|---------------|-----------------------------------------------------------|
| calculate | age_different | <code>if(\${hoh_age_bc} != \${original_age}, 1, 0)</code> |

If there is a discrepancy between the two answers, this field will be recorded with a value of 1; otherwise, it will be recorded with a value of 0. This field is not displayed to the back-checker. Rather, it is discreetly recorded in the survey form so that you can easily check discrepancy rates after you have exported the data. (You could also imagine other designs, however. For example, you could automatically ask the back-checker to double-check an entry whenever it doesn't match with what was recorded in the original survey.)

This sample leaves off at this point. However, you could add another server dataset that merges key information from the listing, household, and back-check surveys. In that dataset, you could have one row, identified by the unique household ID, with information from all three surveys (including whether it was sampled, whether it was back-checked, and what the error rate was in cases where back-checks were done). There are a great many possible designs and configurations.