

# Creating a Real-time HFC Dashboard

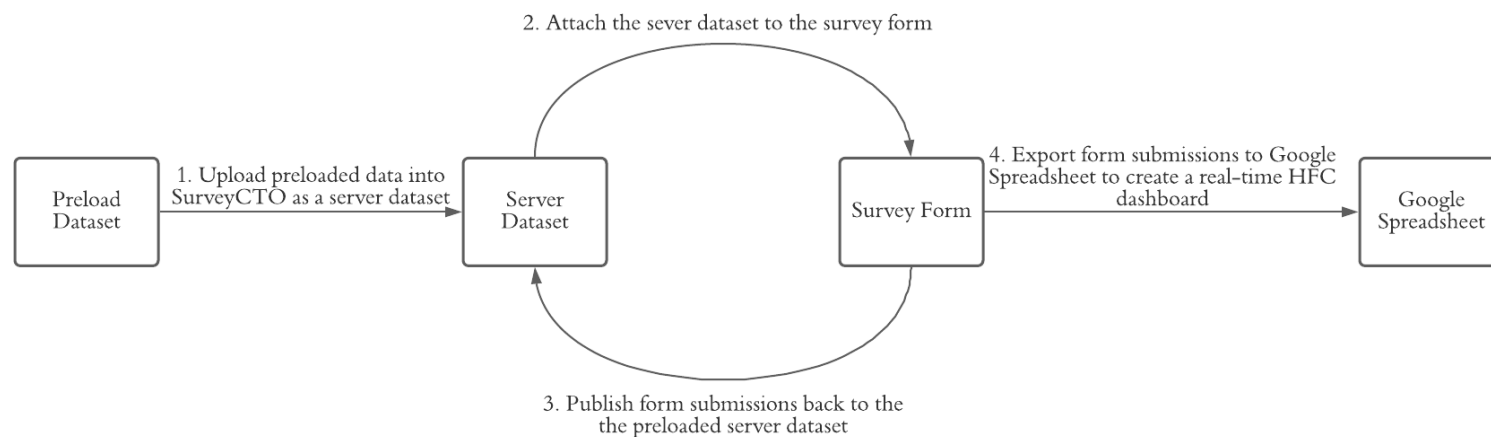
<b>Creating a Real-time HFC Dashboard</b>	<b>1</b>
<b>Introduction</b>	<b>3</b>
<b>SECTION I - Preparing Survey Forms</b>	<b>5</b>
A. Step One: Design survey form	5
B. Step Two: Prepare pre-loaded dataset	8
<b>SECTION II - Uploading Survey Forms</b>	<b>12</b>
A. Step One: Upload your main form into SurveyCTO	12
B. Step Two: Upload preloaded data into SurveyCTO as a server dataset and attaching to the survey form:	12
i. Uploading preloaded data into SurveyCTO as a server dataset:	12
ii. Attaching the server dataset to survey form:	14
C. Step Three: Further editing the surveyCTO forms (with pre-load completed)	14
i. Go back to the survey form (necessary): Edit your survey form to include the following fields	15
ii. Go back to the survey form (optional: module completion checks):	16
iii. Go back to the survey form (optional: additional features)	16
D. Step Four: Publishing form submissions back to the server dataset:	19
<b>SECTION III - Setting Up a Real-time HFC Dashboard</b>	<b>21</b>
A. Step One: Export form submissions to a Google Sheet to create a real-time HFC dashboard.	21
B. Step Two: Creating HFC tabs	23

<b>SECTION IV - Limitations</b>	<b>27</b>
A. Why the issue exists	27
B. Issues commonly faced:	27
C. Best practices	29
<b>SECTION V - Complementary Analysis in R</b>	<b>32</b>

## **Introduction**



This document provides a detailed instruction for creating a real-time high-frequency check dashboard.

As shown in the flowchart below, there are four main steps for creating a real-time HFC dashboard. First, we need to upload a static preloaded dataset into SurveyCTO as a server dataset. Then, we attach this server dataset to the main survey form so that we can reference preloaded data in our survey form. Next, by using the “Publish” command, we stream data from new submissions back into the server dataset and reuse it for the next submission. This feature allows us to turn a static preloaded dataset into a dynamic one. Therefore, we can track the completion status and survey outcomes from previous submissions. Similarly, data from new submissions can also be exported to an external Google Sheet, where we can set up the HFC dashboard by leveraging Google Sheet functions to create descriptive statistics tables and graphs. As survey submissions will be automatically published to the Google Sheet, the HFC dashboard is also updated automatically.



## Links

Document	Without Module Completion	With Module Completion
1. Preloaded Data	<a href="#">+ Preload Data Sample</a>	<a href="#">+ Preload Data Sample (With Module ...</a>
2. SurveyCTO Form Template	<a href="#">+ Survey Form Template (without mo..</a>	<a href="#">+ Survey Form Template (without mod...</a>
3. SurveyCTO Form (Test and Fill)	<a href="https://boruis.surveyccto.com/collect/demo_survey?caseid=">https://boruis.surveyccto.com/collect/demo_survey?caseid=</a>	<a href="https://boruis.surveyccto.com/collect/demo_survey_module_completion?caseid=">https://boruis.surveyccto.com/collect/demo_survey_module_completion?caseid=</a>
4. Dashboard Template	<a href="#">+ HFC Dashboard Template (Viewer-mode)</a> <a href="#">+ HFC Dashboard Template (Editor-mode)</a>	

5. Survey Process Flowchart	 survey flowchart (simplified).png
6. Sample Analysis	 sample_analysis_report.pdf & <a href="#">sample_analysis.R</a>
7. Sample Check	<a href="#">sample_checks.R</a>

## **Data Security and Encryption**

Almost all the data we collect include information that can be used to identify who the respondent is. To protect respondents' privacy and data security, the instructions below must be followed carefully. Eventually we will de-identify the data so it can be shared freely among the research team; before that, all data should be encrypted during each step of the survey collection process.

- A. Encrypting form data on SurveyCTO: When programming questionnaires in SurveyCTO, you need to create a public-private key pair for your survey and use the provided public key to encrypt your survey form. Instructions on how to create an encrypted form and access encrypted data can be found [here](#). The keys should be stored securely in a [password manager](#).
- B. Encrypting identifiable data stored locally: When downloading survey data from SurveyCTO or preparing the preload datasets, you need to create a safe storage place for the PII data. A World Bank-approved software solution is VeraCrypt. Instructions on how to create secure encrypted folders on VeraCrypt can be found [here](#). For

the purpose of data security, it is essential that **you never save data with personally identifiable information on your computer, Dropbox folders, OneDrive folders, or anywhere else.**

## **SECTION I - Preparing Survey Forms**

### **A. Step One: Design survey form**

The first step is to create a surveyCTO data-collection form. Instructions on how to create one can be found [here](#). When designing your survey form, it is important to take into account all possible scenarios that an enumerator may encounter during the collection process. The form templates shared in this documentation, for example, provide a sample questionnaire for phone surveys with 3 possible outcomes. Before moving forward, we clarify two concepts used frequently across this document - completion status and survey outcome.

- **Completion Status (complete vs incomplete):** This variable describes the result of the survey from the perspective of the enumerators. “Complete” surveys are surveys requiring NO FURTHER action from the enumerators regardless of whether the survey is filled out by the respondent or not. Specifically, in the survey process flowchart, “complete” surveys refer to the end node boxes highlighted in green and red, while “incomplete” surveys are the box highlighted in yellow.
- **Survey Outcomes (pending vs responded vs non-responded):** This variable describes the result of survey from the research perspective. It further divides “complete” surveys into “responded” surveys and “non-responded” surveys. Specifically,
  - “Pending” surveys are incomplete surveys (yellow box)

- “Responded” surveys are “complete” surveys filled out by the respondents (green box)
- “Non-responded” surveys are “complete” surveys but NOT filled out by the respondents (red box). This could happen if the respondent is unavailable, refuses to participate or keeps rescheduling.

The table below provides a detailed explanation of the 3 outcomes shown in the survey process flowchart. To summarize the flowchart, it proceeds as follows:

- Is the respondent able to conduct the interview?
  - ➡ **If yes** → introduce the purpose of the survey and ask for consent to proceed
    - ➡ **If agree to participate** → proceed with survey questions
      - Survey Status: Completed, Responded
    - ➡ **If refuse to participate** → end of survey
      - Survey Status: Completed, Non-responded
  - **If no** → indicate the reason why the interview cannot be conducted
    - ➡ For reasons such as wrong number, illness, death → end of survey
      - Survey Status: Completed, Non-responded
    - ➡ For reasons such as no one answer the call, the respondent was busy → call again at another time
      - Survey Status: Incompleted, Pending
      - NOTE: if the enumerator has already contacted the same respondent 3 times but the respondent never answered the call or kept rescheduling, there is no need to call the respondent again and the survey status will become “Completed, Non-responded”

Completion Status	Survey Outcome	Color of End Node Box	Without Module Completion	With Module Completion
Complete	Responded	Green	Respondent agreed to participate and completed the survey	Respondent agreed to participate and completed all modules
Complete	Non-responded	Red	Respondent refused to participate or was unable to conduct the survey due to illness, death, wrong number	
Incomplete (Call Back)	Pending	Yellow	<p>a) Respondent did not answer the call or was busy and wanted to reschedule</p> <p>(NOTE: if enumerators have called 3 times but the respondent never answered the call or kept rescheduling, the survey status will then become “Completed, Non-responded”)</p>	<p>a) Respondent did not answer the call or was busy and wanted to reschedule</p> <p>b) Respondent agreed to participate but did not complete all modules</p> <p>(NOTE: if enumerators have called 3 times but the respondent never answered the call or kept rescheduling, the survey status will then become “Completed, Non-responded”)</p>

*See template for how these are coded in the main form.*

## B. Step Two: Prepare pre-loaded dataset



### *i. Without modules tracker*

The preloaded data should include at least the following items:

(see [Preload Sample \(without Module Completion\)](#))

- a. **basic survey information** such as the household/respondent id and assigned enumerator name.
  - i. Household/respondent id should be unique
- b. an **attempt counter** column with pre-assigned values of 0 (0 submissions)
  - i. This variable describes the number of previous submissions for each survey
- c. a **completion status** column with pre-assigned values of 0 (Incomplete)
  - i. This variable will be changed to 1 once the survey is completed by the enumerator
- d. a **survey outcome** column with pre-assigned values of 2 (Pending)
  - i. Once the completion status becomes 1, this variable will be changed to 1 if the survey is filled out by the respondent or to 0 if the survey is NOT filled out by the respondent

### *ii. With module completion status*

*Please note that module tracker is useful for very long surveys but comes with a set of challenges, so if your survey is short please proceed with the template/explanation above.*

In addition to the variables listed above, the original preloaded data should also include the following item:

(see [Preload Sample \(with Module Completion\) - Part A](#))

- e. a **last module completion status** column with pre-assigned values of 0 (0 module completed):

- i. This variable describes the last module completed by the respondent
- ii. Assuming your survey has 5 modules in total and the 5 modules are organized in a sequential order, in the cases where the respondent left in the middle of the survey, this variable will be changed from 0 to X (X is the number of modules completed by the respondent before leaving). Next time when the enumerator opens the same survey, it will start from the (X + 1)th module instead of the first module.

You will also need to prepare **another preload data sheet (extra!)** with 2 columns:

(see [Preload Sample \(with Module Completion\)- Part B](#))

- a. **module name:** the name of each module
- b. **module index:** the corresponding index of each module (e.g. index = 1 if it is the first module appeared in the survey, index = 2 if it is the second module appeared in the survey)

### **How it works (if you're interested)**

When you submit module A, and then you go back to do the survey again and submit module B - in principle we would think that surveyCTO would overwrite module A and show an incomplete survey but it seems to be keeping the data from module A in the first submission.

- o **Explanation:** In surveyCTO, an empty field can take on two potential statuses. It can be an implicit null value or an explicit null value. Implicit null values are fields INACCESSIBLE to the enumerators during the survey because they do not meet the relevance condition(s). Explicit null values are fields accessible to the enumerators but with an empty value.

- When SurveyCTO exports data to the Google Sheet, explicit null values are exported to the Google Sheet and will REPLACE all existing values in the matched columns. However, implicit null values will NOT be exported and therefore, the existing values in the Google Sheet will NOT be replaced/updated.
- Example (1): In the form template, we asked the enumerators to enter the next call date (field label: next\_call\_datetime) if the respondent is unavailable right now. If the enumerator is able to complete the survey with the respondent, this question will not be shown to the enumerator. Therefore, this field (next\_call\_datetime) will have an implicit null value under such circumstances. However, in the form template, there is a calculate field that formats results from the next\_call\_datetime field. Since it is a “calculate” field, though hidden, it is always updated with every submission. Therefore, this field will have an explicit null value even if the enumerator is never asked to enter a next call date.
  - This means that if the survey is submitted a second time (and completed that second time) the enumerator will not have to fill this out, and the next\_call\_datetime field will remain empty. However, because it is an explicit null value due to the presence of calculate field that formats the result, this data will overwrite any previously stored next\_call\_datetime from an incomplete survey submitted earlier.
- Example (2): Say an enumerator completes Module A on the first attempt, and then when he calls back he is brought to Module B (because module A was already completed). Then in this second survey, all the values from Module A are implicit null (never prompted to fill these out because it was already completed); and values from Module B are filled in. When google sheets exports the data Module A will not be ‘overwritten’ because they are implicit nulls.

### **The benefits/drawbacks of module completion**

- a) Advantages - In long surveys or surveys dealing with sensitive topics it may be difficult to get respondents to stay for the whole duration in one sitting so this allows submission of partial surveys and can be picked up on a later date / time, even by another enumerator.
- b) Disadvantages - It can be time consuming to code up and the code is prone to errors. Next, when you start up an incomplete survey, it will start after the last module not question so if halfway through module X the respondent drops off, the next time the survey will begin at module X's first question, so you lose data.
  - i) Cleaning is also difficult: you have to collapse different submissions for the same survey into 1 and it becomes hard to track
  - ii) The issues related to duplicates that we detail in the limitation section below are further exacerbated.

## SECTION II - Uploading Survey Forms

A. Step One: Upload your main form into SurveyCTO

B. Step Two: Upload preloaded data into SurveyCTO as a server dataset and attaching to the survey form:

*i. Uploading preloaded data into SurveyCTO as a server dataset:*

- a. Under the SurveyCTO console's Design tab, click "Add server dataset" -> "New dataset for data".
- b. Select the preloaded data from your computer or Google Drive depending on the file location.
- c. More information can be found [here](#).

SurveyCTO

1. Design

2. Collect

3. Monitor

4. Export

Configure

How to design your survey forms

Your forms and datasets

Tools

Refresh

Search

Organize

Help

Start new form

Add group

Upload form definition

Add server dataset

Create a new dataset

Help

Is this what you want? Server datasets are for attaching data to forms, case management, and advanced data flows. You don't need a server dataset to download data submitted for any form; for that, just go to the *Export* tab. [Click here to learn more about server datasets...](#)

New dataset for data

New dataset for cases

New dataset from definition

Dataset title:

The dataset title here...

Dataset ID:

The dataset id...

Optionally, upload a file with initial dataset contents. You can upload it directly from your computer (.csv or .xlsx file) or use Google Drive (.csv file or Google Sheet):

Upload from computer

Upload from Google Drive

Please choose a dataset file to upload (.csv or .xlsx, optional):

Select file...

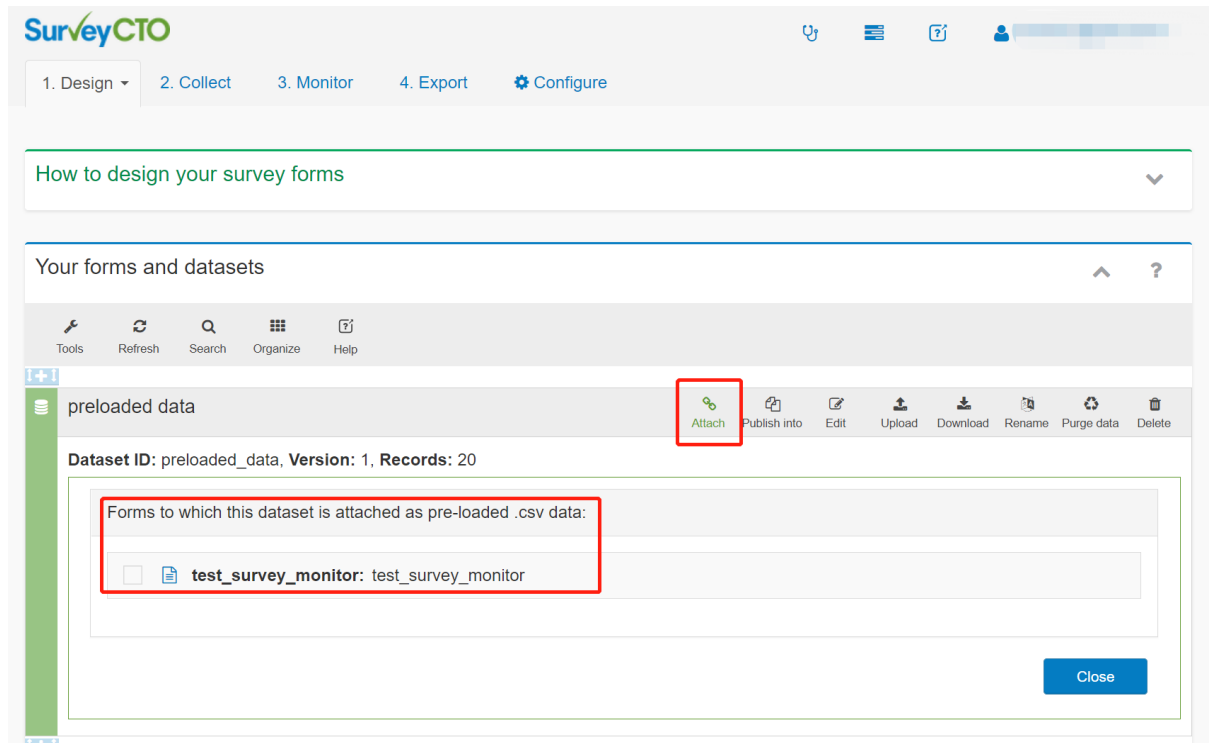
Cancel

Create dataset

14

*ii. Attaching the server dataset to survey form:*

- d. Under the SurveyCTO console's Design tab, click the "Attach" icon on the selected server dataset and then check the survey form you want to attach
- e. This step allows users to reference variables in the preloaded dataset.



C. Step Three: Further editing the surveyCTO forms (with pre-load completed)

*i. Go back to the survey form (necessary):* Edit your survey form to include the following fields

Since the survey form needs to actively pull data from this server dataset - your survey form needs to have calculate fields that will reference this server dataset.

*NOTE: For encrypted forms, all the following fields must be explicitly marked as “publishable” (with a “yes” in their publishable column). Fields explicitly marked as publishable are left unencrypted so that they can be conveniently published to cloud services or directly downloaded from the server.*

- a. Generic: “Calculate” fields that pull variables from the preloaded data. There should be one “Calculate” field for each column in the preloaded data
  - Example: `pulldata('hhplotdata', 'plot1size', 'hhid_key', ${hhid})` will pull a value from an attached server dataset called hhplotdata (This is the dataset id you entered when you created the server dataset). The value will come from the plot1size column of the pre-loaded data, and the hhid field will be used to identify the matching row in the pre-loaded data's hhid\_key column
  - Recommended naming convention: adding a “\_pl” suffix to the original name
- b. For tracking: A “Calculate” field that updates the **attempt counter** variable every time the enumerator submits a survey.
  - Example: `${attempt_counter_pl} + 1`, where attempt\_counter\_pl is the name of the pulled variable
- c. For tracking: Two “Calculate” fields that respectively update the **completion status** variable and the **survey outcome** variable every time the enumerator submits a survey
  - This can be created by setting the ifelse argument with/or coalesce argument. For specific examples, please see the surveycto form template.



- Recommended naming convention: adding a “\_update” suffix to distinguish from the pulled variables

*ii. Go back to the survey form (optional: module completion checks):*

**a. Submission in the middle of a survey:**

- A yesno “select\_one” field at the end of every module (except the last module) that asks “*Is the respondent still available? (Please select “No” if you agreed to stop the survey here and start again a some other time)*”
- A “calculate” field (must be placed at the end of the survey) for every module you have in the survey. This variable tracks the completion status for each module. It should be equal to 1 if a particular module is completed and otherwise 0.
- A “calculate” field that adds up the **last\_module\_completed** variable pulled from the preload sheet and all the 0-1 module completion status calculate fields you created for each module.
- Add a relevance expression for every module group you have in your survey form so that only modules that are not completed in the previous submissions are visible to the respondents.

*iii. Go back to the survey form (optional: additional features)*

**a. Duplicate submissions check:**

- This check refers to the orange “Check 1” box in the survey process flowchart.
- After the enumerator selects a household id, the survey form will check the completion status for the selected household id from the preloaded server dataset. If the completion status is equal to 1 (“Complete”), a warning note will appear preventing the enumerator from moving forward.
- This warning is created by adding a “Note” field with a relevance expression of “`{complete_status_pl} = 1`” and “required action” option set to “Yes”.

The screenshot shows a survey application interface. At the top, there is a header bar with a back arrow icon, the text "Enumerator & Call Info", and a "Go to" button with a right arrow icon. Below the header, there is a central message box with a double left arrow icon on the left. The message box contains the following text: "The survey for farmer **Md. Gudor mondal** has been marked as complete because the respondent has already filled out this survey or the maximum number of attempts has been reached. Please go back and select a different farmer to start with." Below this, it says "If the above description is incorrect, please contact XXX to resolve." At the bottom of the message box, there is a red bar with the text "Sorry, this response is required!". At the bottom of the interface, there are two buttons: "Previous" with a left arrow icon and "Next" with a right arrow icon.

b. **Maximum number of submissions** (subject to change):

- This check is designed to close a survey if the number of previous submissions has reached a certain threshold, which in the sample survey form is 5. In other words, if the enumerator has contacted the respondent 5 times and the respondent either never picked up the call or kept rescheduling, the enumerator DOES NOT have to contact the respondent again.
- This check is created by pulling the attempt counter variable from the preloaded dataset. If attempt counter + 1 is equal to the threshold, completion status will be changed from "Incomplete" to "Complete", and survey outcome will be changed from "Pending" to "Non-responded".

c. **Next call date** (subject to change):

- This constraint refers to the 7.1 box in the survey process flowchart.

- If no one answers the phone or the respondent reschedules with the enumerator, the enumerator needs to enter a next call date and time. The next call date must be within the next 2 days.
- This constraint is created by adding a constraint expression of “( number(`${next_call_datetime}`) - number(now()) ) <= 2”.

d. **Check whether a response was changed:**

- By adding a “calculate here” field with the expression “`once(${fieldname})`” right after the question, we are able to capture the initial response entered to see if the enumerators/respondents fill in answers and then go back to change them.
- As this expression only captures the initial responses, it is recommended to add one additional calculate field with the argument “`if(once(${fieldname}) != ${fieldname}, 1, 0)`” to create a dummy variable that returns 1 if the answer is changed.
- For more information, see the form template or this [link](#).

e. **Monitor Module Duration:**

- Create a “calculate here” field with the expressions “`once(format-date-time(now(), '%Y-%b-%e %H:%M:%S'))`” at the beginning and the end of a module to capture the start time and end time for the module.
- This allows you to track how long an enumerator completes a module.

f. **Enumerator Check:**

- it's often better to assign enumerators to each survey ahead of time, but when you can't then we propose the following:

- The enumerator selects their name, the village, and the farmer's name/household ID they want to survey.
- For every submitted survey, we track the enumerator's name so that if the survey is not completed, a note will appear the next time the survey is opened to remind whomever is opening the survey who the previous enumerator to complete the survey was.

***D. Step Four: Publishing form submissions back to the server dataset:***

- a. Under the SurveyCTO console's Design tab, click the "Publish into" icon on the selected server dataset
- b. Click "Add a form" and then select the survey form that the preloaded data is attached to
- c. Under "Field Mapping", the following fields should be added
  - A variable/field that identifies the unique records. In the example below, "hhid" is the unique id in both the preloaded dataset and survey form.
  - Any dynamic variables created in form design. In the case below, these are attempt counter, completion status and survey outcome. As these variables are updated after every submission, they are published back to replace the original variables in the preloaded dataset.
- d. Under "Form field to identify unique records (optional)", select the same variable in step two-02-d-i
- e. More information can be found [here](#)

SurveyCTO

1. Design

2. Collect

3. Monitor

4. Export

Configure

boruis11@gmail.com

Tools

Refresh

Search

Organize

Help

preloaded\_data

Attach

Publish info

Edit

Upload

Download

Rename

Purge data

Delete

Dataset ID: preloaded\_data, Version: 1, Records: 20

Forms that publish to dataset with ID 'preloaded\_data'

+ Add a form

Add a form

Select a form for which new submissions will stream data into this dataset:

test\_survey\_monitor

Field mapping:

+ Add

+ Add all

hhid	→	hhid	Delete
attempt_counter_update	→	attempt_counter	Delete
complete_status	→	complete_status	Delete
survey_result	→	survey_result	Delete

Repeated fields are listed with an \* at the end of their names, and they must be mapped to a destination that also has an \* at the end. When repeated data is published, the \* will be automatically replaced by \_1 for the first instance, \_2 for the second, and so on.

Form field to identify unique records (optional):

hhid

If there is already a matching row in the dataset, it will be updated.

Include form submissions whenever this field is 1 (optional):

Publish existing data?

Close

Save

Close

## SECTION III - Setting Up a Real-time HFC Dashboard

### A. Step One: Export form submissions to a Google Sheet to create a real-time HFC dashboard.

1. **Readying the Google Sheet:** Create a Google Sheet that is the exact same as the preloaded data but without any identifiable information.
2. **Publishing form submissions to the Google Sheet:**
  - a. Under the SurveyCTO console's Export tab, under "Advanced" settings, click the "Configure" icon on the selected survey form
  - b. Click on "Add connection" and select the Google Sheet created above
  - c. Under "Field Mapping",
    - i. Repeat steps in section II-C-iii-c
    - ii. Add any additional variables you need for the HFC dashboard, including but not limited to survey duration for evaluating enumerator performance, outcome variables for outlier checks, etc...

SurveyCTO

1. Design2. Collect3. Monitor4. Export

Configure

Advanced: publishing form and dataset data to the cloud

RefreshSearchHelp

preloaded data

test\_survey\_monitor

Publishing to: (nowhere)

test\_survey\_monitor

Publishing to: (nowhere)

Form publishing options

GOOGLE SPREADSHEETS

Inactive

+ Add connection

New Google Sheets connection

Where do you want to publish this form to?

HFC Dashboard Template

Field mapping:

hhid

complete\_status

survey\_result

attempt\_counter\_update

hhid

complete\_status

survey\_result

attempt\_counter\_update

Delete

Delete

Delete

Delete

Repeated fields are listed with an \* at the end of their names, and they must be mapped to a destination that also has an \* at the end. When repeated data is published, the \* will be automatically replaced by \_1 for the first instance, \_2 for the second, and so on.

Form field to identify unique records (optional):

hhid

If there is already a matching row on the Google side, it will be updated.

Include form submissions whenever this field is 1 (optional):

Publish existing data?

Close

Save

## **B. Step Two:** Creating HFC tabs

***Leveraging functions in Google Sheet to create HFC dashboard:*** After setting up the Google Sheet connection in SurveyCTO, your data will be automatically streamed into the connected Google Sheet after every submission. The final step is to build up the HFC dashboard inside the Google Sheet by using Pivot Tables and other functions. The shared template includes the following elements:

**Convert all columns in google sheet into text format before exporting any data.**

- a. **Tracking Sheet:** This sheet is designed for FCs and enumerators to track the progress of each survey. It is created by first referencing relevant information from the exported sheet and then using conditional formatting to highlight “incomplete” surveys (definitions of “complete” vs “incomplete” surveys are detailed in section I-A). This sheet should be similar to the raw, exported data but with finer formatting.
- b. **Aggregated Tracking (Enumerator-level):** This sheet provides data monitoring at the enumerator-level. Specifically, it tracks:
  - Number of surveys completed
  - Average response rate
  - Average survey duration and number of attempts to contact
- c. **Aggregated Tracking (Village-level):** This sheet provides data monitoring at the village-level. Similar to the enumerator-level tracking, it calculates:



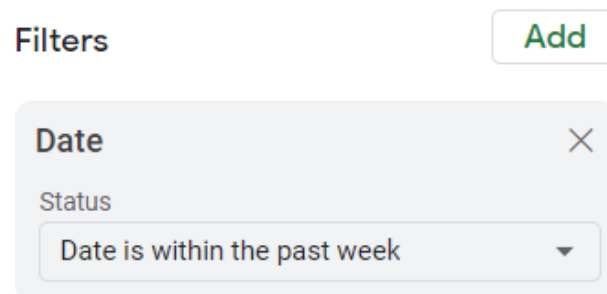
- Number of surveys completed
- Average response rate

**d. Data Quality (by Enumerator, full data):** The purpose of this sheet is to give a preview of the outcome variables and check for abnormal patterns, illogical values and outliers. For each module, we create

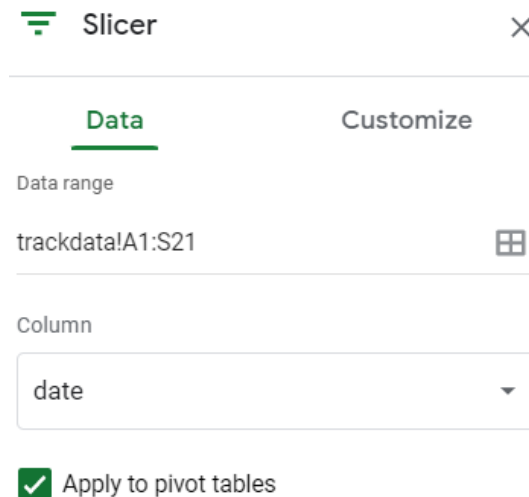
- Summary statistics tables that describe the distributions of outcome variables
- (optional) Percentage of times that an answer is changed (by using conditional formatting, we highlight values that surpass a certain threshold - in the template it is 10%)

**e. Data Quality (by Enumerator, recent data only):** Considering that during later stages of data collection, recent trends might be neglected and unnoticeable due to the large volume of existing data, therefore, it is important to create a separate data quality sheet just for data submitted over the past 7 days. This also allows us to capture if enumerators have rectified any early mistakes that we catch. The most easiest way to do this is

- Duplicate the Data Quality Full Data sheet
- Add a filter argument to each PivotTable you've created and under "filter by condition", select "Date is within past week"
  - Note: you need to first export the "SubmissionDate" variable to your Google Sheet dashboard and then convert it to datetime format



- f. **Data Quality (by Enumerator or by a particular date):** For the same reasons as mentioned above, it is necessary to have the option to filter for specific enumerators or specific dates. The easiest way to do this is
- Duplicate the Data Quality Full Data sheet twice (one for filtering for enumerator, one for date). Remove any tables which are not relevant for these tabs like those tabulating responses by date and by enumerator in a single table.
  - Go to “Data” in the Google Sheets Toolbar and select “Add a slicer”.
  - Then select the sheet that your pivot tables are pulling data from (in our sample dashboard, this is the “trackdata” sheet) and select the entire dataset.
  - Select the variable you want to filter by. In our sample dashboard, we use enumerator name to filter under (Enum\_Wise) and date in (Date\_Wise). Ensure that the box is checked to apply this slicer to all pivot tables on that sheet.



- Enum\_Wise demonstrates how to view performance of one or more enumerators over time. Similarly, Date\_Wise allows us to view performance on a particular day or selected dates by all enumerators. This is

relevant if there are some dates which are crucial to monitor, or if a single or group of enumerators' performance needs to be observed. The filters can be updated even without access to edit the dashboard.

**Instructions on Google Sheet functions:** The graphs and tables in the HFC template are created mainly using the following functions:

- a. **PivotTable**: PivotTable functions are used to create aggregate tables.
  - i. Select the raw SurveyCTO-exported data and in the menu at the top, click **Data > Pivot Table**.
  - ii. In the side panel, **Rows** are your level of aggregation (e.g. use enumerator IDs as Rows if it is enumerator-level tracking), and **Values** are the calculations you want to show in your columns.
  - iii. In our template we create three types of pivot tables
    1. Continuous vars: one that shows mean/median/min/max (by enumerator)
    2. Continuous vars: one that shows means over time (by enumerator)
    3. Categorical vars: one that shows response frequency by category
    4. Categorical vars: one that shows response frequency for one answer in the category option set (e.g. don't know or other) over time. This can also be used for questions where "yes" triggers a host of additional questions and you want to check the frequency that enumerators select the "yes" option over time.
- b. **Reference Data from Other Sheets**: The Tracking tab is created by referencing data from the sheet with the raw exported data. Since the table under this tab is not at an aggregated level, we cannot use PivotTable functions.
  - i. For every cell, type "=" followed by the sheet name, an exclamation point, and the cell being copied". For example, a cell with argument "=Sheet1!A1" is copying data from the A1 cell in Sheet1.

- c. **Conditional Formatting**: Under the Tracking tab, we use conditional formatting to highlight incomplete/pending surveys.
- i. In the menu at the top, click **Format > Conditional Formatting**.
  - ii. In the side panel, under “**Apply to range**”, select all cells (excluding headers) and under “**Format rules**” select “**Custom formula is**” and type “= \$G3 = FALSE”. The \$ before the G ensures that the format for each column will always look to column G of the current row. This allows you to highlight all cells in the selected range if the corresponding value in the Completion Status column is FALSE.

## SECTION IV - Limitations

### A. Why the issue exists

Our surveys are designed to pull modules and survey completion dynamically from a server dataset which updates the number of attempts and completion status with each new survey filled. In the event that the server has not been updated or is unable to pull new data, we see duplicates. The SurveyCTO Collect app does not do this uploading and downloading automatically unless the device is connected to WiFi and surveys need to be sent to the server and datasets refreshed manually. Further, there is a brief delay when publishing new submissions into server dataset/Google Sheets from SurveyCTO. The published data will only update 5-10 minutes after the most recent submission has been received.

### B. Issues commonly faced:

- i. Duplicates
- ii. Imperfect number of attempts
- iii. Encryption

***The first two issues below come from imperfect practices by enumerators (not uploading right away) or imperfect Internet (which means things aren't synching well)***

- i. **Duplicates** are created and go unnoticed in 1 of 2 ways

- a. Completed surveys: This occurs when an enumerator does not upload completed surveys to the server. Thus, the server dataset does not mark the survey as completed. The next time an enumerator attempts the same UID/HHID it will not alert you that this has already been filled, and thus leads to it being filled again.
- b. Incomplete surveys: This issue occurs when a surveyor makes an attempt wherein they are unable to start the survey or are only able to complete it partially, and are unable to send these surveys to the server. Thus, even if a second or greater attempt is made without the server dataset being updated, anyone else attempting the same HHID/UID will not see the latest number of attempts made. This also creates an issue in partially completed surveys. If a survey is partially completed till module X without the dataset being updated, the next attempt will start the survey at whichever point it was last updated instead of at module X+1.

**ii. Imperfect number of attempts:** The survey captures the number of attempts incorrectly owing to poor network and bad practices. For example, you make one attempt which is uploaded to the server; and then you make further attempts (which would be attempt 2, 3, 4, 5) but do not upload them to the server immediately, but instead altogether later (i.e. they aren't sequentially uploaded to the server because of internet connection or bad practices). Then once you do upload all, the tracker doesn't register this as 4 separate attempts, it registers it as a second attempt only.

However, capturing the number of attempts is desirable. Submitting a survey for each attempt allows the most accurate tracking of both, enumerator effort and reachability of the sample. If in the same sample multiple surveys are being conducted, it is a strong indicator of the number of attempts after which the effort to conduct a round of calls is too high for the conversion of successful attempts.

**iii.** Lastly, it doesn't seem like the server dataset in surveyCTO is **encrypted** - this can be further looked into before pursuing this route.

## C. Best practices

1. Wait 5-10 minutes after each submission before starting the next one
2. Compare the attempt counter note shown in the survey form with the one listed in the Google Sheet to ensure the preloaded data is up-to-date
3. Notes for field coordinators for better data management:
  - i. When you're on a mobile network, SurveyCTO doesn't automatically upload the surveys to the server. It waits until you're connected to WiFi or requires you to upload the surveys manually. This is why we see lags in survey submission.
    1. This can be changed within the application (under General Settings) to allow for automatic upload when any mobile network is available. However, if enumerators are working in areas with little to no internet coverage, this will remain an issue. Ultimate decision - depends on WiFi and/or mobile network availability

**C** Collect > General Settings

**Auto Send/Receive**

Auto send with Wi-Fi  
Auto send when Wi-Fi is available ☐

Auto send with network  
Auto send when network is available ☐

Auto download with Wi-Fi  
Auto download form updates when Wi-Fi is available ☐

Auto download with network  
Auto download form updates when network is available ☐

Auto download on demand  
Check for updates when opening case list or blank form ☐

Auto install downloaded updates  
Auto install any updates that have been downloaded ☐

Display send/receive status  
Show form send/receive status on main menu ☐

#### 4. Protocols for updating the server dataset

- a. Enumerators should change surveycto settings to auto upload surveys and auto download the new server dataset when connected to a MOBILE network
- b. Whenever enumerators are in a network zone (whether mobile or WiFi), they should manually upload and download to ensure they are working with the most updated version. This is of utmost importance between attempts to the same household / respondent.



- c. These processes must be followed as frequently as possible. While after each survey is ideal, field coordinators should remind them to do so as frequently as possible. If enumerators devices do not have internet, field managers should have a hotspotting device

## SECTION V - Complementary Analysis in R

The HFC dashboard provides snapshot of a subset of the data (Field Manager)

- 1) Enumerator Tracking tab: Completion / Responses
- 2) Enumerator Productivity tab (time):
- 3) A few data points (1 per module mapped out - especially focusing on questions that lead the survey to be longer as enumerators may be inclined to answer a certain way to shorten survey)

**Reasons to do complementary analysis in R (RA)**

- 1) **Deep dive into the variables** (see *sample\_analysis.R*):
  - a) Track multiple select outcomes that are difficult to visualize using pivot tables in google.
  - b) Track all of the variables in the dataset to produce a latex file that allows the team to scroll through all the variables in the dataset
    - i) We produce a template code that shows how to create 1) summary stats table; 2) bar graph (for categorical variables; 3) histogram and k-density for continuous variables.
- 2) Produce a list for duplicates for complete surveys (same UID submitted twice but with different answers - we need to check with FC which one to keep) (see *sample\_checks.R*)
- 3) Finalize list of incomplete surveys (sometimes preloads are not updated in a timely fashion, and may indicate that a survey is incomplete when the correct number of attempts (5) have been made and the respondent wasn't reached OR the survey is actually complete. We need to do a more manual check to remove these cases so that the FC doesn't go back to the team asking for more submissions) (see *sample\_checks.R*)

- 4) Flag inconsistencies in incomplete surveys (the number of attempts/completion status in the tracker (which result from the limitations we discuss above in the server capturing this information). (see *sample\_checks.R*)
- 5) Produce a list of outliers (see *sample\_checks.R*)
- 6) Time series deep dive (see *sample\_analysis.R*): Track how response behaviors are changing over time. This is only really helpful for smaller survey teams where the graphs are more legible.
  - a) Number of surveys per day + per day/per enumerator
  - b) Average survey duration per day + per day/per enumerator
  - c) Share of 'other' per day + per day/per enumerator
  - d) Average response to a question per day + per day/per enumerator