Branching Logic in REDCap – How-To Guide

This guide describes how to format and apply branching logic, also known as skip logic, to questions using the *Online Designer* method to build a database in REDCap. Branching logic can be as complex as you need it to be, with many layers of logic.

What is branching logic?

Branching Logic, also known as skip logic, may be employed when fields in the database need to be *hidden* during certain circumstances. For example, it may be best to hide fields related to pregnancy if the subject in the database is male. If you wish to make a field visible ONLY when the values of other fields meet certain conditions (and keep it invisible otherwise), you may provide these conditions in the Branching Logic section in the *Online Designer*, shown by the double green arrow icon 

Note: The logic for showing or hiding a question goes into the field(s) you want to show or hide, not the main / parent question. For example, if someone answers 'Yes' to *Do you smoke?* and you want an additional question to appear asking *For how many years*, you would put the branching logic on the question that asks how long, not on the *Do you smoke* question.

For basic branching, you can simply drag and drop field names as needed in the Branching Logic dialog box in the *Online Designer*. If your branching logic is more complex, you will create equations using the syntax described below.

In the equation you must use the project **variable names** surrounded by **[ ] brackets**. You may use mathematical operators **(=, <, >, <=, >=, <>)** and Boolean logic (**and/or**). You may nest within many parenthetical levels for more complex logic.

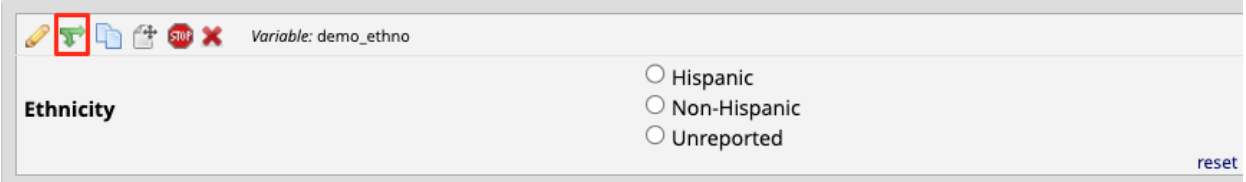
You must **ALWAYS** put single or double quotes around the values in the equation UNLESS you are using > or < with numerical values (i.e., height, weight, BMI – text fields validated as a number). Example: [weight] > 80 (the subject's weight is greater than 80 kg)

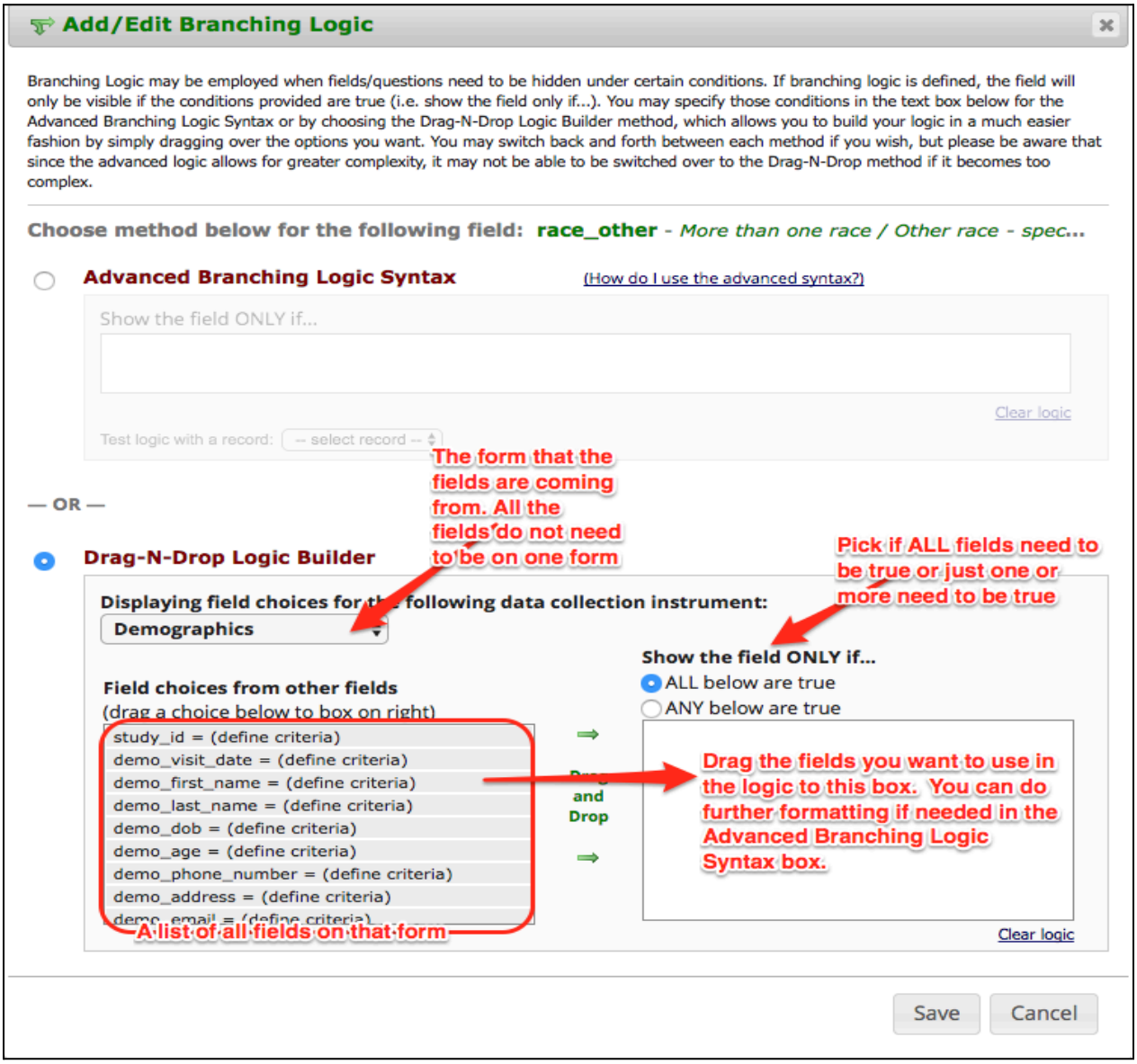
The field for which you are constructing the Branching Logic will ONLY be displayed when its logic has been evaluated as TRUE. Please note that for items that are **coded numerically, such as dropdowns and radio buttons, you will need to provide the coded numerical value in the equation** (rather than the displayed text label). See the examples below.

[sex] = "0"	Display question if sex = female (Female is coded as 0)
[sex] = "0" and [given_birth] = "1"	Display question if sex = female (Female coded as 0) and given birth = yes (Yes is coded as 1)
([height] >= 170 or [weight] < 65) and [sex] = "1"	Display question if (height is greater than or equal to 170 OR weight is less than 65) AND sex = male (Male is coded as 1)
[last_name] <> ""	Display question if last name is not blank (if last name field has data)
[visit_date] =””	Display question if visit date is blank (if visit date has no data)

[race] = “6”	Display question if race = other (Other coded as 6)
[race(6)] = “1”	Display question if ‘Other race’ is checked (Checkbox field type)

When you click on the **green double arrow icon**, you will be taken to the screen below. It is here that you will format the branching/skip logic.





Is it possible to use branching logic to skip an entire section?

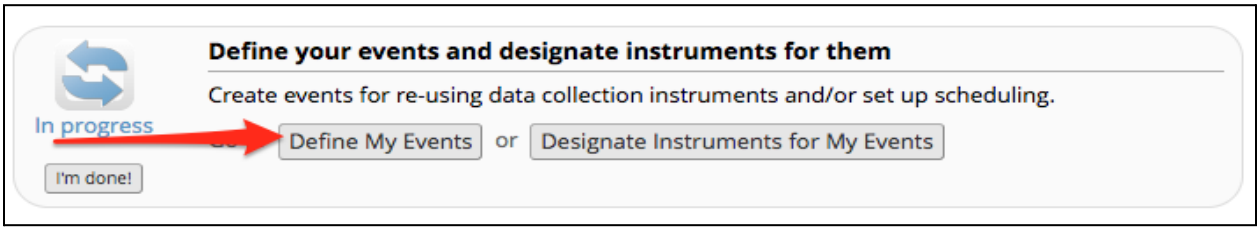
No, branching logic must be applied to *each field*. It cannot be applied at the form or section level. Section headers will be hidden only if **all fields in that section are hidden**.

Is it possible to use branching logic to skip an entire form or forms?

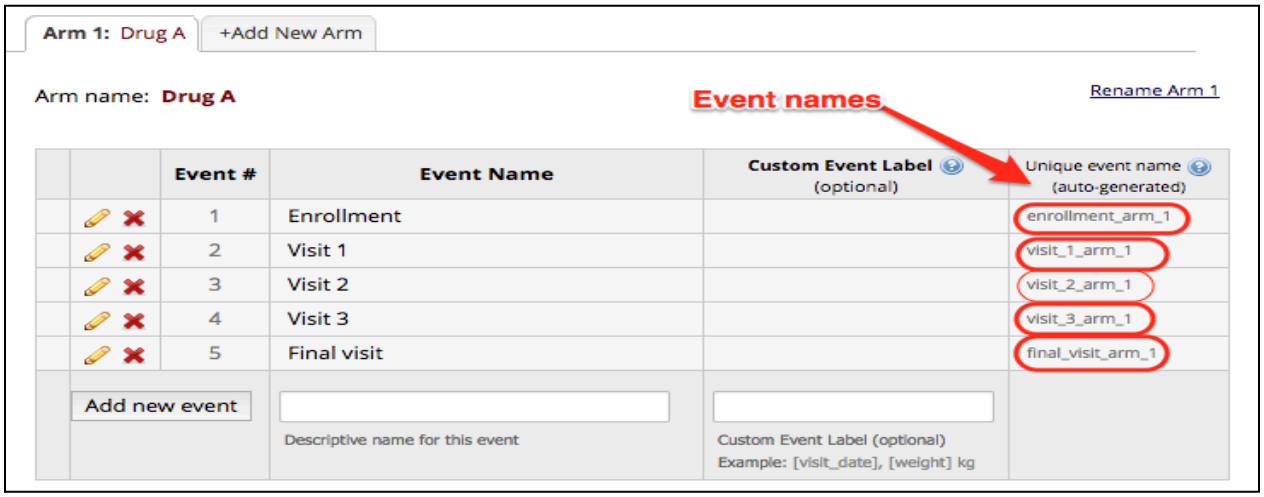
Branching logic will only hide questions, not entire data collection instruments. If you need to hide an entire form unless a certain condition is met, you would use **Form Display Logic** found on the Online Designer page.

Can fields from different EVENTS be used in branching logic (longitudinal only)?

Yes, for longitudinal projects (i.e. with multiple events defined), branching logic may utilize fields from other events (i.e. visits, time-points). The branching logic format is somewhat different from the normal format because the **unique event name must be specified in the logic for the target event**. The unique event name must be prepended (in square brackets) to the beginning of the variable name (in square brackets), i.e. **[unique_event_name][variable_name].** Unique event names can be found listed on the project's *Define My Event's* page on the right-hand side of the events table, in which the unique name is automatically generated from the event name that you have defined (see below).



(Project Setup page)



(Define My Events page)

For example, if the first event in the project is named "Enrollment", in which the unique event name for it is "enrollment_arm_1", then we can set up the branching logic utilizing the "weight" field from the Enrollment event: ([enrollment_arm_1][weight]/[visit_weight] > 1). Thus,

presuming that this field exists on a form that is utilized on multiple events, it will always perform the branching logic using the value of weight from the Enrollment event while using the value of [visit_weight] for the current event the user is on.

Is branching logic for checkboxes different?

Yes, special formatting is needed for the branching logic syntax in 'checkbox' field types. For checkboxes, simply add the coded numerical value inside () parentheses after the variable name:

[variablename(code)]

To check the value of the checkboxes:
“1” = checked
“0” = unchecked

See the examples below, in which the 'race' field has two options coded as '2' (Asian) and '4' (Caucasian):

[race(2)] = "1"	Display question if Asian is checked
[race(4)] = "0"	Display question if Caucasian is unchecked
[height] >= 170 and ([race(2)] = "1" or [race(4)] = "1")	Display question if height is greater than or equal to 170 AND Asian OR Caucasian is checked

Can I use branching logic with repeating forms?

Yes, piping, branching logic, and calculations can be used with repeating forms.

If you only use the variable name, REDCap will assume that you are referring to the variable on your current instance.

To refer to the variable on a specific instance, you can append the instance number after the variable name. For example: [variable1][5] will reference the data in variable1 on instance five.

To refer to the variable on a relative instance, there are several smart variables. They all append after the variable name.

- [current-instance] refers to the current instance
- [previous-instance] refers to the instance immediately previous
- [next-instance] refers to the instance immediately after
- [first-instance] refers back to the very first instance
- [last-instance] refers back to the very last instance

For example, to reference the data in [variable2] on the previous instance, you would use [variable2][previous-instance].

Will the @DEFAULT action tag work on fields hidden by branching logic?

No, the @DEFAULT action tag is incompatible with branching logic. REDCap will populate any default value to @DEFAULT'ed fields at the point of render. If that field with that value should not appear due to upstream branching you will receive an error message.

Please note: If the action tag '@DEFAULT= ' is used on a branching logic hidden field, the action tag does fire on page load but it does throw an error message.

Adding Branching Logic Using the Online Designer

For basic branching, you can simply drag and drop field names as needed in the Branching Logic dialog box in the Online Designer. If your branching logic is more complex, you will create equations using the syntax described below.

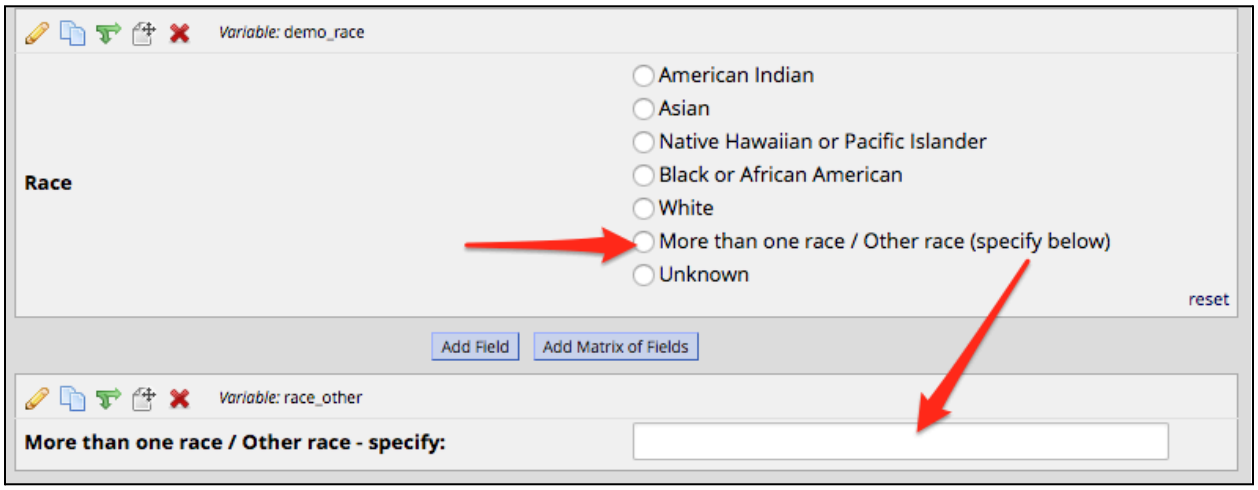
In the equation you must use the project variable names surrounded by **[ ]** brackets. You may use mathematical operators (=, <, >, <=, >=, <>) and Boolean logic (and/or). You may nest within many parenthetical levels for more complex logic.

You must **ALWAYS** put single or double quotes around the values in the equation UNLESS you are using > or < with numerical values.

Make sure the parentheses () are used correctly to bundle conditions together:

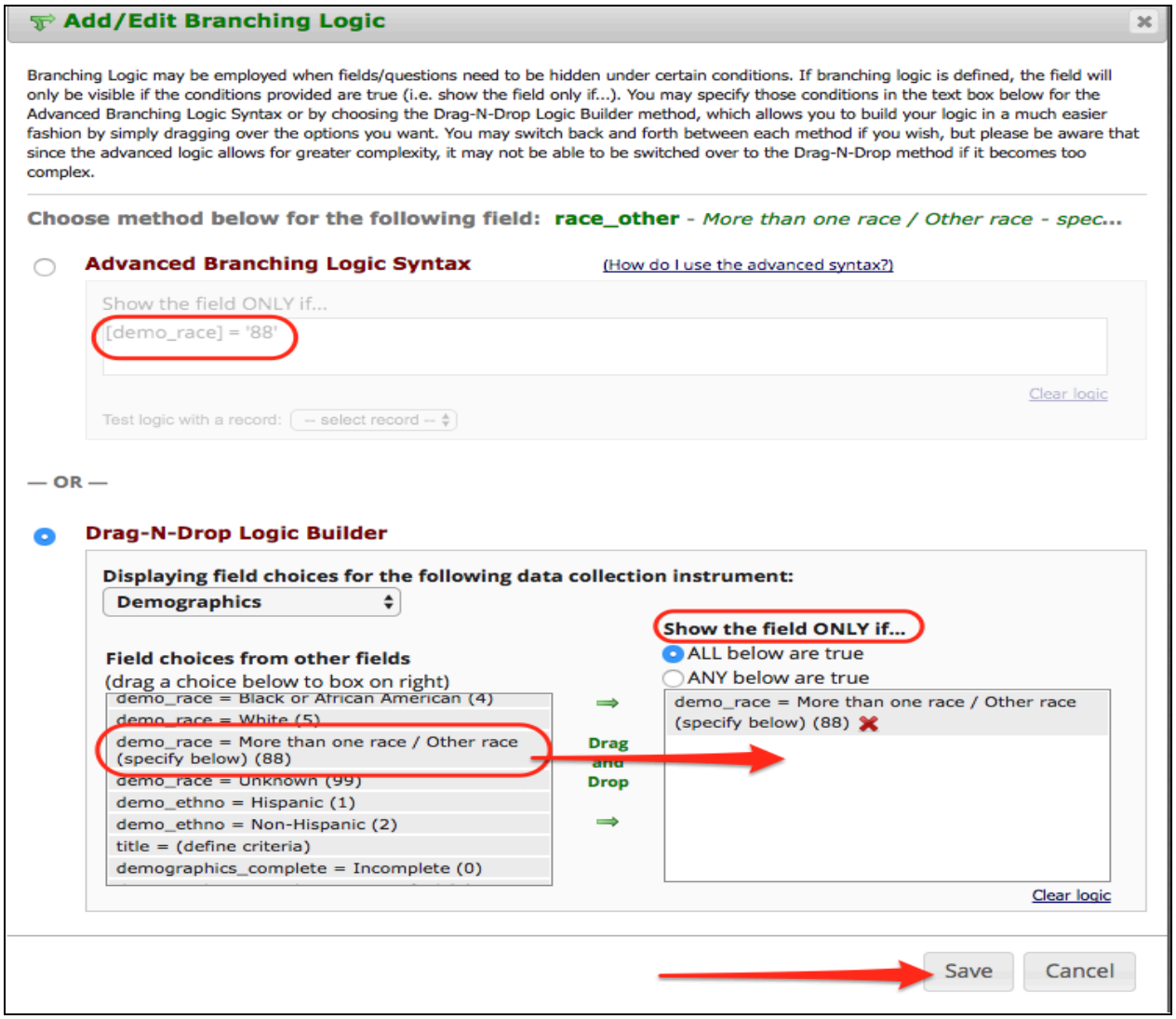
> **(**[gender] = '1' **and (**[age] > 10 **and** [age] < 50**)) or (**[gender] = '2' **and (**[age] > 14 **and** [age] < 55**))**

In the example below, I want to add branching logic to the "More than once race / Other race – specify:" question so that it only appears if someone selects "More than one race / Other race (specify below)."



To add branching logic, click on the double green arrow above the question you want to add logic to. **The logic gets applied to the field you want to hide ("More than one race…"), not the 'parent' field ("Race").**

There are two methods of applying branching logic to questions, the **Advanced Branching Logic Syntax** and the **Drag-N-Drop Logic Builder**. You would use the Advanced Branching Logic Syntax for more complex branching logic (i.e., >=, <=, <>"", ="") and the Drag-N-Drop Logic Builder for simple branching logic (i.e., and/or, =). Even if using the advanced method, it can be helpful to drag the data fields you want to use into the "Show the field ONLY if…" box and then format the logic in the Advanced section.



(Add/Edit Branching Logic page)

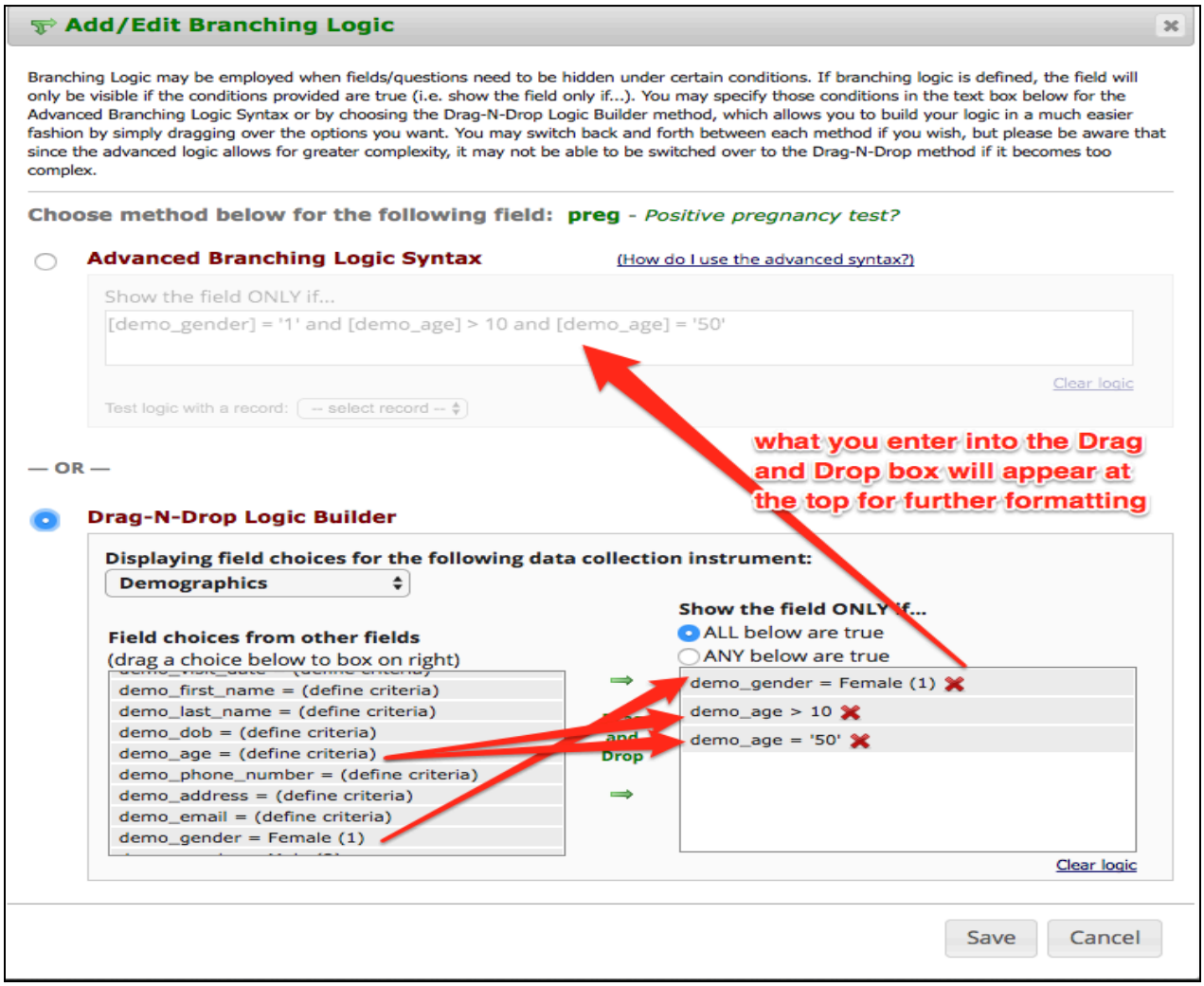
In the example above, the [race_other] field will only appear if [demo_race] = '88' is selected ('88' = "More than one race / Other race").

Show the field ONLY if…

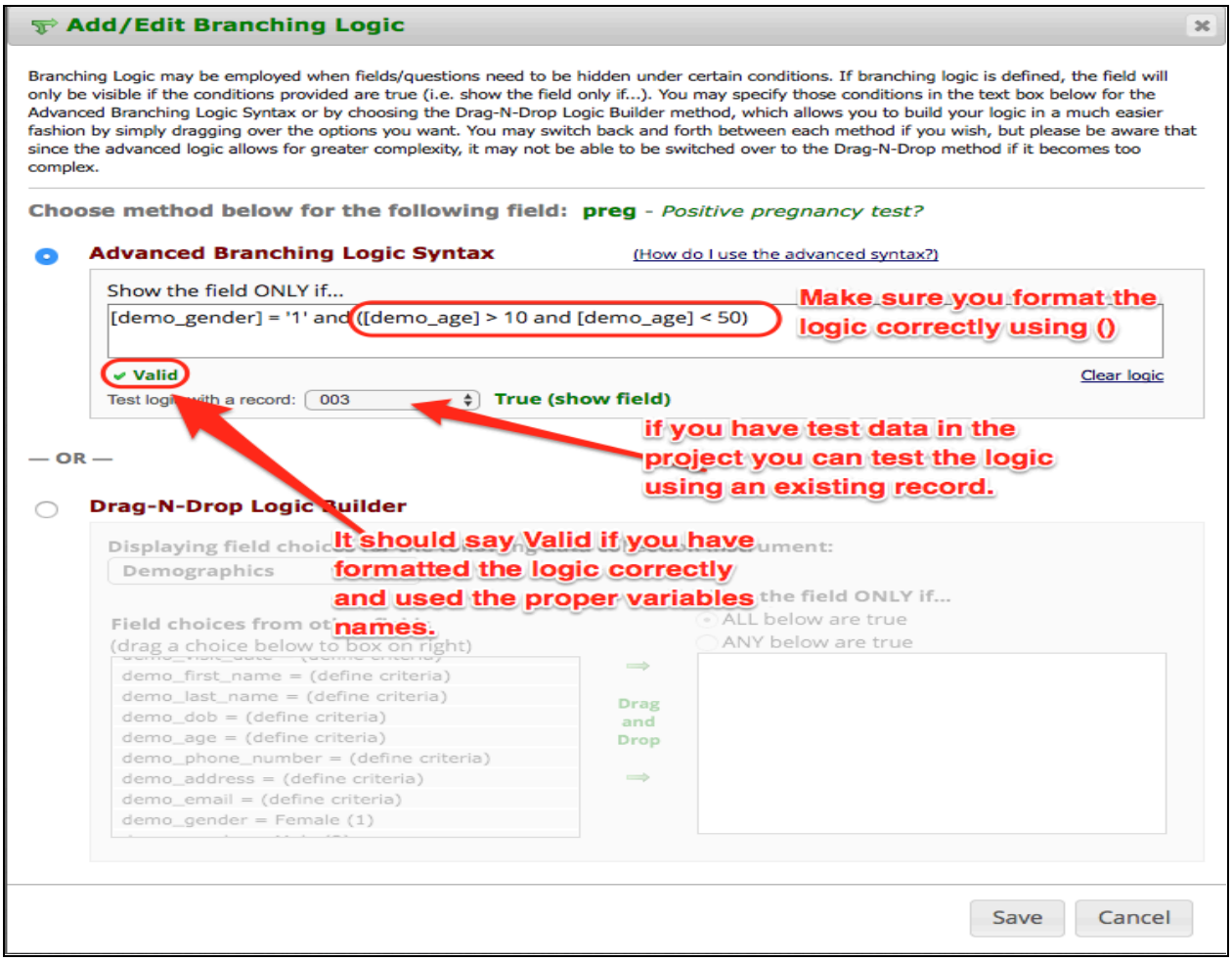
"**ALL below are true**" – EVERY field in the syntax must be true to show the field. For example, if you wanted a field to show up the subject is Female ([gender] = '1') AND they are more than 10 years old ([age] – text field validated as a number), the branching logic would be:

[gender] = '1' **and** [age] > 10

if you wanted a field to show up the subject is Female ([gender] = '1') AND they are more than 10 years old AND less than 50 years old ([age]), the branching logic would be:
[gender] = '1' **and (**[age] > 10 **and** [age] < 50**)** (see example below)



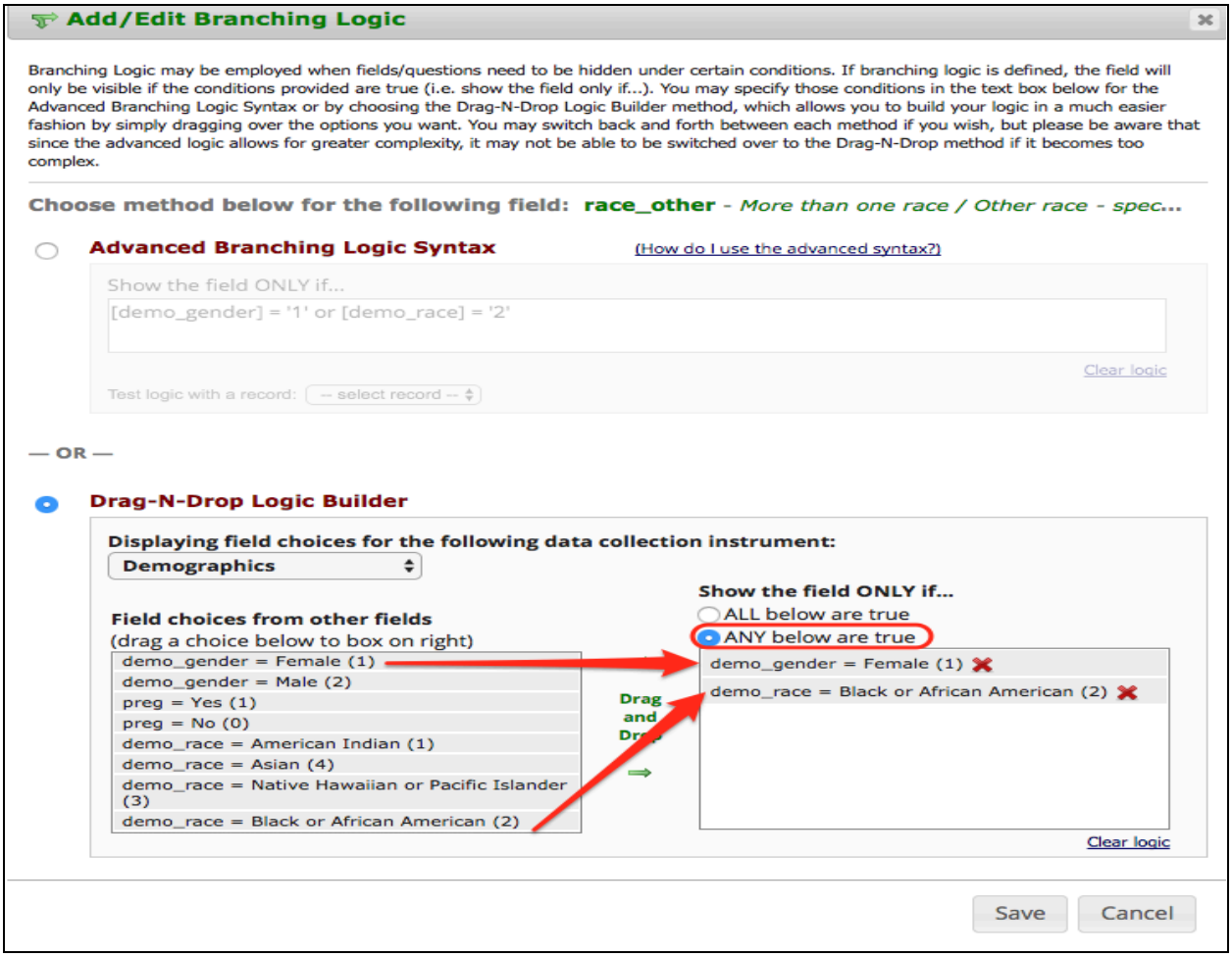
(In this example, I dragged over the gender and age fields so I can format them using Advanced logic.)



(I then formatted the logic in the advanced logic syntax box)

"**ANY below are true**" – only ONE field in the syntax needs to be true to show the field. For example, if you wanted a field to show up if the subject is Female ([gender] = '1') OR if the subject is African-American ([race] = '2'), the logic would be:

[gender] = '1' **or** [race] = '2' (see example below)



Branching Logic Tips

You can use <, >, <=, >= symbols to make branching logic simpler for multiple choice fields.

For example, if you want a second field to appear if someone selects answers 1-4 but not 5 or 6 for the variable called [question].

1, Red
2, Orange
3, Yellow
4, Green — Show the field
5, Blue
6, Purple — Hide the field

You can format the branching logic to be:

Show field if: **[question] < 5** (the answer value of [question] is less than 5)

---versus---

Show field if: **[question] = ‘1’ or [question] = ‘2’ or [question] = ‘3’ or [question] = ‘4’**

When you have a list of fields, such as medications where someone may be on up to 10 medications, but you do not want all 10 entries to appear on the page, just the ones that will have data, you can use the <>”” operator to format this.

Fields	**Branching logic – show field if:**
Medication 1 Name	
Medication 2 Name	[med1] <>”” (medication 1 is not blank)
Medication 3 Name	[med2] <>”” (medication 2 is not blank)
Medication 4 Name	[med3] <>”” (medication 3 is not blank)
Medication 5 Name	[med4] <>”” (medication 4 is not blank)
Medication 6 Name	[med5] <>”” (medication 5 is not blank)
Medication 7 Name	[med6] <>”” (medication 6 is not blank)
Medication 8 Name	[med7] <>”” (medication 7 is not blank)
Medication 9 Name	[med8] <>”” (medication 8 is not blank)
Medication 10 Name	[med9] <>”” (medication 9 is not blank)

Additional Resources:
REDCap Branching Logic Course

REDCap Branching Logic

REDCap Beginning Branching Logic