Market Research Surveys

CHAPTER 1

Web Surveys for Market Research

Market Research Surveys are similar to any other Web Survey, except that they need more advanced capabilities to deal with things such as ensuring you have a balanced sample, and minimizing bias within the Survey Data.

Correctly managing these surveys requires a powerful Web Survey tool specifically designed to handle these advanced capabilities.

SECTION 1

Introduction

IN THIS SECTION

Challenges facing Market Research Professionals doing Web Surveys

Market Research professionals are faced with a rapidly changing landscape for Web Surveys where clients are expecting surveys that are more engaging, and that can be completed on a large array of devices.

CREATING COMPLEX WEB SURVEYS EFFICIENTLY

As complexity has increased, Web Survey software tools have had to meet the challenges faced in efficient ways to ensure that the creation of MR surveys is still within the capabilities of non-technical users.

Web Surveys are now expected to:

1. Support “flashy” interfaces that engage respondents
2. Allow advanced functionality like drag and drop for responses, rather than more traditional choice questions and grids
3. Provide complex validation based upon multiple rules
4. Manage respondent quotas so that a representative group of respondents can be found
5. Provide real-time results to stakeholders through interactive online portals

It is important for MR professionals to be able to meet all these needs without excessive increases in their time - after

all, regardless of anything else clients always want things cheaper as well!

Avoiding the Software "tax"

There are a number of players in the Market Research arena - many of them quite large. They are used to dealing with the big end of town where upfront costs *and* per response costs of $1 or more are the norm.

This software "tax" hurts your bottom line every time you try and do something. It makes good sense to try and find a system that works well, and has unlimited responses, or low per-response costs.

Need for high quality, unbiased responses

Collection of accurate, high quality data is the most critical goal for any market research survey. As researchers, we are constantly fighting against:

- Apathetic & lazy respondents
  e.g. no care in answering, speedsters rushing to the reward
- Mis-aligned respondent goals
  e.g. some just want the biggest reward for their response
- Inappropriate respondents
  e.g. the wrong person completing a response
- Fatigued respondents
  e.g. a survey is too long to maintain respondent interest

While completely eradicating these problems is a tall order, there are a number of things that can be done to minimize their occurrence.

The next section explains the features of a Market Research Survey that can be used to minimize bias.

Need for a balanced group of Respondents

It is often important to ensure that your respondents match a broad section of the community. Failure to do so could lead to results that appear to be valid, but are in fact seriously affected by the people that have been responding.

An extreme example of this would be asking the following question:

Which of the following did you play with when you were a child?

1. Toy Trucks
2. Barbie Dolls
3. Make-up

This question has a substantial gender bias, and is likely to produce wildly different responses depending upon the gender of the respondent.

Quota management is used for balancing respondents to a survey - this is the subject of our next chapter.

KEEPING UP WITH THE MOVE TO MOBILE

One of the biggest changes in recent years is the move to surveys that are responded to on mobile devices. In less than two years, allowing respondents to complete their surveys on their mobile phone or tablet has gone from a “nice to have” to a “must have”.

Often, the best place to catch a respondent is on their phone. They may not be willing to sit at their computer and answer a survey because there are things they would prefer to be doing, but if they are on a bus or train, or just sitting around, a survey on their mobile might even be a welcome distraction.

No survey package can be considered in this market unless one of it’s key features is high quality mobile survey delivery.

INTEGRATION WITH OTHER SOFTWARE PACKAGES

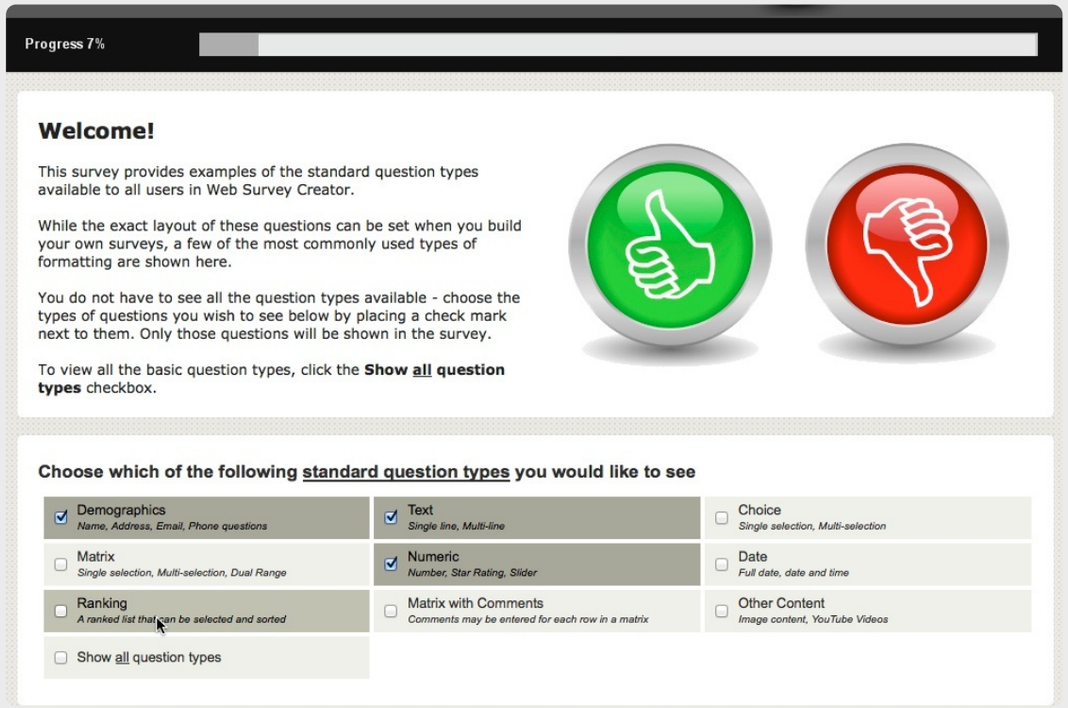
A survey is often the middle of a complete process. For example, the process may start with an invitation sent out from a Panel Management tool, and may end with an export to a statistics package. It is important that the Web Survey tool chosen can support this sort of integration.

Recommended Solution for MR Surveys: Web Survey Creator

This book talks about the creation of Market Research Surveys from the perspective of users of Web Survey Creator. This software solves all the problems described in this section and much more.

For further details about this product, or to download the *free version* of the software, visit the Web site, which can be found at http://www.websurveycreator.com.

GALLERY 1.1 Samples of Surveys Created with WSC



General question type sample
http://www.websurveycreator.com/c/survey-sample-1.aspx

SECTION 2

Dealing with Bias

IN THIS SECTION

In many ways, Market Research Surveys are just like any other survey. You need to ask questions, and get responses to those questions. As discussed in the previous section, there are specific problems that must be dealt with when performing market research that require added functionality beyond a basic survey tool.

We will have a look at some of the fundamental functionality needed for market research in this chapter. We will demonstrate this functionality in Web Survey Creator.

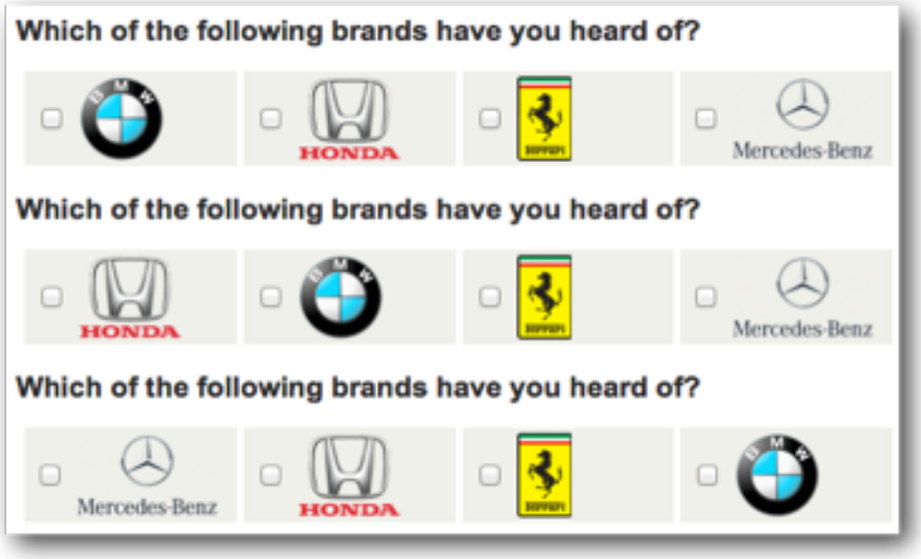
You can not “convince” a respondent to be unbiased. What surveys have to do is balance the effect of bias so that it is effectively “cancelled out” when looking at the data as a whole.

Randomization simply refers to “shuffling” survey content so that it will appear in a different order for different respondents. This will give content an equal chance of being considered by a respondent, and spreads the effect of:

1. Questions and choices earlier in a survey being given more consideration (before a respondent’s attention drifts)
2. Less care and thought being taken by a respondent the longer a survey continues (as they become fatigued with the whole process)

Choice Randomization

A choice question with randomized choices will “mix up” the choices differently for each respondent. When randomized, the same question may appear as follows:



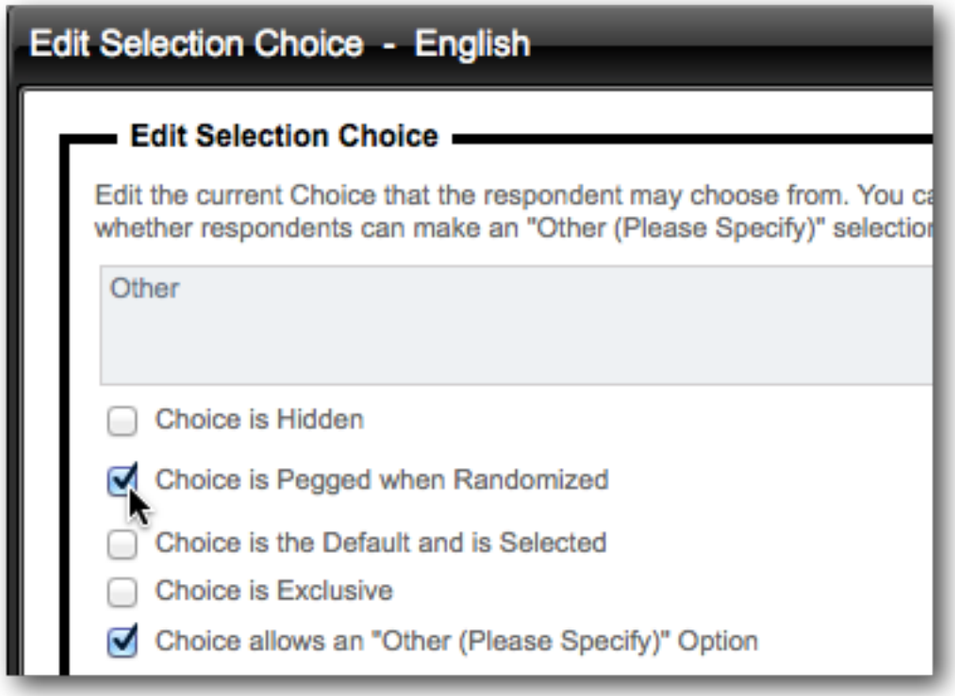
Pegging Choices

There are some situations when you don’t want to randomize every single item in the list.

If we randomize this question, it could look as follows:

This doesn’t really work - having the choices “other” and “don’t like fruit” in the middle of the list is confusing, and off-putting (particularly since the “don’t like fruit” option is exclusive and disables the other values).

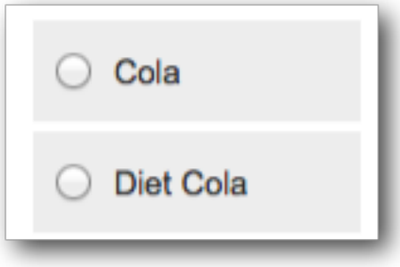
Fortunately, Web Survey Creator has a simple solution to the problem - individual values can be “pegged”.



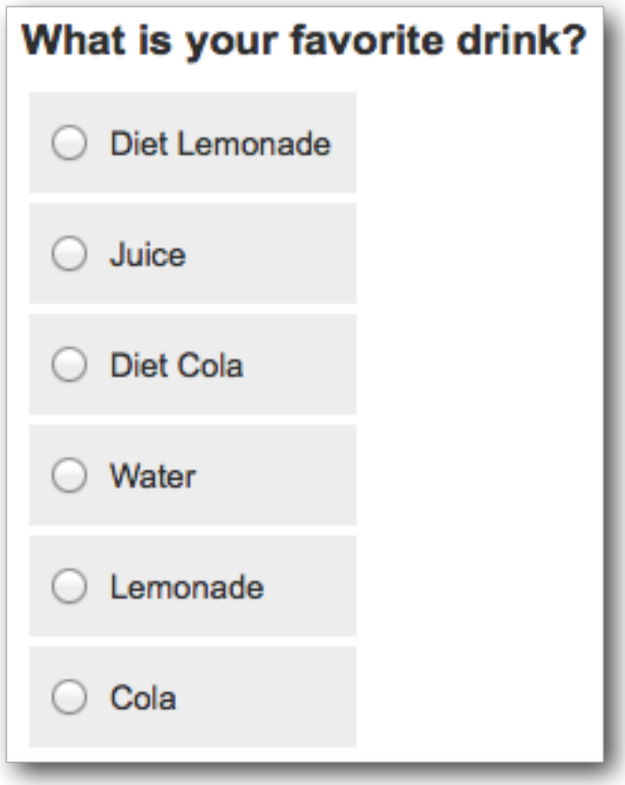
Pegged choices will not change their position - other choices are randomized around them.

Grouping Related Choices

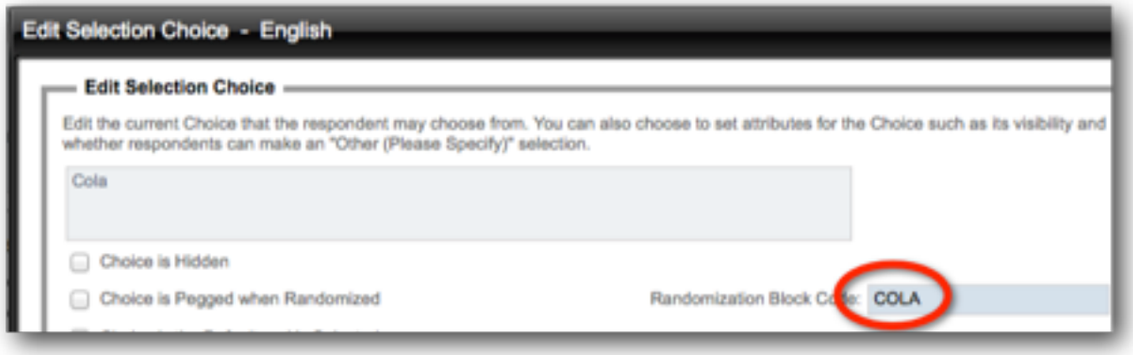
Another issue with randomization is where you want some choices to be kept together within the random choice list. These would be related choices, such as:



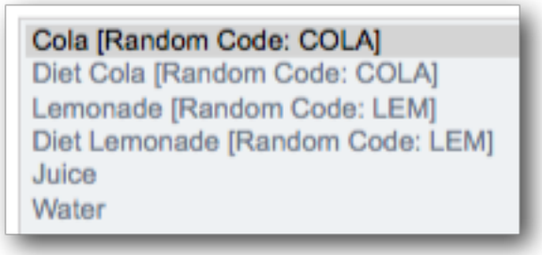
Even when randomizing all choices, you may want to keep the Cola drinks together so that Cola lovers can easily choose between diet and non-diet variants. What we ant to avoid, is a question that looks something like this when randomized:



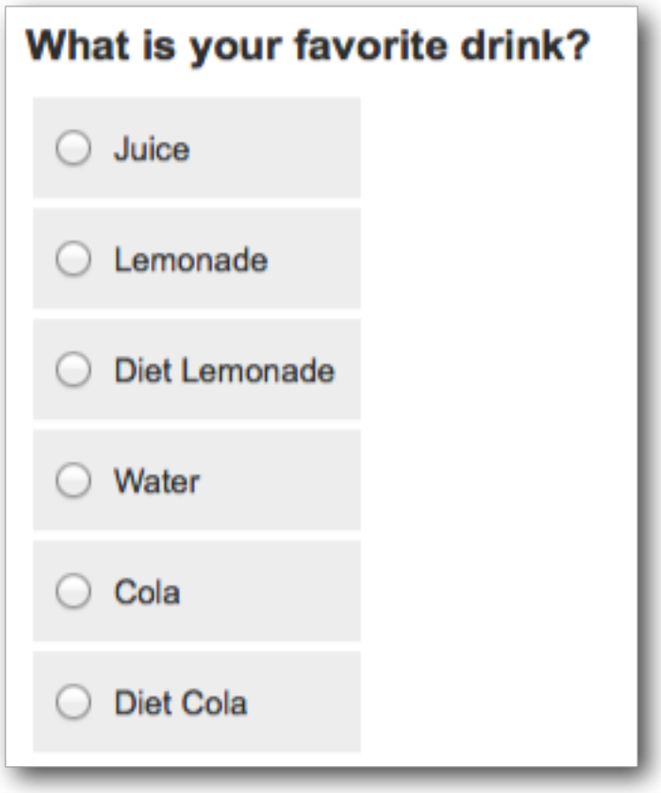
In Web Survey Creator, consecutive choices can be kept together by using the same *Block Code* for the choices.



If we use a block code “COLA” for the Cola drinks, and “LEM” for the Lemonade, we can then randomize the choices and the related drinks will always be kept together.



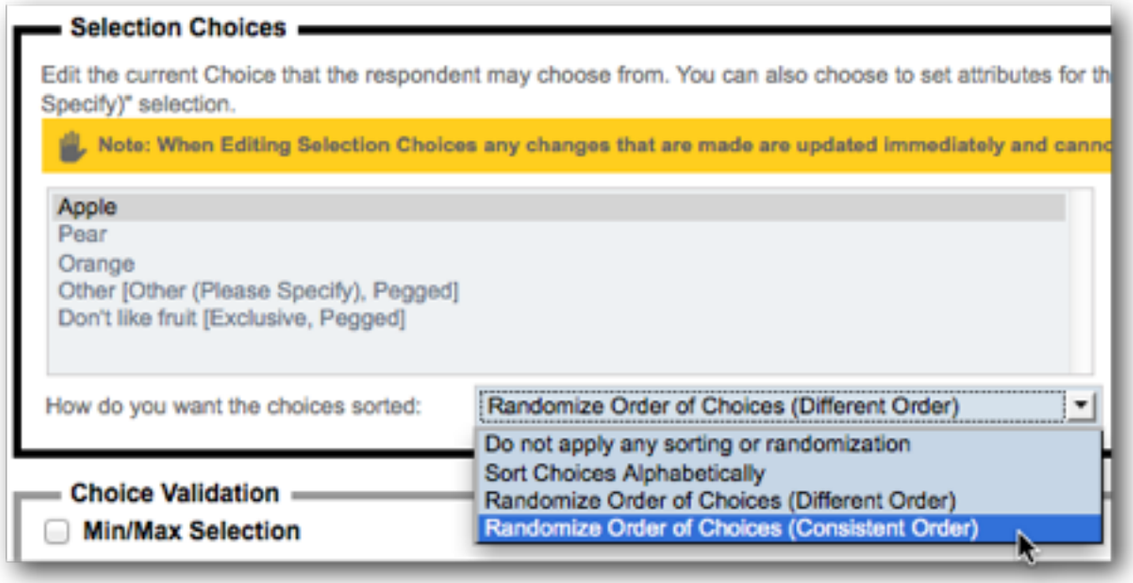
A example of how the question might look is as follows:



Randomizing Consistently

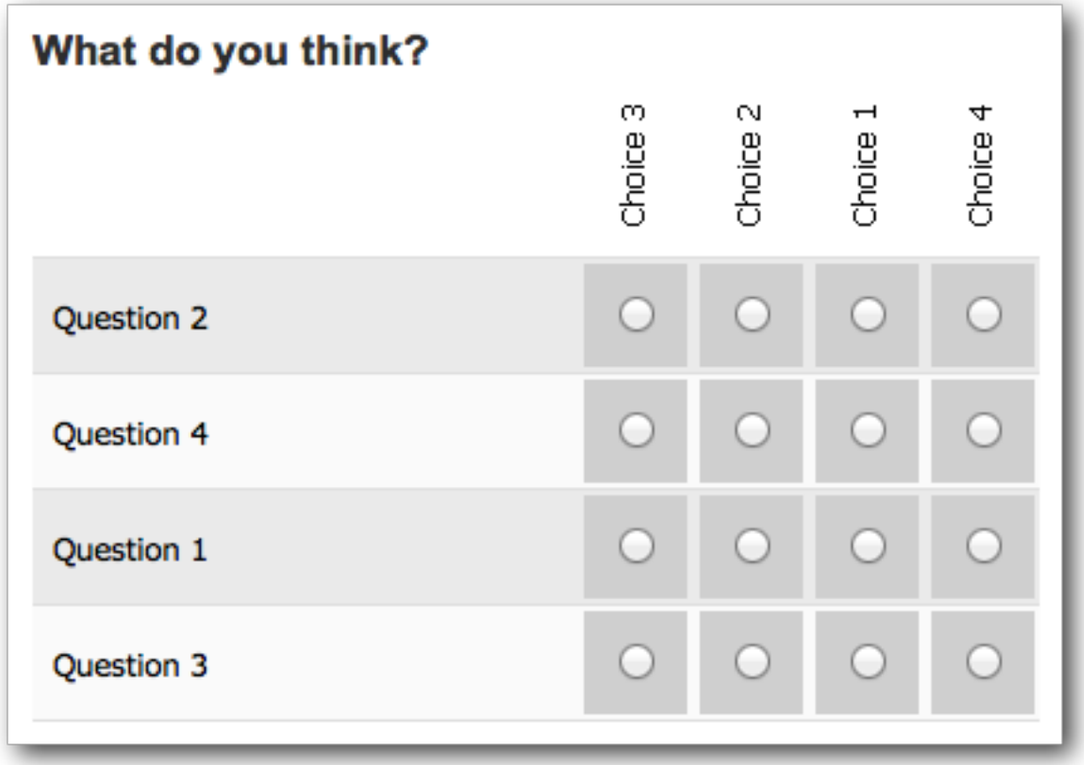
It sounds strange, but “consistent randomization” can be important in a survey. For example, if you are asking multiple questions about a series of products, you may want the products to be randomized *in the same way* for every question. This is consistent randomization.

Web Survey Creator allows you to pick consistent randomization as one of the options when you turn randomization on.



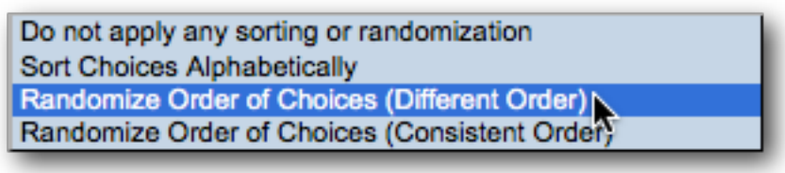
Matrix Randomization

Rows and columns in a matrix can be randomized in exactly the same way as choices.



Matrix Columns

Columns in a matrix are exactly the same as choices in a choice question, and randomization is the same.

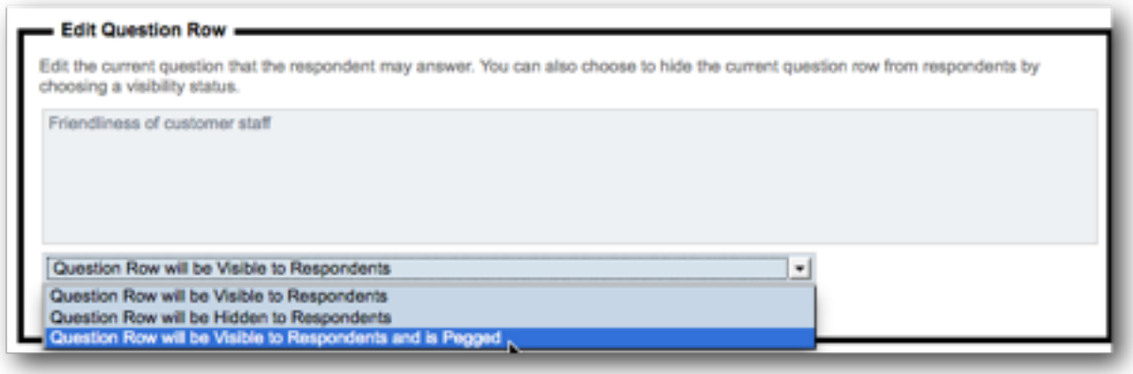


Columns can be pegged and grouped as well.

Matrix Rows

The rows of a matrix are usually questions or statements. You may wish to randomize their order so that people aren't "over it" every time they hit the same last questions in the matrix.

The randomization options are the same as for choices, though the way you peg a row is slightly different - you choose pegging from a drop-down list of options (which include hiding the question completely).



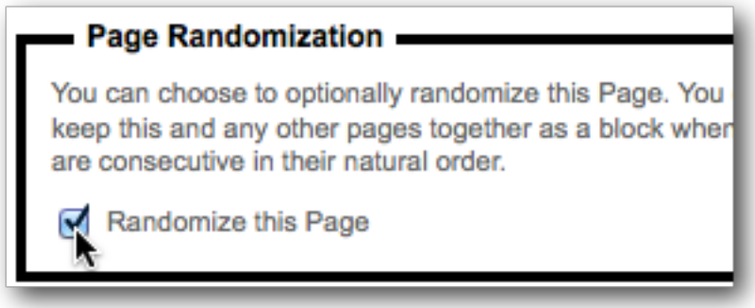
Page Randomization

Page randomization provides the most dramatic way of randomizing survey content. You can change the order of consecutive pages in the survey randomly for each respondent using this capability.

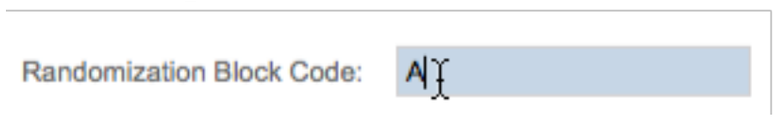
To use page randomization, you need to:

1. Edit the first page you want to randomize

2. Check the randomization check box



3. If you want to group pages together in the random order, enter an optional Block Code



4. Save the page

5. Repeat for each subsequent page you wish to randomize.

Tips for Page Randomization

Page randomization can have a big effect on the flow of your survey, and therefore is the most "destructive" method of randomization if you get it wrong. It is therefore important to be careful when implementing this type of randomization.

We have a couple of tips to ensure you get the most from page randomization.

Tip 1: Ensure Your Random Pages Are Consecutive

The system will randomize all pages that are consecutive and flagged as random. You can not have a non-random page in the middle of a series of random pages and expect the random pages to mix - you will end up with two groups of random pages - either side of the non-random page.

Tip 2: Don't Create Impossible Logic!

Whenever you randomize a page, you get an explicit warning:

It is up to you to ensure you don't create a situation where the page order created through randomization. Consider the following example:

Page 1

Do you smoke?

Page 2 (Hide for people who don't smoke)

Are you looking to quit smoking?

We could randomize these pages, but it would be a very bad idea. One of the random orders of pages would be:

Page 1 (Hide for people who don't smoke)

Are you looking to quit smoking?

Page 2

Do you smoke?

Flow control will not work in this instance, because a page that is hidden by flow control appears before the question used in the flow control is even asked...

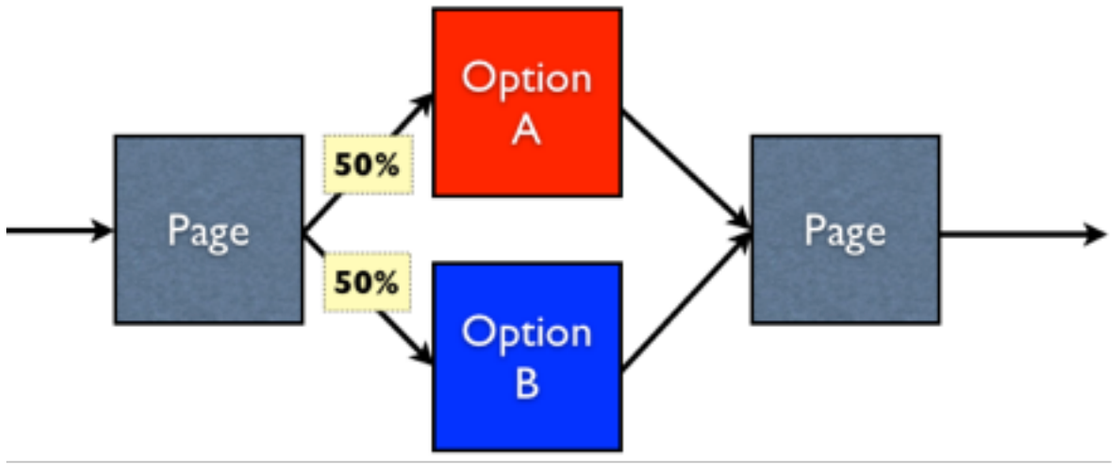
Tip 3: Test Thoroughly

Random pages are unpredictable by definition. It is important to enter test responses multiple times so you can try many variations of the page orders.

A/B Testing

Users of A/B testing will distribute multiple samples of a test to see which single variable is most effective in increasing a response rate or other desired outcome. The test, in order to be effective, must reach an audience of a sufficient size that there is a reasonable chance of detecting a meaningful difference between the control and other tactics.

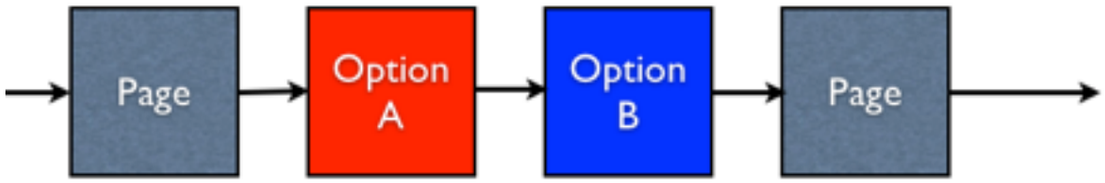
Web surveys are a great candidate to use A/B testing, since gaining access to a large audience is relatively easy. A simple example of an A/B test is shown below.



What we want is 50% of respondents to go to the page Option A, and 50% to go to the page Option B. Achieving this result is another example of randomization.

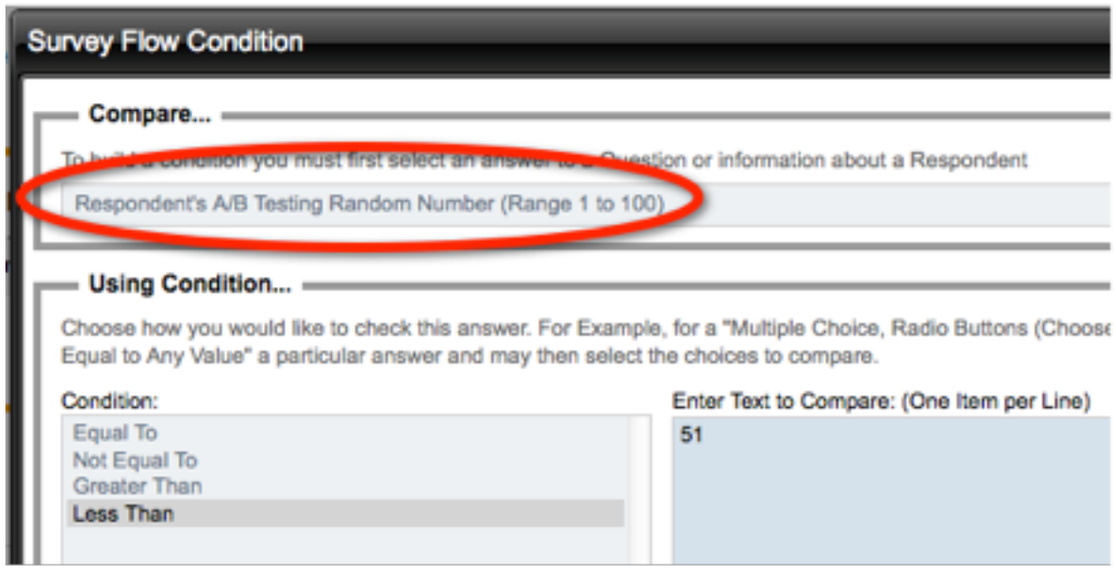
As soon as a respondent begins a response in Web Survey Creator, they are allocated a random number. This number can be used for various things - one of which is A/B testing. Setting up an A/B test is actually done through flow control.

The true structure of a survey that has the A/B test above would be:



The create our A/B test, we want to hide the Option A page 50% of the time, and hide the Option B page 50% of the time.

Hiding the option A page would require the following flow:



We use the **A/B testing Random Number** for the Respondent (which is always a number between 1 and 100) to do the flow. We want to hide the Option A page if the number is less than 51 (i.e. the number is between 1 and 50).

The hiding of the option B page would be the opposite test - hide the page when the A/B testing Random Number is greater than 50 (i.e. the number is between 51 and 100).

Of course this methodology wouldn't just apply to a two page test - you could have up to 100 options that are randomly chosen between, since the AB testing Random Number is equal to 1 of a possible 100 values.

Chapter 2

Quota Management

Quota management is a key element of most market research surveys.

By controlling the quotas for a survey, you can ensure you get a balanced sample, and minimize your costs for panel respondents by ensuring you only pay for the people you actually need.

Section 1

Quota Management Explained

In this Section

Why use Quotas?

The “More is Better” Rule

Many surveys work on the principle “the more respondents, the better” and therefore place no limits on how many people may answer the survey.

An example of such a survey would be a customer survey - you want as many customers as possible to complete the survey, and you don’t want to place any limitations on this.

The reason why this rule works for these surveys is that each respondent’s “voice” is as important as any other respondent. In our example, every respondent is a “customer”, and their views are equally important.

The “Balance is Better” Rule

Let’s consider a different example - let’s say we have a survey about what people like and dislike about a fast food brand. This may seem to be another perfect candidate to apply the “more is better” rule. This would be a mistake, and here’s why:

1. As a general rule, men prefer (and eat more) fast food than women
2. Younger men are likely to eat more fast food than older men (once their cholesterol and age catch up with them and they need to be more careful)

3. Middle-aged women and anybody over 60 are more likely to respond to a survey than people under 25 (who think Email is archaic, and surveys are a waste of their time)

OK, so what would this all mean if respondents are left to their own devices?

People who like fast food the least would make up a disproportionate number of survey responses - women and older men. Extrapolating this to say something about the community as a whole is meaningless. Young people who love fast food won't be represented sufficiently to provide a balance.

The answer is to provide a balanced group of respondents, by only taking a set number of people from various sections of society. Once you have enough answers from middle aged women, you simply don't need any more. You need to go out and find a sufficient number of young males!

A "Quota" can be based on anything you like, but commonly is based upon:

- Age
- Gender
- Location

If you take these three things into account when considering your respondents, you can make the respondent makeup mimic the makeup of society, and therefore provide a more accurate view of society's views.

How can Quotas be managed?

Early Survey Termination

You can not control who will click through to your survey, and when they will do it. This means that an unbalanced sample of people are likely to start the survey.

> *The goal of a quota management system is to stop respondents you don't want at the earliest possible point.*

Being terminated in a survey will be at least mildly annoying to someone who has taken the time to complete the survey. The last thing you want to do is inflame the situation by making them answer a lot of questions before they are terminated.

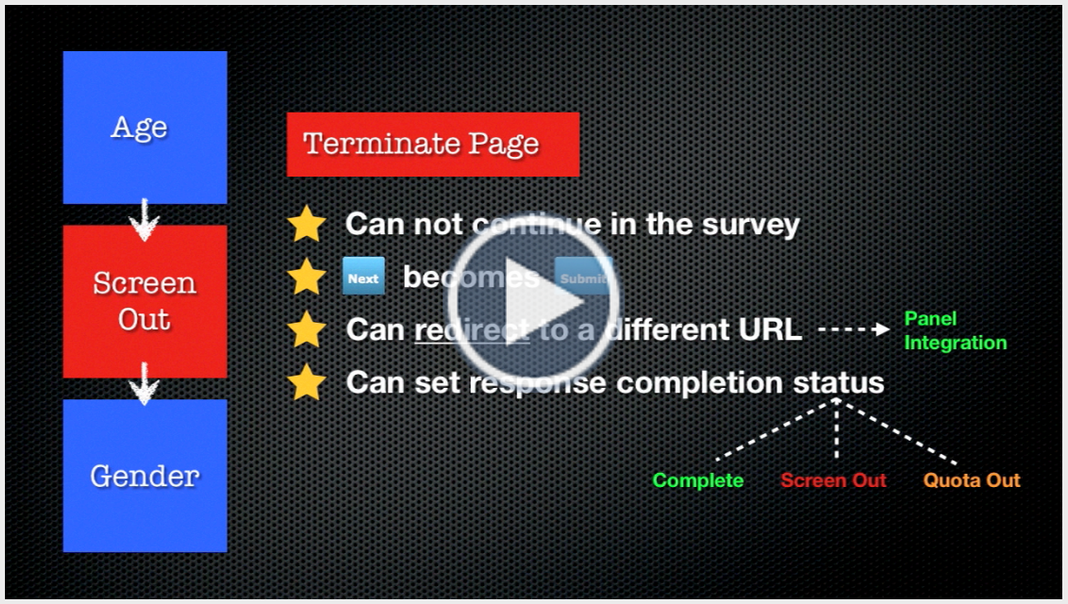
Best practice when creating Web Survey for Market Research is to have a survey with the following structure:

1. All questions to determine whether a respondent is suitable for the survey are asked in the first few pages of the survey
2. **Terminate Pages** for "Screen-Outs" (people who you are not interested in) and "Quota-Outs" (people you are interested in, but you want to limit the numbers) appear next.

3. Flow control is used to skip appropriate people over the terminate pages. Everyone else hits one of the terminate pages, and the survey is over.

Terminate pages are simply pages that end the survey - they don't have a *Next button* like other pages - they simply have a *Submit button*.

MOVIE 2.1 Terminate Pages in WSC



This video explains terminate pages in WSC
http://vimeopro.com/websurveycreator/wsc-training-videos/video/31361203

SETTING QUOTA RULES

Quota management all comes down to using a set of rules to determine whether someone should be completing a survey. Each quota basically deals with 2 things:

1. Who are we looking for? (eg. Males, 18-25)
2. How many of them do we want as respondents in our survey?

Web Survey Creator keeps track of the rules that have been created, and the number of people who have met each quota, in its Quota Management System.

Quota Title	Limit	In Quota	Remaining
City - Adelaide	51	51	-
City - Brisbane (inc Gold Coast)	100 (20)	118	2
City - Melbourne	152 (28)	185	-
City - Perth	62	50	12
City - Sydney	157	162	-
Female, 16-24	33 (7)	36	4
Female, 25-29	36 (19)	53	2
Female, 30-39	48 (27)	76	-
Female, 40-49	43 (22)	67	-
Female, 50-54	25 (15)	33	7
Female, 55-64	37 (13)	48	2
Female, 65+	46	34	12
Male, 16-24	34	23	11
Male, 25-29	36	25	11
Male, 30-39	47 (13)	60	-
Male, 40-49	41 (20)	46	15
Male, 50-54	25	18	7
Male, 55-64	36	22	14
Male, 65+	38	25	13

Dealing with Tough Quotas

There are often particular quotas that are harder to fill than others. A particular survey may be considered worthless if all quotas are not filled, so this can be a serious issue.

If we consider our previous example, we can see where the harder quotas are:

Quota Title	Limit	In Quota	Remaining
City - Adelaide	51	51	-
City - Brisbane (inc Gold Coast)	100 (20)	118	2
City - Melbourne	152 (28)	185	-
City - Perth	62	50	12
City - Sydney	157	162	-

We can see that while there are plenty of people in most of the other location quotas, we are still looking for 12 people from Perth.

Tough quotas are normally dealt with by "opening up" a survey until we get enough of the quota. This means going over quota in other areas so that we can at least get to the minimum level needed for the hard quota.

Opening up a survey is a very rough technique that has to be manually handled.

Fortunately, Web Survey Creator provides two features that make this process a whole lot easier and more controlled - quota overflow, and priority quotas.

Quota Overflow

In the event original quota numbers have to be modified, an "overflow" figure can be entered rather than changing the original figure - making it a lot easier to see what changes we have made.

Quota Title	Limit	In Quota	Remaining
City - Adelaide	51	51	-
City - Brisbane (inc Gold Coast)	100 (20)	118	2
City - Melbourne	152 (28)	185	-
City - Perth	62	50	12
City - Sydney	157	162	-

In our example, Brisbane has an overflow of 20 respondents (on top of the original quota of 100) and Melbourne has an overflow of 28 respondents (on top of the original quota of 152).

"Priority" Quotas

Quotas in Web Survey Creator work as follows:

1. As soon as all questions relating to quotas are completed, the system allocates every quota that a respondent meets to them. For example, a respondent may have the following quotas attached to him:

2. Based on our example, we already have too many male 30-39 respondents

Therefore this respondent would fail quota - *even though we really need people from Perth* - because letting them through would mean we end up with too many 30-39 year old men.

There are times when having one quota go over target is less of a problem than not getting enough of another quota.

If we wanted to make sure we let someone from Perth through, we could set our Perth quota as a "priority" quota. This tells the system to allow anyone from Perth through as a respondent, even if their other quotas are full. As soon as Perth is full, the priority quota will no longer be in effect.

Priority quotas make it very easy to focus on your hard-to-get people. They need to be used cautiously, however, as they will lead to other quotas being over-subscribed.

Tips for Quota Management

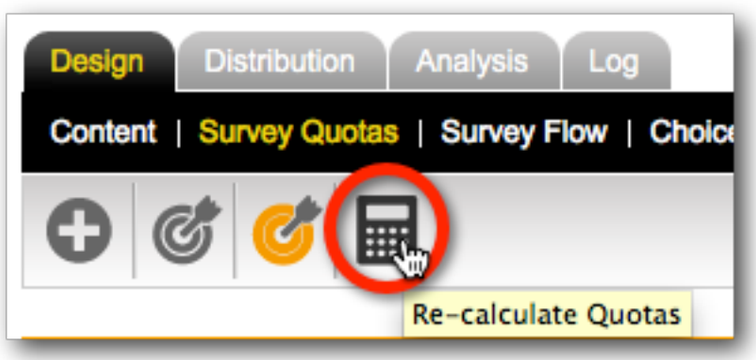
Adding quotas after a survey has commenced

The best advice that can be given about adding quotas after a survey has commenced (and responses have been received) is to avoid doing it! The main reasons to try and avoid the late inclusion of quotas are:

1. You will have completed responses from people who were never tested against your new quotas (since quotas are calculated as a person enters their survey response).

2. You risk making mistakes like inclusion of questions in quotas that appear after the quota out page (therefore making the quota out page logic invalid).

If you do add or change a quota after a survey has commenced, Web Survey Creator does help you play "catch-up" with existing responses - you can generate the correct quotas for existing responses by clicking the *Re-calculate Quotas* toolbar button under the *Survey Quotas* menu.



Testing whether someone fails the quotas in multiple places within a survey

There is nothing worse than a respondent almost getting to the end of a survey, and suddenly they are kicked out because their quota is full. Web Survey Creator avoids this situation by calculating quotas as early in the survey as possible, and from that point on there is a simple rule:

Once you're in, you're in!

What this means is if at the time the quotas were calculated there was still room for a respondent, they will be allowed to complete the survey. This may lead to a slight over-run of numbers for certain quotas if there are a number of people in that quota who finish around the same time.

A person doesn't raise the number in a quota until they have submitted their completed response, so the quota out won't be triggered until the needed number of people have completed their response.

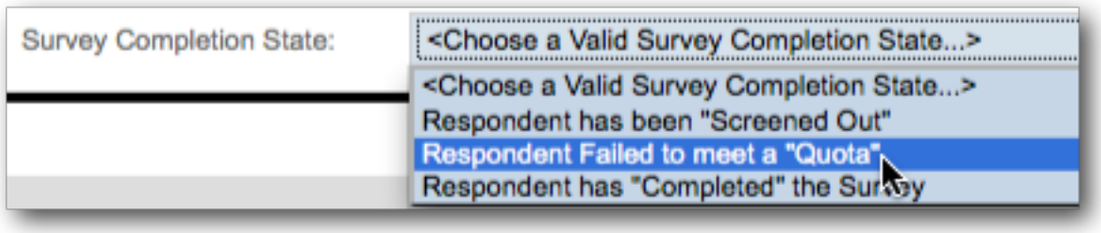
There may be times when this default behavior will cause too much of an over-run, or add unnecessary cost to a job. For example:

1. A survey may be really long, so a lot of people could be in the middle of the survey (beyond the quota test) when their quota is finally filled.

2. A significant number get past the quota test, and then leave their response in draft, to be completed later. No matter when they come back (even long after their quota is filled) they will be allowed to complete their response and add to the number in their quota.

So, how do we test for quotas more than once in a survey? Fortunately Web Survey Creator makes it simple.

If you add more than one *Quota Fail Terminate Page* to your survey, quotas will be retested prior to the second and subsequent terminate pages.



What this means is you can choose when a re-check occurs (if at all). If you don't care about annoying a respondent who has spent their time completing the answers to the survey, you could even have an additional quota fail page right near the end of the survey.

SECTION 2

Quota Management Example

IN THIS SECTION

Our Example Explained

Our example survey will be for a fictitious Fast Food chain - Worldburger. They have recently introduced a new “healthier choices” menu that adds healthier options to their menu, and they want to see whether people are aware of the change, and what they think of the change.

MAKEUP OF RESPONDENTS REQUIRED

In order to get a good cross-section of respondents, they have requested the following gender makeup for responses:

	16-24	25-29	30-39	40-49	50-54	55-64	65+
FEMALE	33	36	48	43	25	37	46
MALE	34	36	47	41	25	36	38

This will provide a good spread of ages, and a slight skew towards women (who are the primary focus of the “healthier choices” campaign).

An additional requirement is to have a geographical spread of respondents that roughly represents the proportional number of people in various cities who visit Worldburger stores in any given week.

SYDNEY	MELBOURNE	BRISBANE	PERTH	ADELAIDE
157	152	103	62	51

Both of these requirements add to a total of 525 people. They are not *interlocking quotas*, so it doesn't matter what the gender/age makeup is for each geographical location.

Preparing Our Survey

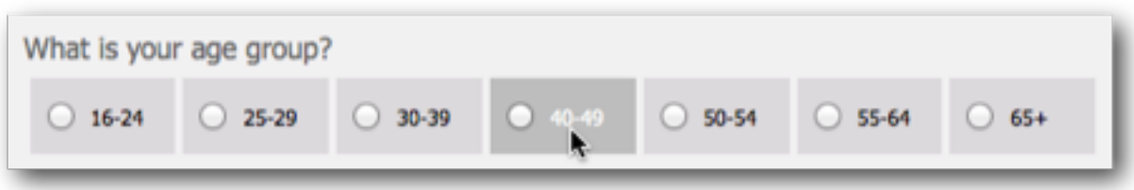
Gender/Age/Location Questions

Quotas are calculated based on responses entered into a survey. For our survey, we therefore need to ask about a person's gender, age and location.

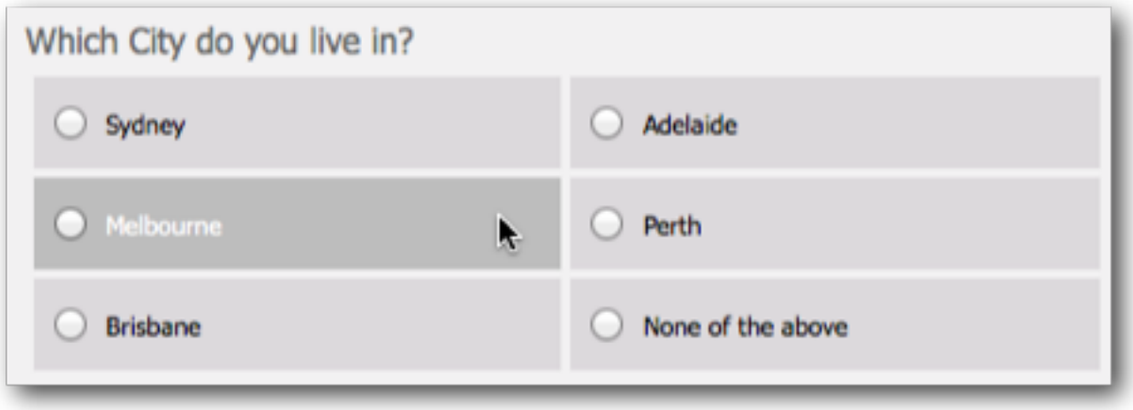
For a detailed explanation about adding questions, see our basic survey guide. For the purposes of this example, we will just look at the completed questions.

On our first page, we will ask for the respondent's gender:

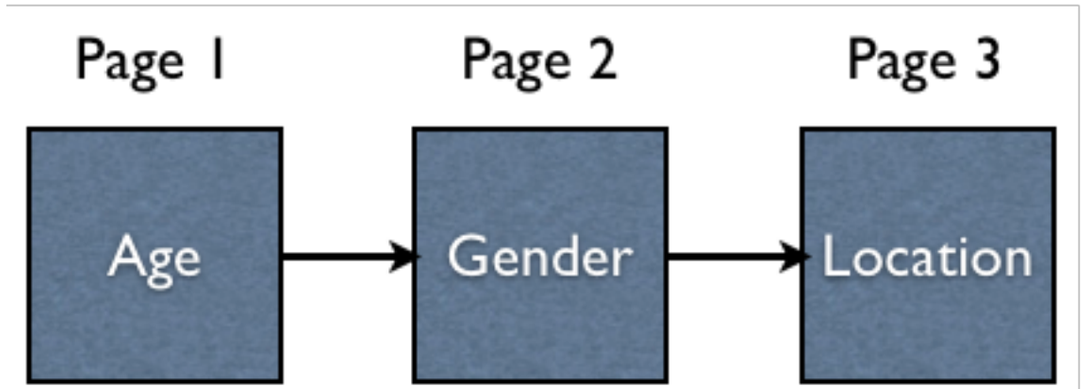
On the second page, we will ask for the respondent's age:



On the third page, we will ask where the respondent lives:



We now have a survey with the following structure:

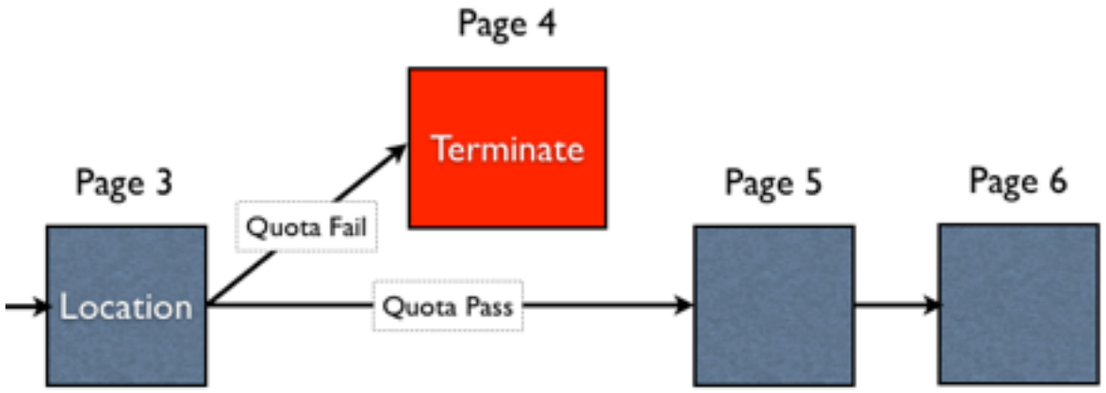


Once a respondent has answered these three questions, we will know whether or not they fit a quota.

> *If a respondent doesn't fit into a quota, we need to end the survey for them.*

The fourth page in the survey will therefore be a *Quota Fail Terminate Page*. If a respondent does not fit into quotas, they will see this page. If they do fit into a quotas, this page will be hidden and they will move onto page 5.

This logic is explained in the following diagram:

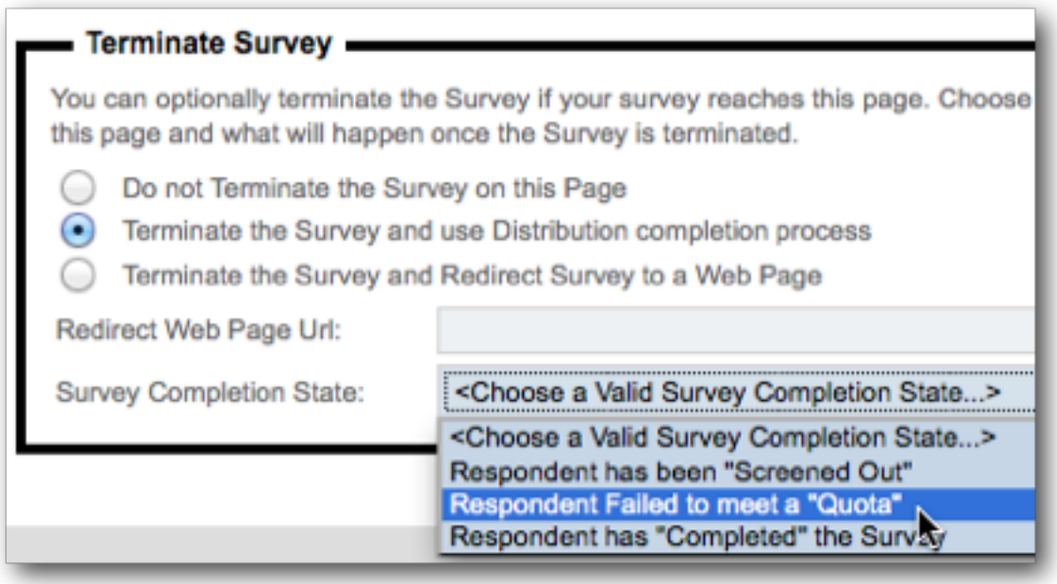


Quota Fail Terminate Page

We need to create the Quota Fail Terminate Page. This is simply a matter of creating a page in the survey, then:

1. Clicking the *Edit button* to edit the page details

2. Changing the type of page to indicate that it is a terminate page.



A terminate page is different to a normal page because:

1. It has a submit button, not a next button on it. This means it will be the last page seen by the respondent in the survey.
2. It has the ability to redirect to a specific location - which is very important when integrating with external respondent panels (a point we will discuss more fully in a future chapter).

Creating Our Quotas

So far we have the questions that are used in our quota, and we have a terminate page. What we are lacking is a calculation of the quotas, and a survey flow to properly handle the showing or hiding of the terminate page. Let's look at these issues in turn.

Using the Quota Builder

We have to set up quite a few quotas in this survey. In fact, we have to set up:

2 Genders x 7 Age Ranges = 14 Gender/Age Quotas
and
5 Location Quotas

That's a total of 19 Quotas.

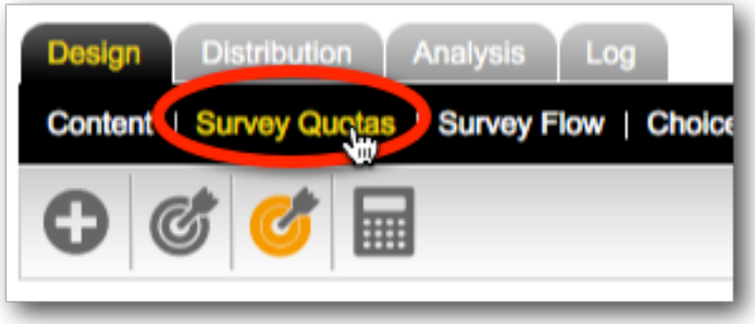
While setting each quota up manually can be done, fortunately Web Survey Creator has a **Quota Builder** that

does most of the work for us. It let's us create related quotas in a single process.

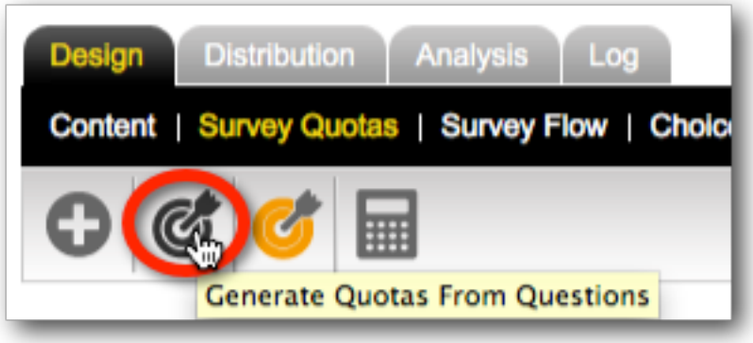
Creating The Gender/Age Quotas

Let's create the gender and age quotas first. The steps for creating these quotas are as follows:

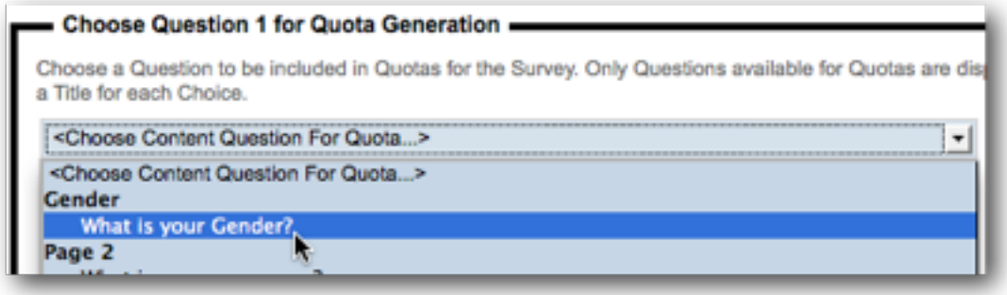
1. Click on the *Survey Quotas menu* under the *Design tab*



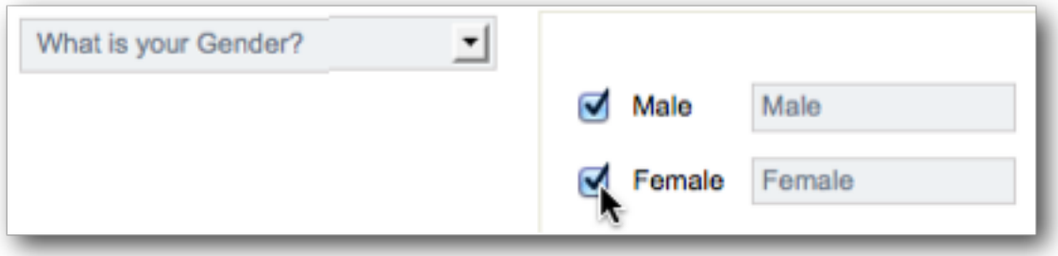
2. Choose the *Quota Builder* from the toolbar



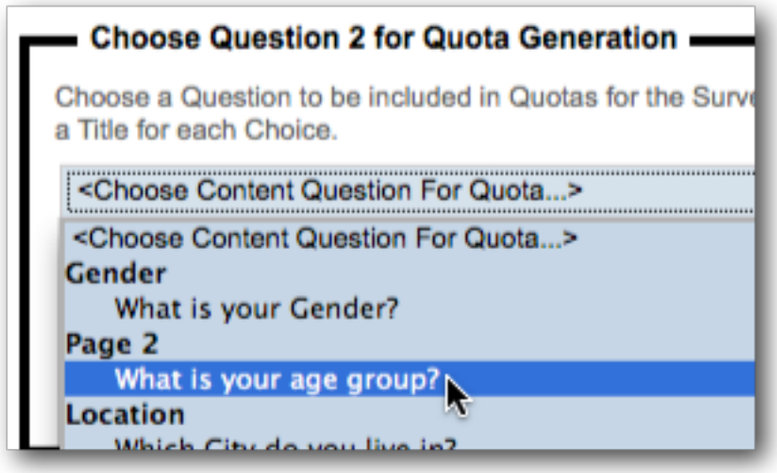
3. Choose *Gender* as the first question we want to use in our quotas



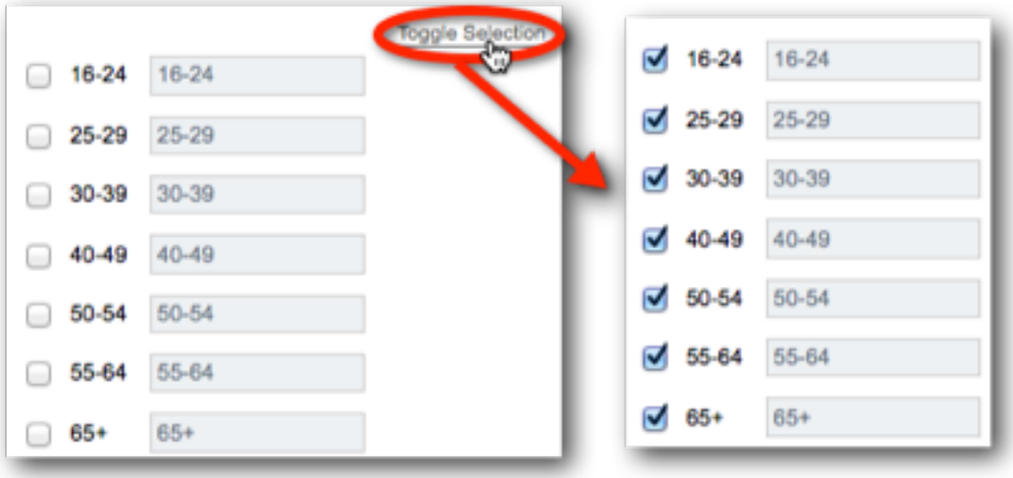
4. Choose both of the values for this question to indicate that they will both be used in the quota (note that we can optionally change how they will be described in the quota)



5. Choose *Age Group* as the second question we want to use in our quotas



6. Again, choose all values. We can do this quickly by pressing the *Toggle Selection* link



7. Press the *Set Quota Limits* button

8. All combinations of the choices selected will be presented ready for quota numbers to be entered

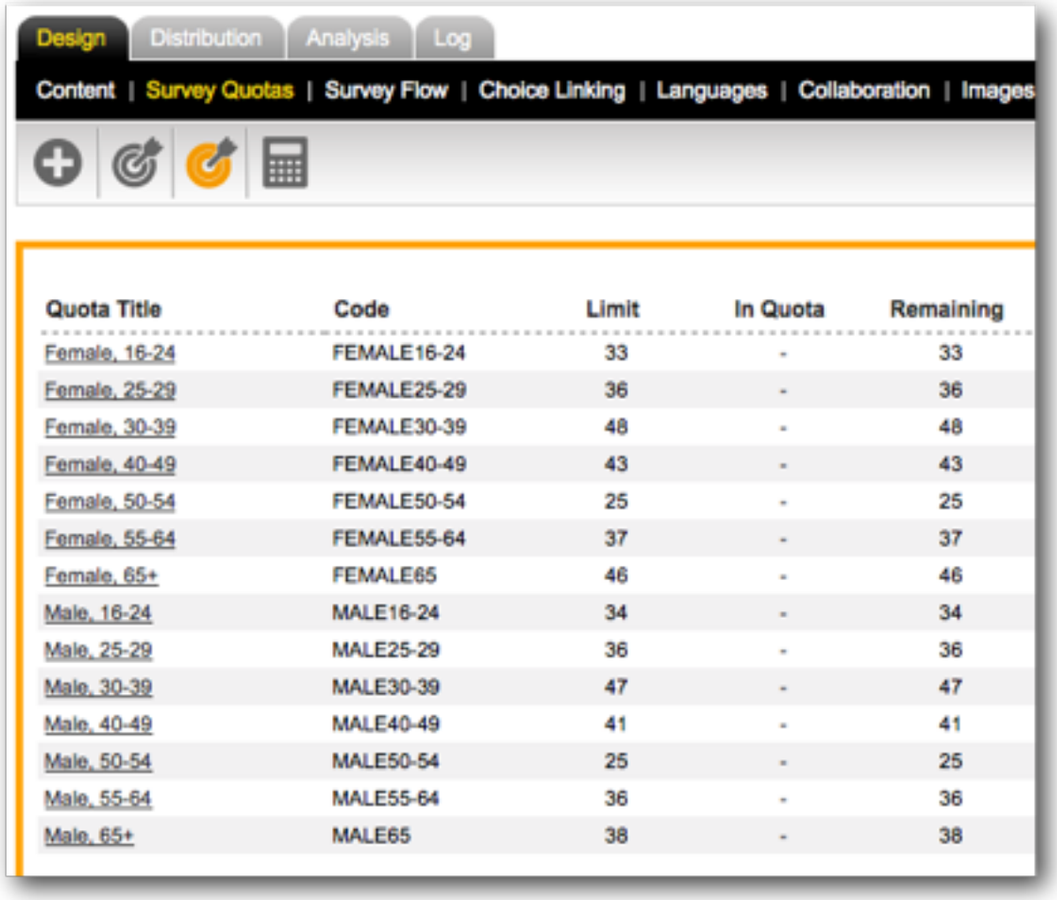
#	Title	Quota Access Code	Limit
1	Male, 16-24	MALE16-24	0
2	Male, 25-29	MALE25-29	0
3	Male, 30-39	MALE30-39	0
4	Male, 40-49	MALE40-49	0
5	Male, 50-54	MALE50-54	0
6	Male, 55-64	MALE55-64	0
7	Male, 65+	MALE65	0
8	Female, 16-24	FEMALE16-24	0
9	Female, 25-29	FEMALE25-29	0
10	Female, 30-39	FEMALE30-39	0
11	Female, 40-49	FEMALE40-49	0
12	Female, 50-54	FEMALE50-54	0
13	Female, 55-64	FEMALE55-64	0
14	Female, 65+	FEMALE65	0

9. Enter the quota numbers as per the original request from Worldburger

#	Title	Quota Access Code	Limit
1	Male, 16-24	MALE16-24	34
2	Male, 25-29	MALE25-29	36
3	Male, 30-39	MALE30-39	47

10. Press the *Save Quotas* button

11. The Quotas will all be shown under the *Survey Quotas* menu

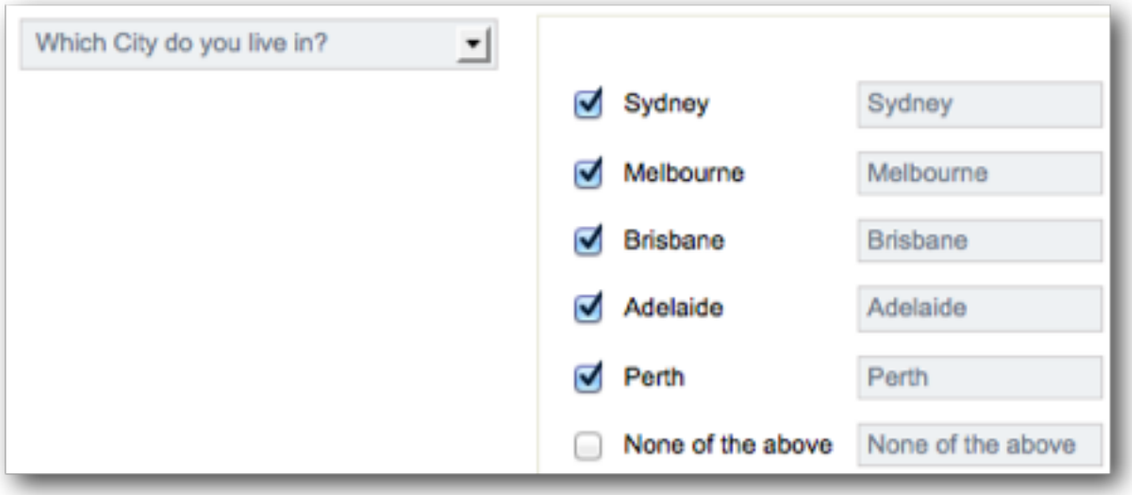


Quota Title	Code	Limit	In Quota	Remaining
Female, 16-24	FEMALE16-24	33	-	33
Female, 25-29	FEMALE25-29	36	-	36
Female, 30-39	FEMALE30-39	48	-	48
Female, 40-49	FEMALE40-49	43	-	43
Female, 50-54	FEMALE50-54	25	-	25
Female, 55-64	FEMALE55-64	37	-	37
Female, 65+	FEMALE65	46	-	46
Male, 16-24	MALE16-24	34	-	34
Male, 25-29	MALE25-29	36	-	36
Male, 30-39	MALE30-39	47	-	47
Male, 40-49	MALE40-49	41	-	41
Male, 50-54	MALE50-54	25	-	25
Male, 55-64	MALE55-64	36	-	36
Male, 65+	MALE65	38	-	38

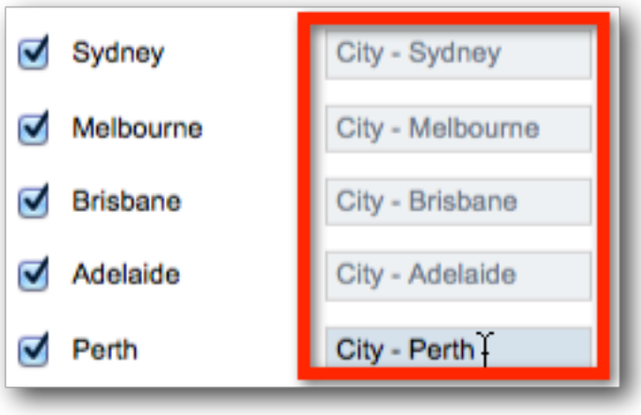
Creating The Location Quotas

The location quotas can be created in the same way as the gender/age quotas - in fact, they are even easier to create, since all the quotas are only based on a single question - *City lived in*.

We need to choose all values except “None of the above”, which would be inappropriate as a quota.



Quotas are shown alphabetically in the Quotas Browse - but we want the cities to appear together. We can do this by changing the descriptors for the cities before saving.



Once the location quotas are added, they will also be shown at the top of the Survey Quotas Browse (because they are first in the alphabetical order).

Quota Title	Code	Limit	In Quota	Remaining
City - Adelaide	ADELAIDE	51	-	51
City - Brisbane	BRISBANE	103	-	103
City - Melbourne	MELBOURNE	152	-	152
City - Perth	PERTH	62	-	62
City - Sydney	SYDNEY	157	-	157
Female, 16-24	FEMALE16-24	33	-	33
Female, 25-29	FEMALE25-29	36	-	36

Adding Quota Fail Terminate Page logic

We have already created our quota terminate - there’s just one problem!

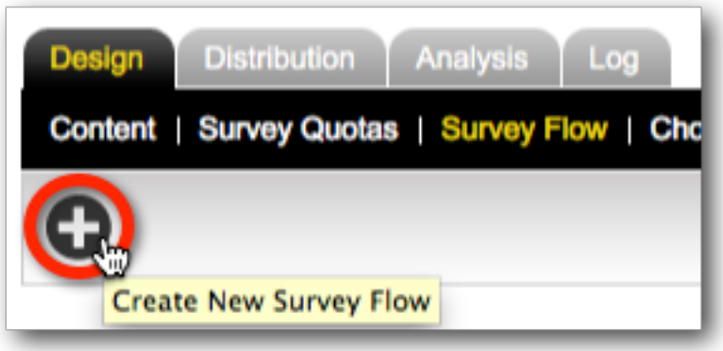
If survey flow logic is not added to hide a terminate page, the page will always be shown - and every respondent will be terminated!

Now that we have our quotas set up, we can correctly hide the terminate page (since the hide rule will be directly related to whether a respondent can fit in the quotas).

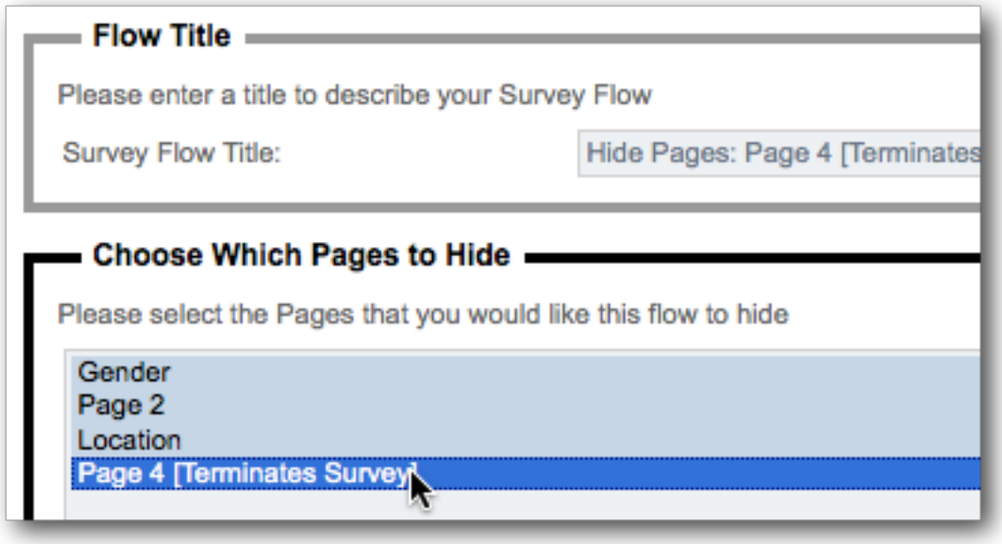
The steps for creating our terminate page logic are as follows:

1. Click on *Survey Flow* under the *Design tab*.

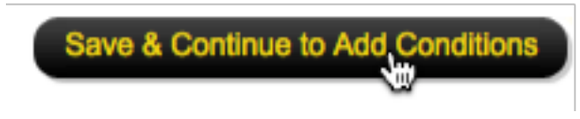
2. Add a *New Flow*



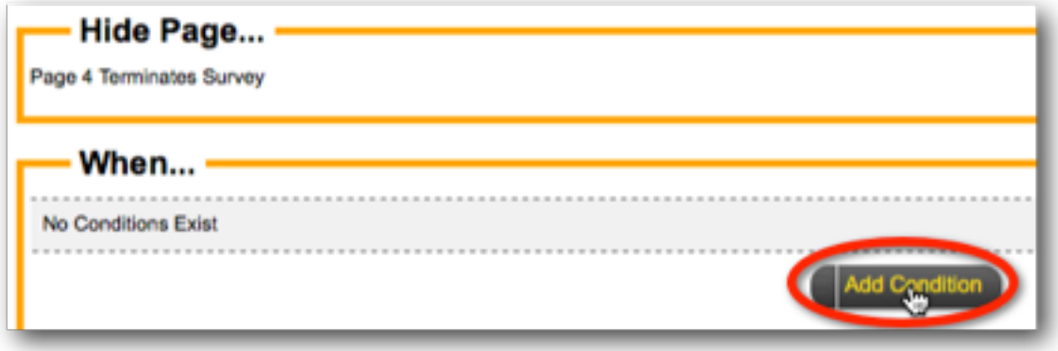
3. Choose to hide the terminate page



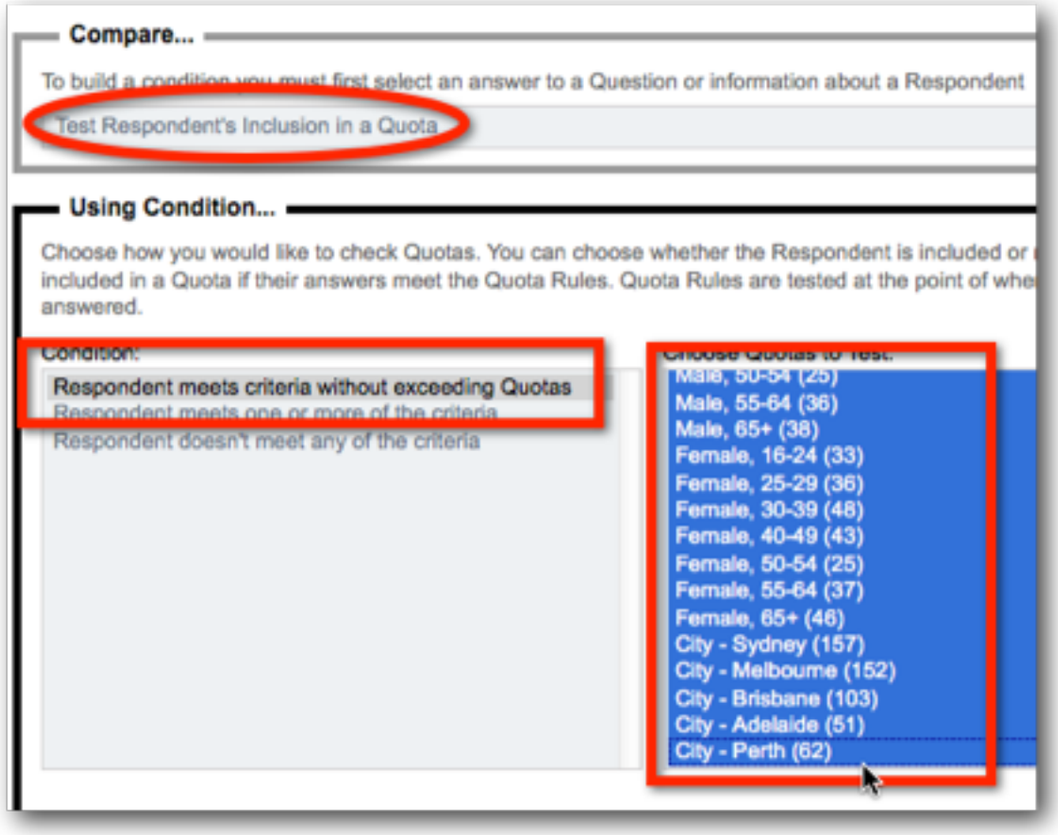
4. Click the *Save & Continue to Add Conditions* button



5. Click *Add Condition*



6. Choose to test all the quotas, as shown below



7. Click *Save Condition*

The terminate page will now only be shown under the following conditions:

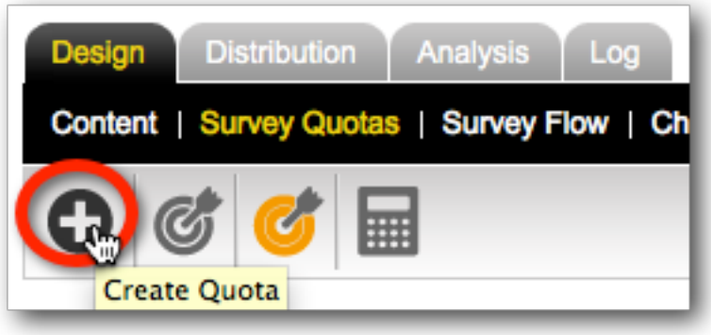
1. The respondent matches at least one of the quotas
2. All of the quotas attached to the respondent still have room (the only exception to this rule is if *Priority Quotas* exist, as discussed in the previous section).

Manual Adjustment of Quotas

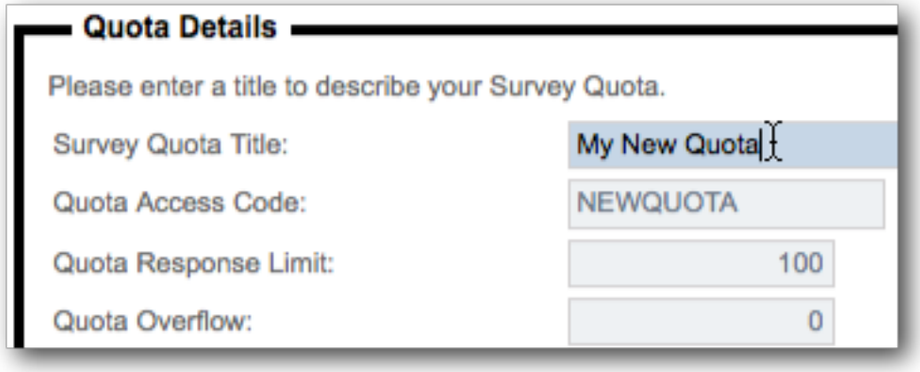
If you need to add or edit single quotas, you can do this through the Quota Management System.

Adding A Single Quota

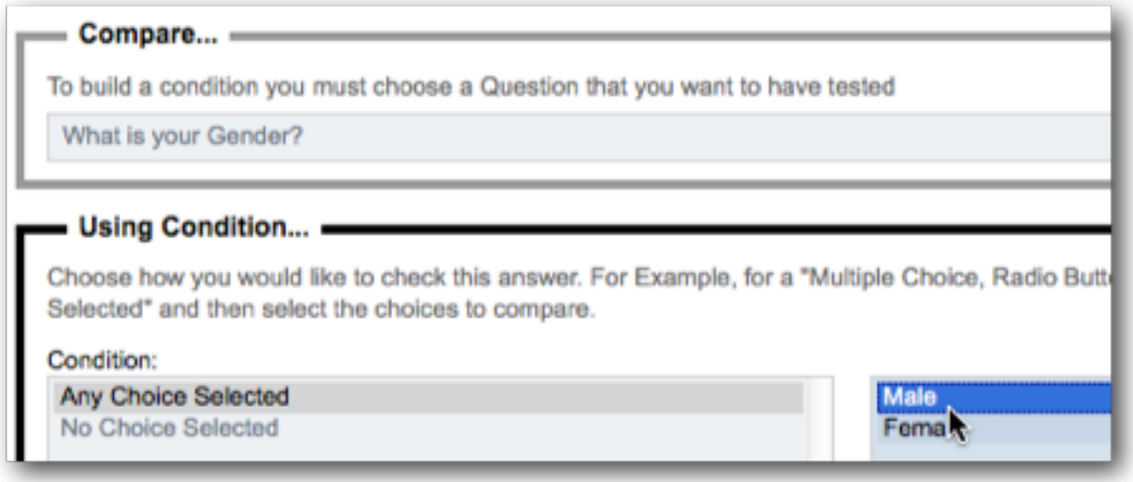
To Add a single Quota, click on the *Create Quota* toolbar button.



All details for the quota need to be entered, including a title for the quota.



The Quota Rules can be added one at a time, in a similar way to flow control.

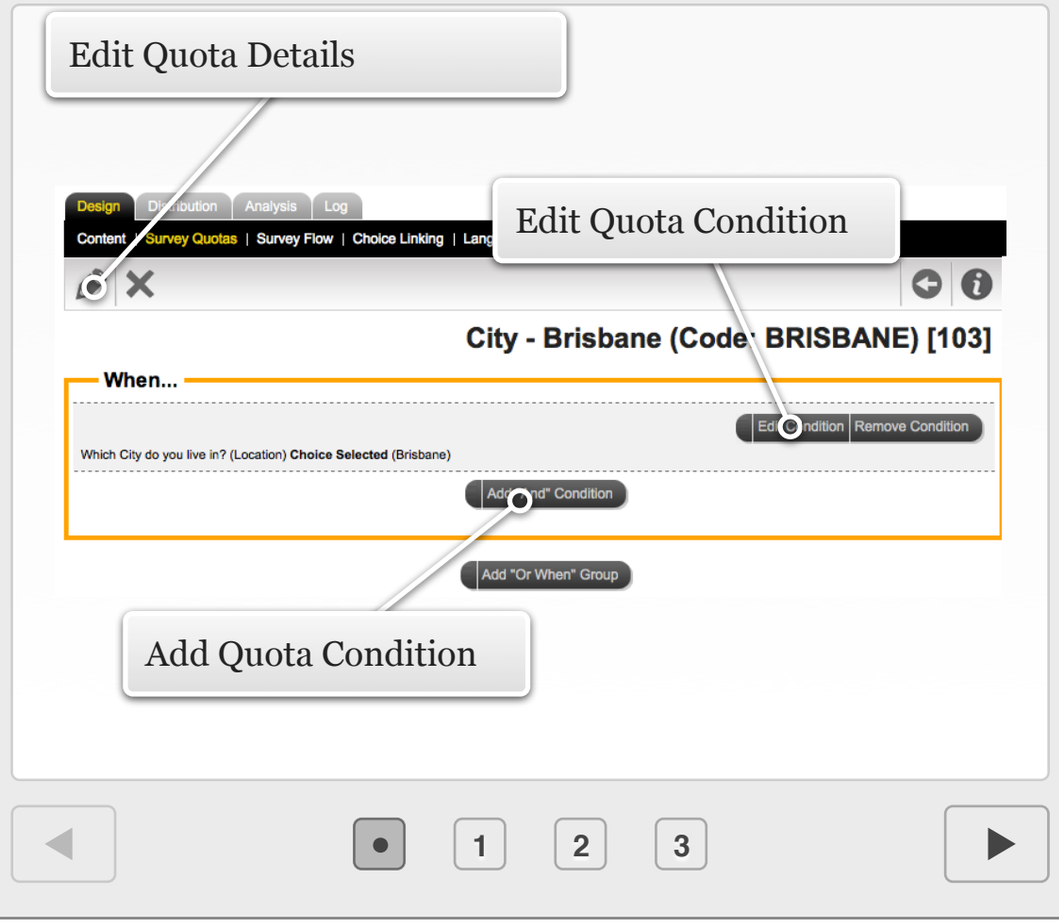


Editing A Quota

Editing a Quota is simply a matter of clicking on the Quota Title in the Quota Browse.

Quota Title	Code	Limit
City - Adelaide	ADELAIDE	51
City - Brisbane	BRISBANE	103
City - Melbourne	MELBOURNE	152

INTERACTIVE 2.1 Editing a Quota

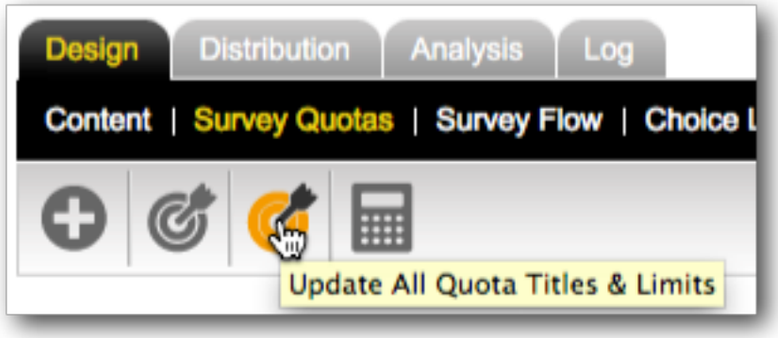


Editing Multiple Quotas At Once

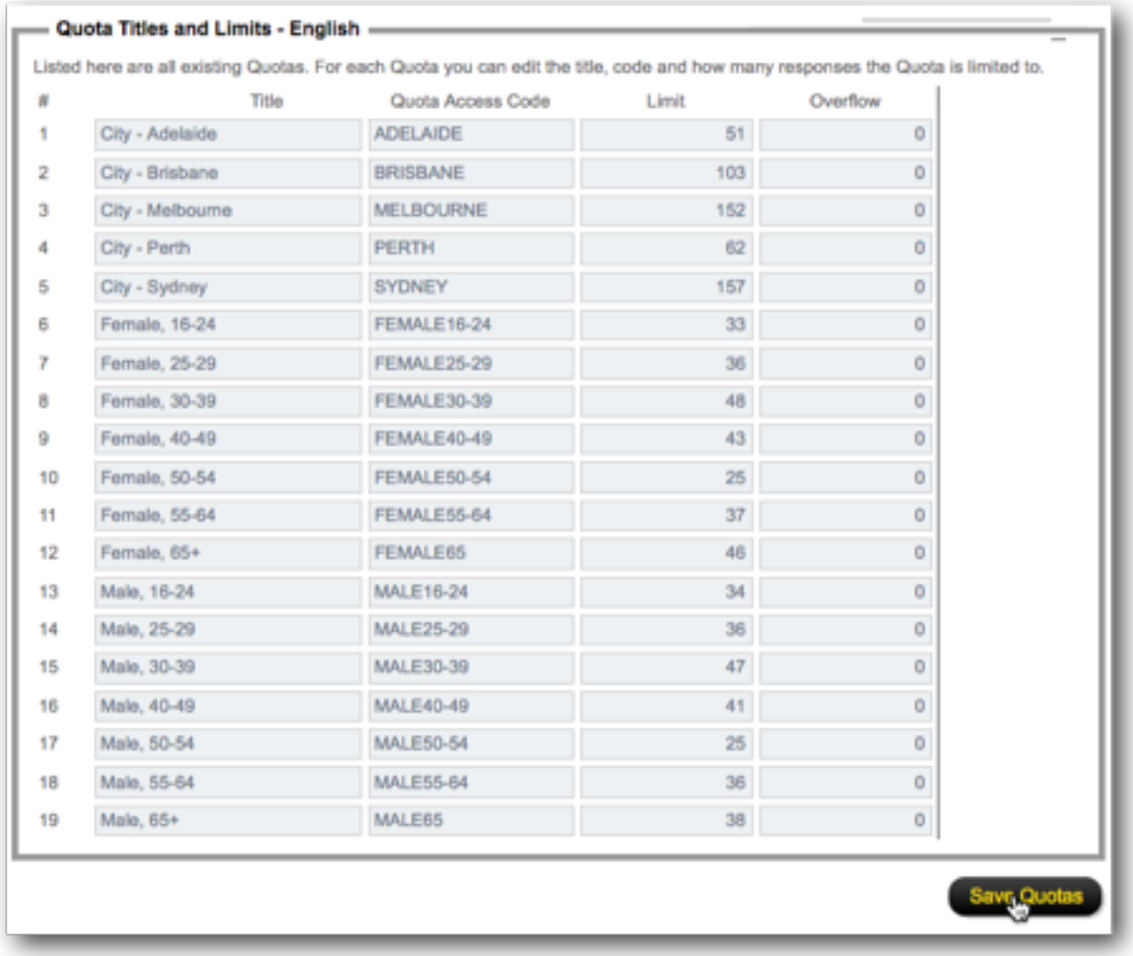
If you need to modify quotas, quite often it's going to be one the following details that needs to be changed:

1. How many people you are looking for in the quota
2. How much of an overflow you will allow
3. The access code or description for the quota

Web Survey Creator makes it easy to modify these details *en masse* by clicking on the *Update All* toolbar button under *Survey Quotas*.



Titles, access codes, limits and overflow amounts can all be adjusted from a single screen.



#	Title	Quota Access Code	Limit	Overflow
1	City - Adelaide	ADELAIDE	51	0
2	City - Brisbane	BRISBANE	103	0
3	City - Melbourne	MELBOURNE	152	0
4	City - Perth	PERTH	62	0
5	City - Sydney	SYDNEY	157	0
6	Female, 16-24	FEMALE16-24	33	0
7	Female, 25-29	FEMALE25-29	36	0
8	Female, 30-39	FEMALE30-39	48	0
9	Female, 40-49	FEMALE40-49	43	0
10	Female, 50-54	FEMALE50-54	25	0
11	Female, 55-64	FEMALE55-64	37	0
12	Female, 65+	FEMALE65	46	0
13	Male, 16-24	MALE16-24	34	0
14	Male, 25-29	MALE25-29	36	0
15	Male, 30-39	MALE30-39	47	0
16	Male, 40-49	MALE40-49	41	0
17	Male, 50-54	MALE50-54	25	0
18	Male, 55-64	MALE55-64	36	0
19	Male, 65+	MALE65	38	0

Tracking Quotas

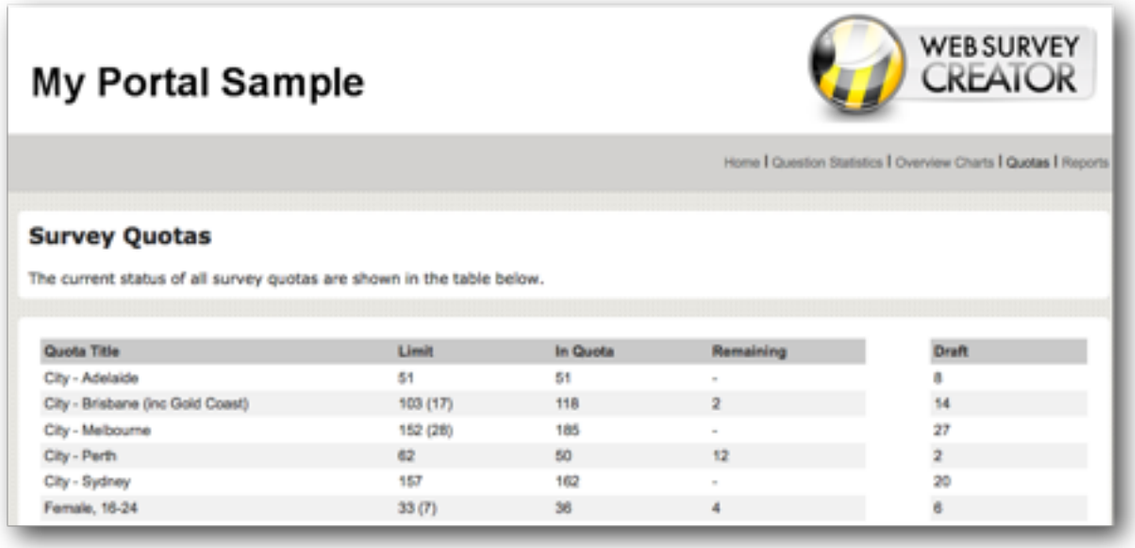
Quotas can be easily tracked under *Survey Quotas*, as we have already seen. This works well if you have set up the survey, and you are a Web Survey Creator user, but what if you are an outsider - like a client?

Let's assume that the management at Worldburger want to keep an eye on the progress of the survey. Fortunately there is a way for them to do this in WSC - through a *Web Portal*.

A Web Portal can show a number of things including:

Current Status of Quotas

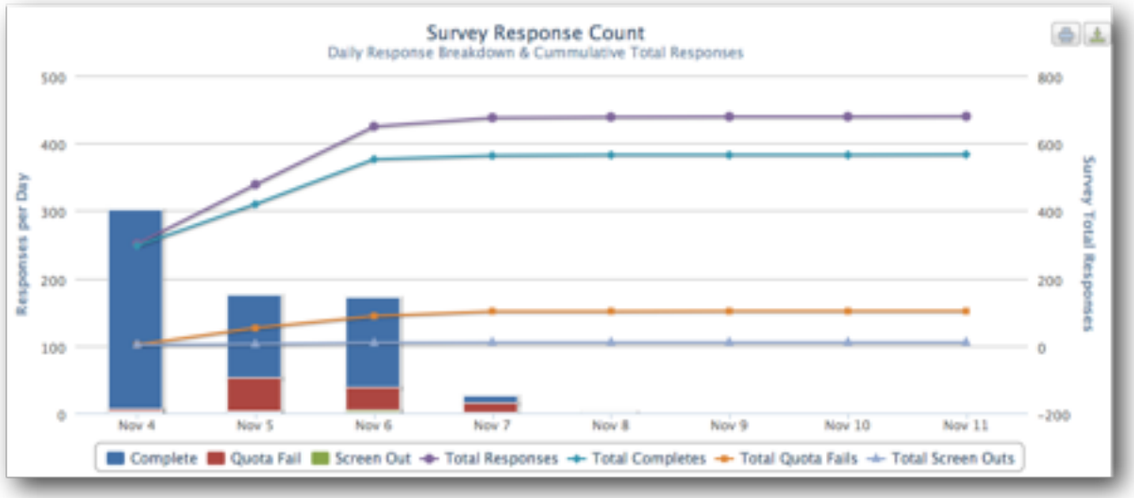
All quotas are listed, together with the number of respondents who are currently "In Quota" and how many are still needed.



Quota Title	Limit	In Quota	Remaining	Draft
City - Adelaide	51	51	-	8
City - Brisbane (inc Gold Coast)	103 (17)	118	2	14
City - Melbourne	152 (28)	185	-	27
City - Perth	62	50	12	2
City - Sydney	157	162	-	20
Female, 16-24	33 (7)	36	4	6

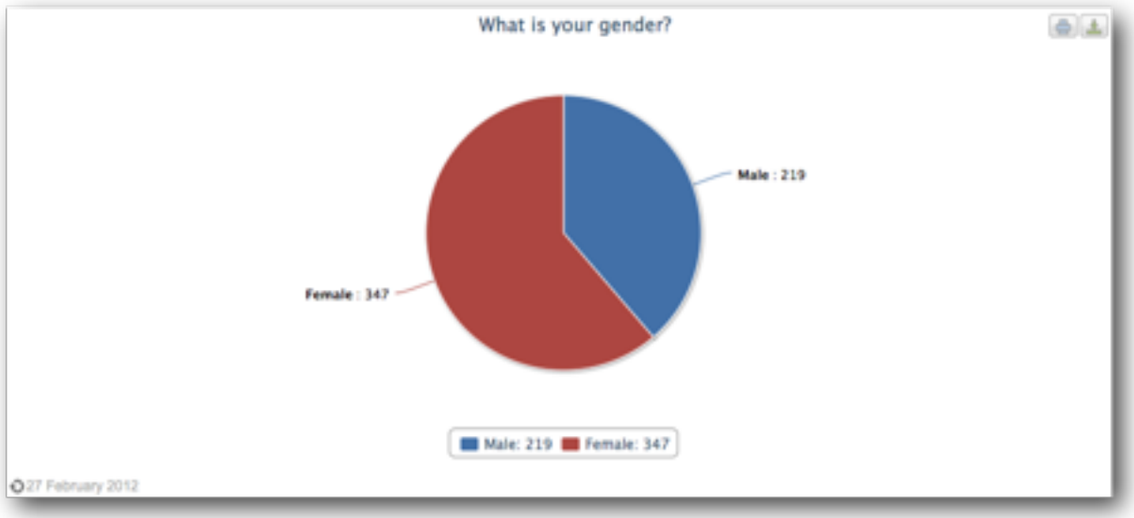
Response Counts

A daily count of responses are shown, together with a cumulative total - broken down by type (quota outs etc.)



Individual Question Statistics

Current counts for responses to each of the questions in the survey.

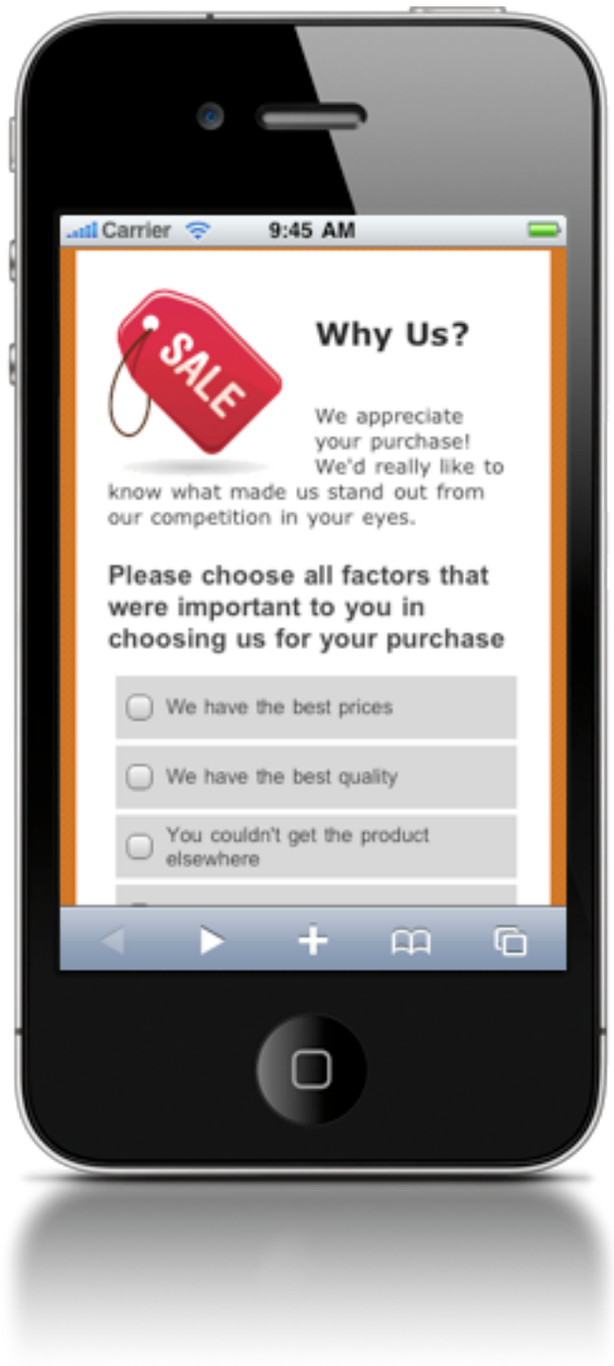


CHAPTER 3

Mobile Surveys

The world of Surveys is currently changing at a pace not seen since the original shift to the Web. Having a Web Survey is no longer enough - your surveys now need to be accessible on mobile devices.

“Smart phones” are your window into higher response rates, and more satisfied respondents.



SECTION 1

The Move to Mobile

IN THIS SECTION

Mobile vs Desktop

The expectations that respondents now have when it comes to Web Surveys are quite easy to quantify. Unfortunately they are not so easy to meet, since many of the expectations are not easy to achieve.

EXPECTATION	PROBLEM
Survey completion on a mobile device or a desktop computer - not just one	Many survey solutions have full-featured "desktop" and cut-down "mobile" versions. Choose one or other when setting up (not both).
"Sexy" surveys	"Sexy" usually equals flash-based desktop questions. Not mobile compatible.
If using a mobile device, survey is still full-featured	Usually formatting and functionality on mobile devices is cut down

The existence of mobile devices with true Web browser capabilities is a relatively new phenomenon (some suggest it began with the iPhone in 2007) and many of the gaps between expectations and the reality of Web Surveys on mobile devices is simply a result of large, established software packages still trying to "catch up" with the expectation.

Going "half-way" with cut-down mobile-specific survey tools misses the point - people have "cool" mobile devices and expect the things they do on those devices to be "cool" as well.

There is an added benefit to getting to people via their mobile phone or tablet - the sorts of people who use these devices are often the younger demographic that are so hard to get any response out of.

The Humble P.C. Is Far From Dead...

Regardless of how much people like using their mobile devices, there is still an important place for desktop (and laptop) computers. It is therefore critical that a Web Survey continues to work well in a standard PC environment. There are a number of benefits to this environment that will never be eradicated by the onslaught of mobile devices:

- Large amounts of text are easier to enter on a full-sized keyboard
- Virtually every household has a PC (and can share it, unlike personal mobile devices) for completion of surveys
- A significant percentage of respondents do not have or want smart mobile devices
- Older generations often feel more comfortable using a PC, since that is what they are used to

Mobile doesn't have to mean basic

With use of the right technologies, there is only really one limitation that you have to work around on a mobile device...

Mobile devices have less screen real estate than desktops. The interface provided must take this into account.

Layout Management

Images need to be resized, and wide layouts changed to fit a mobile screen.

Advanced text question types - like demographics - look best using a form-style layout on a desktop. Again, on a mobile device, the layout needs to be modified.

Text Entry

On a desktop, text entry is easy - the keyboard in the primary input device. On a mobile, text fields not only need to be formatted for the screen - they also need to work with the input system on the device.

Question "Morphing"

Some question types simply can not be used on a mobile because the screen size simply doesn't allow for a useable experience. An example is a matrix question, which requires

the full width of the screen to be able to be entered on a desktop.

Keeping the "Sexy" in your Surveys

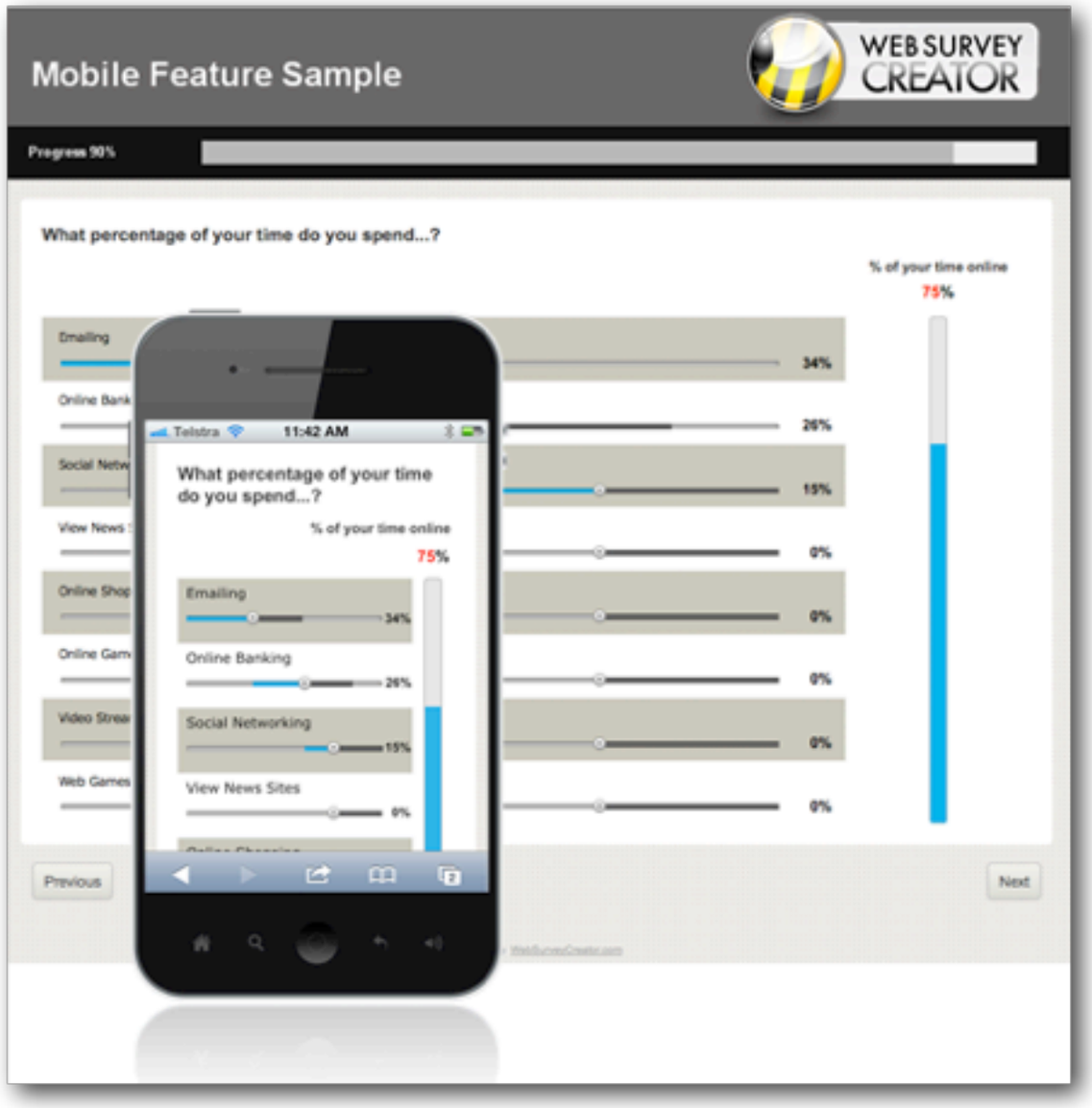
One of the biggest issues faced when supporting mobile devices is to avoid losing all the "sexy" capabilities that respondents have become used to.

Gotta Have Video!

Many market research surveys use video as a way to show product material such as advertisements.

Touch-enabled Sliders

Sliders are used more and more in surveys. It's important to have the same capability on mobile devices - including advanced implementations like the component breakdown slider question below.



Drag 'n' Drop

Mobile users love their touch-screen smartphones. Drag 'n' drop needs to be available on the phone to mimic dragging operations on the desktop version of survey questions.

Going Mobile: Working smarter, not harder

If you want to create Web Surveys that support both mobile and desktop respondents, you need to adhere to some rules when looking for a Web Survey design tool:

1. Never use flash-based questions. They immediately exclude mobile users

2. Avoid solutions that split mobile and desktop survey design into separate modules - you'll be doing twice the work, and possibly have incompatibilities due to limitations in the mobile version

3. Avoid survey solutions for desktop respondents that have mobile capabilities "tacked on". Mobile is too important to be an afterthought.

4. Avoid overly basic mobile survey capabilities - you'll be able to get to mobile users, but not with the kind of survey you need to create.

SECTION 2

Mobile Survey Example

IN THIS SECTION

Our Example Survey

The example we will be reviewing in this section has been created in Web Survey Creator. Images of mobile survey capabilities in the previous section came from this example.

To see the full example, visit the WSC sample online:

http://www.websurveycreator.com/c/survey-sample-3.aspx

Making a WSC survey mobile-capable

A survey can tell what browser - and therefore what type of device - it is being run on. Web Survey Creator uses this information to determine how a survey will be displayed.

What do you need to do to make a WSC survey work on a mobile device?

NOTHING

Any survey created in Web Survey Creator will work on a mobile device - they whole basis of the system is you design a survey once and it can be used everywhere.

Targeting Specific Platforms

While Web Survey Creator will manage surveys on a mobile device automatically, there may be times when you actively want to do something different for respondents who answer on a mobile as opposed to people who answer on a PC (or on a tablet for that matter).

Whenever Web Survey Creator shows a survey to a respondent, it determines which of the following types of device is being used to view the survey:

PC	Computer running a full PC operating system such as Microsoft Windows or Mac OSX. These are the types of machines that have traditionally been used to complete surveys.
Tablet	Mobile devices that have moderately sized (7”+) touch-screens, and are generally referred to as “tablets” (e.g. iPad, Samsung Galaxy Tab)
Mobile	Mobile devices with relatively small screens (<5”) that are generally referred to as “Smartphones”.

Web Survey Creator can determine the type of device from the browser that is running on the device. Knowing which device is being used can open opportunities to change the behavior of the survey for different devices.

Device Targeting through Flow Control

Our Mobile Survey Sample doesn't have a single introduction page - it has three. One for each type of device.

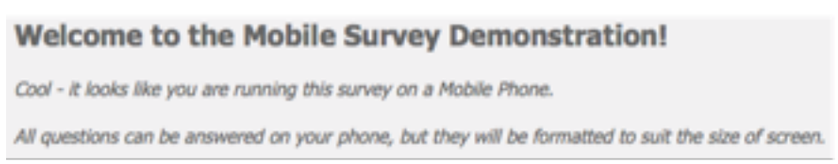
While multiple introduction pages is certainly not mandatory - since WSC will render a single page correctly on all three styles of device - it does give some flexibility in survey design. It allows the designer of the survey:

1. To show different introduction text based upon the type of device being used.

1.1. PC

1.2. Tablet

1.3. Mobile

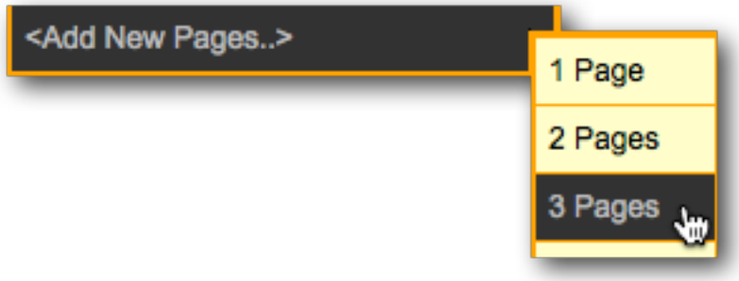


2. You can proactively limit content based upon the device. In our example survey, the introduction for mobile phones is shorter and does not contain any images - taking into account the fact that the survey is likely to be done on a 3G network.

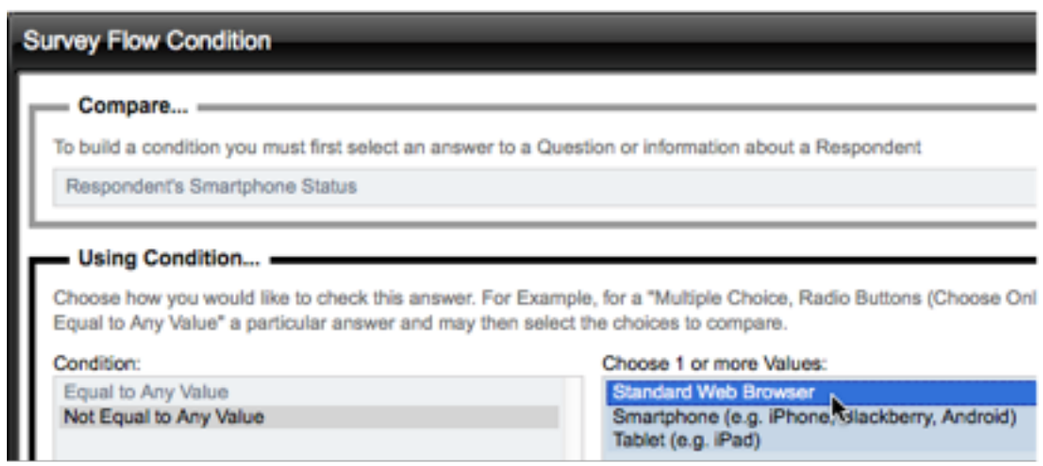
Setting Up Flow Control

To set up the introduction pages in the sample, we did the following:

1. We created three pages - one for each of the devices we wanted to target



2. We then created three survey flow rules - one rule each for the introduction pages. The rules tested the type of device that way being used to view the survey



Chapter 4

Product Testing

Product testing comes in many forms. In this chapter we will look at one example of product testing where we consider a series of car brands, and then ask specific questions about them.

This example brings together multiple advanced concepts, including page looping, data piping, flow control and choice linking.

SECTION 1

Product Testing Overview

IN THIS SECTION

The Problem with Product Testing

Product testing can be a powerful way of finding out information about a large number of related products in a single survey.

One of the biggest issues with setting up such a survey *is* the large number of products. Let's consider an example:

I want to product test 20 products. For each product I want 4 pages with 2 questions per page. So, what do I need to set up?

Pages: 20 x 4 = 80 pages
Questions: 20 x 4 x 2 = 160 questions

We can see from the example above that it doesn't take long before the setup becomes unwieldy - and it only gets worse if there is complex logic to be included as well.

Using Surveys to do Product Tests

When you create a survey for product tests, there are a number of standard processes you need to go through to set the survey up. These are briefly discussed in this sections.

Note that a more detailed explanation of these processes - using the Web Survey Creator product - is the topic of the next section.

Base list of Products

A base list of the products we want to test will need to exist in our survey.

For our example, we will be using car brands as our "products". This "list" will be a question with each of the possible car brands as answers.

It is common for our base list to include a lot more information than simply a piece of text. For example, images can be used for brands:

Questions for Each Product

Once you have determined that someone knows a brand, you can start asking additional questions relating to that brand. In our case, we are trying to get an overall opinion of a brand, together with specific views on products within the brand's range of cars.

Our questions will include:

What is your overall opinion of the brand?

Would you recommend this brand to your friends?

Have you ever owned a car from this brand?

Which of the following models have you owned?

How would you rate the car(s) you have owned from this brand?

We would want to ask these questions for each of the brands, and we would like to customize them by brand as well. How can all this be done? Through the use of advanced survey functionality!

Advanced Survey Functionality

All survey tools will allow you to ask a simple set of questions. Product testing, however, requires something much more powerful. We need:

1. To be able to ask the same questions for each of the products
2. To be able to refer to product-specific details - such as the product name - when we ask the questions
3. To deal with advanced capabilities - like flow control - on a product-by-product basis

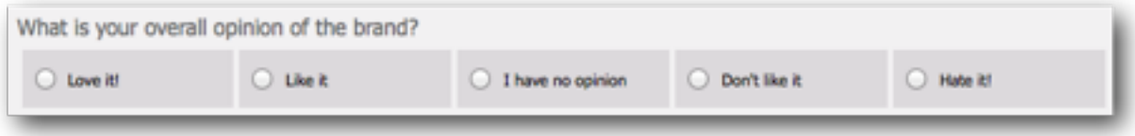
All of these things require advanced functionality as described below.

Page Looping

Let's consider the first question we want to ask:

What is your overall opinion of the brand?

We clearly need to ask this question for each of our brands. This question would be on it's own page.



This page would need to be "Looped" for each brand. "Looping" has two effects:

1. There will be a page for each brand asking this question
2. If someone does not chose a brand, the page relating to that brand would be automatically hidden

Flow Control

Flow Control refers to the flow of the survey. More specifically, it refers to the hiding of pages (and therefore questions) that are not needed for a particular survey.

The flow of pages that are looped for a product test will automatically be hidden when page loops are set up - but what about pages *within* the loop?

Let's consider an example. We have three questions in our list of questions as follows:

Have you ever owned a car from this brand?

Which of the following models have you owned?

How would you rate the car(s) you have owned from this brand?

If a respondent has never owned a car from a particular brand, there is no point rating the cars owned (since there won't be any!)

Flow control can be used to ensure invalid pages are not shown to a respondent.

Data Piping

All the questions we have are very generic. Data piping allows you to pipe answers from earlier in the survey into later survey content. For example, rather than:

What is your overall opinion of the brand?

we could have:

What is your overall opinion of BMW?

This would of course only appear in the BMW loop - the appropriate brand would be used in each of the loops.

A more advanced use of data piping is the listing of models to choose from. Without data piping, the model question would have to be a generic question with a text answer.

Enter a list of models you have you owned from BMW

There is little we can do with a text list, and statistical analysis would be average at best. We will see in the next section that with data piping, we can pipe actual car models into this question, so it would look more like:

Which of the following BWM models have you owned?

- 1 Series
- 3 Series
- 5 Series
- 7 Series

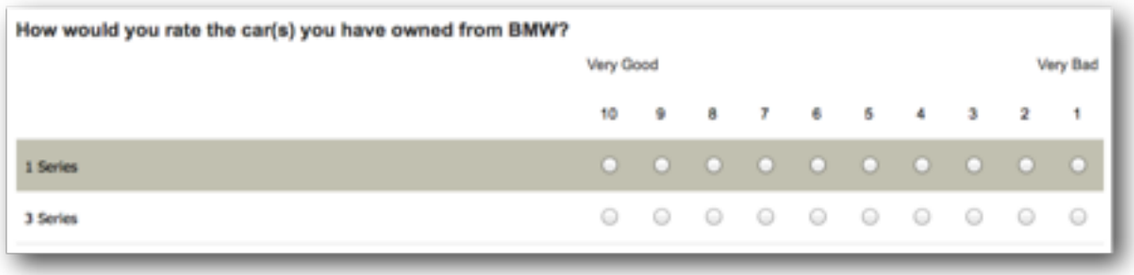
Choice Linking

Choice linking refers to showing a list of choices in a survey based upon a previous list.

In our example, the best use of choice linking would be to refer to the appropriate models for the question:

How would you rate the car(s) you have owned from this brand?

We could show this as a matrix, with each of the cars owned showing as a row - if I pick two models, for example, the matrix would have two rows.



SECTION 2

Product Testing Example

IN THIS SECTION

Our Example Explained

For the purposes of this section, we will consider an example that shows off all the key elements of a product test without getting too crazy (we'll test 4 brands - not 20!). Everything discussed can be used with real product tests of any size of course.

OUR PRODUCTS

In our example, our "products" will be Car Brands, and within those car brands we will be looking at four models. The car brands and models are listed below:

BRAND	MODELS
BMW	1 Series 3 Series 5 Series 7 Series
Ferrari	599 458 California F70
Honda	Civic Accord Odyssey NSX
Mercedes-Benz	A Class B Class C Class S Class

Product Questions

For each of the brands and models, we want to ask the following questions:

Page 1

What is your overall opinion of the brand?

Would you recommend this brand to your friends?

Have you ever owned a car from this brand?

Page 2

Which of the following models have you owned?

Page 3

How would you rate the car(s) you have owned from this brand?

Setting up the Product Test

We know what our products are, what models we have for each product, and what questions we are going to ask. The approach we will take will be as follows:

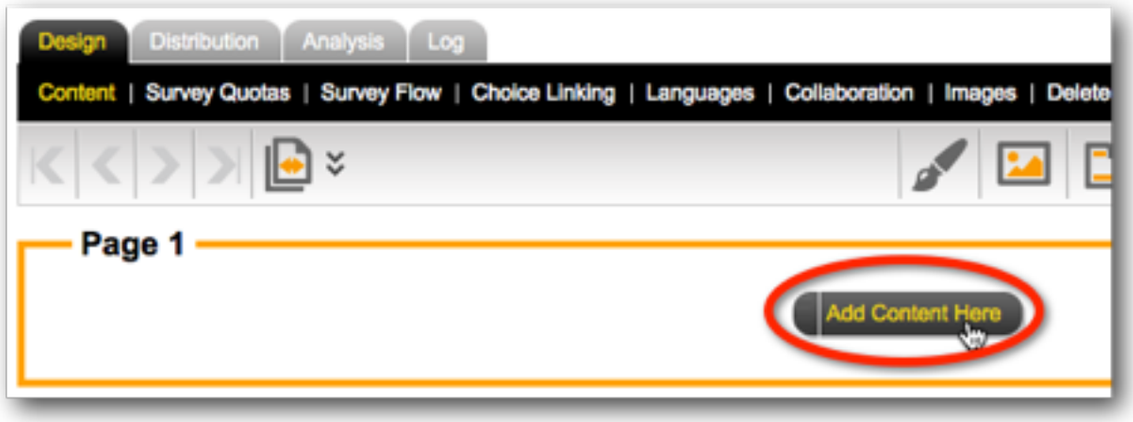
1. We will set up our list on products in our initial question
2. Three Pages (the *Looping Source* Pages) will be added
3. The *Product Questions* will be added to the pages
4. Flow control will be set up to ensure pages 2 and 3 are hidden for people that haven't owned a car of the particular brand
5. Choice linking will be used for the matrix on Page 3
6. We will use *Page Looping* to build all the pages needed to manage the Product Test
7. We will preview our product test!

Creating Our Example

We are going to build our example using Web Survey Creator. We will skip over many of the basics, as these are covered in our basic guide to Creating Web Surveys.

Our Initial Question

Setting up the initial question is very straightforward - we create a new survey, and then on the first page click *Add Content Here*.



We will add a *Choice Question* with multiple values - since more than one brand can be selected.

When entering our choices, we can take advantage of the quick entry of images by including the image details for each choice:

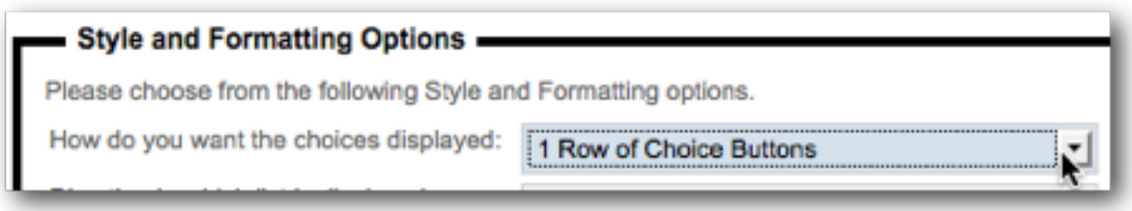
```
BMW||IMAGE:Car-BMW.png;;50
Ferrari||IMAGE:Car-Ferrari.png;;50
Honda||IMAGE:Car-Honda.png;;50
Mercedes-Benz||IMAGE:Car-Mercedes.png;;50
```

The format of an entry in the choice list is as follows:

Choice Text | *Choice Value* | IMAGE:*abc.png* ; *width* ; *height*

For our choices, we don't need either a value or an image width. By only entering the image height, we are telling the system to automatically make the width proportional to the height.

Just for looks, we are also going to show the choices on one row, and use 75% width.

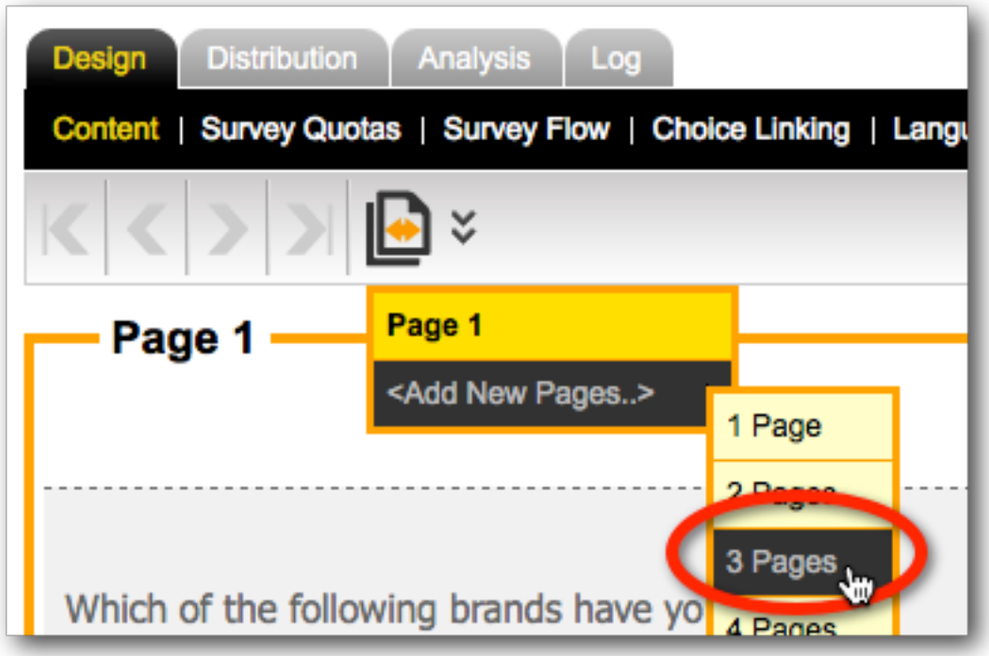


When we add the question, it looks as follows:

Looping Source Pages

A "Looping Source Page" is simply a page that, when we come to creating our page loops, will be used as one of the pages that will be duplicated for each loop.

At this point, these pages are added just like any other page. We can add all three pages in one go from the *Pages toolbar button*.



Product Questions

Our questions about the products need to be added to each of the pages in the order described earlier in this section.

Adding questions is a basic procedure covered in our previous manual, so what we want to look at here are the things that are different when creating questions in preparation for our loops.

Data Piping The Brand Name

Our first question is as follows:

What is your overall opinion of the brand?

As we have discussed, we want this to be more specific to the brand we are looking at - for example:

What is your overall opinion of BMW?

This is achieved through **Data Piping**. As with any data piping, we have to start by adding a *Question Access Code* to our product question. We will give this question a code of *CARBRAND*:

OK, so how do we use this piping? Under normal circumstances, our question would appear as follows:

What is your overall opinion of [@CARBRAND@]?

The question is, will this work? While your first instinct may be that this would be right, you need to consider how we are going to end up using this question - it will be copied 4 times - once for each of the page loops.

> *Data piping must be thought of differently when it is used on a page that will be copied for page looping. It must show something different for each loop.*

A Data Pipe must be used that directly refers to the choice that relates to the current loop. To achieve this you must use the following syntax:

What is your overall opinion of [@CARBRAND:N@]?

When this is copied by creating a page loop, the four pages that will be generated will have the following data pipes:

What is your overall opinion of [@CARBRAND:1@]?

What is your overall opinion of [@CARBRAND:2@]?

What is your overall opinion of [@CARBRAND:3@]?

What is your overall opinion of [@CARBRAND:4@]?

These data pipes will be converted to:

What is your overall opinion of BMW?

What is your overall opinion of Ferrari?

What is your overall opinion of Honda?

What is your overall opinion of Mercedes-Benz?

Data Piping The Models For A Brand

For each brand that is chosen, a list of models needs to be shown. We need to pipe these details into the possible answers for the question:

Which of the following models have you owned?

This raises two questions:

1. Where are the models going to be stored for each of the brands?
2. How are we going to pipe these models into the question about models owned?

Let's consider the issue of storage of the models first. The answer to storing the models for a brand, and anything else that may be applicable for the brand, is to use **Choice Tags**.

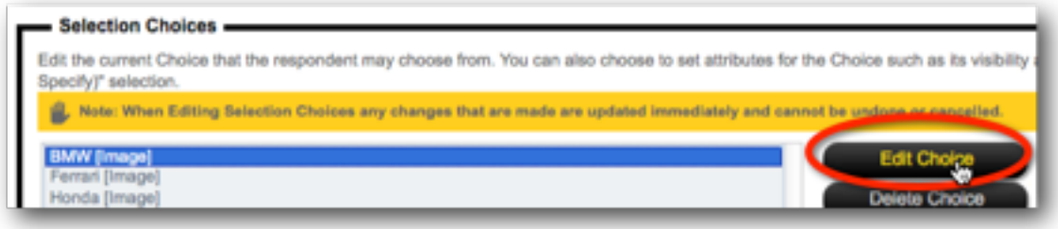
These tags can contain any text data you wish. Tags have a simple structure:

```
TAGNAME: Value
```

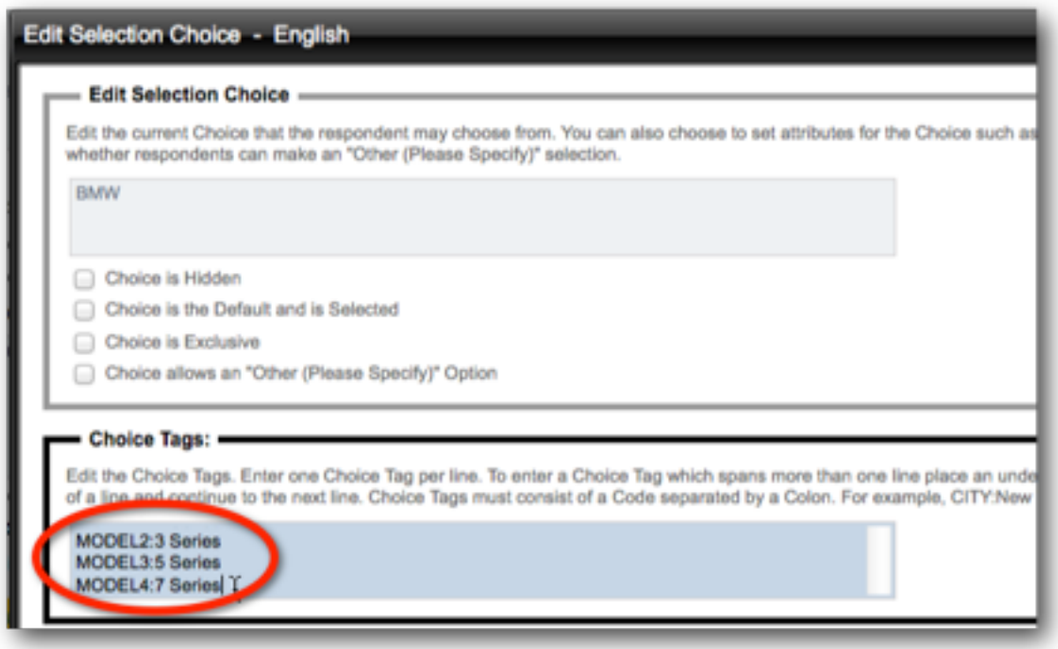
Setting up tags on choices can be done by:

1. Editing the question with our choices by pressing the *Edit button*.

2. Editing a choice for the question.



3. Enter new tags, one per line.



4. Saving the choice, and repeating for the other choices.

Note that tags can be added along with the other choice details when the choices are first added to - using the following format:

```
BMW||IMAGE:Car-BMW.png;;50|MODEL1:1 Series|MODEL2:3 Series|MODEL3:5 Series|MODEL4:7 Series
Ferrari||IMAGE:Car-Ferrari.png;;50|MODEL1:599|MODEL2:458|MODEL3:California|MODEL4:F70
Honda||IMAGE:Car-Honda.png;;50|MODEL1:Civic|MODEL2:Accord|MODEL3:Odyssey|MODEL4:NSX
Mercedes-Benz||IMAGE:Car-Mercedes.png;;50|MODEL1:A Class|MODEL2:B Class|MODEL3:C Class|MODEL4:S Class
```

Once we have added all our tags to our choices, they can be used through data piping.

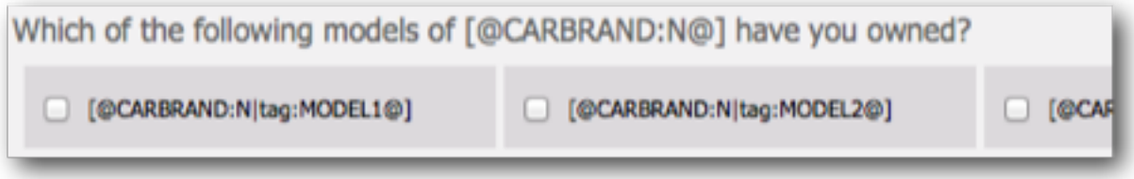
We want to pipe into the values for the question:

Which of the following models have you owned?

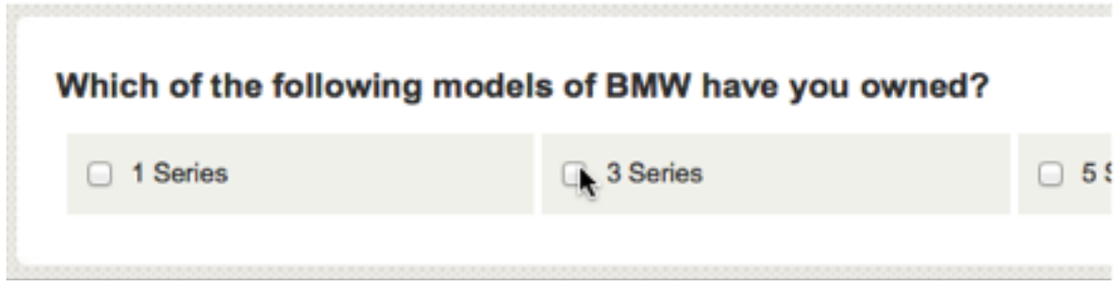
We can do this by setting up the choices in this question as data-piped values:

```
[@CARBRAND:N|tag:MODEL1@]
[@CARBRAND:N|tag:MODEL2@]
[@CARBRAND:N|tag:MODEL3@]
[@CARBRAND:N|tag:MODEL4@]
```

These codes tell the system which choice to use for the data pipe, and the tag to show from that choice. The question would look as follows in the designer:



If BMW was chosen by a respondent, the question would show in the survey as:



Flow Control

Flow control is used to hide pages based on answers in a survey. In our example, we have the following questions:

Page 1

...
Have you ever owned a car from this brand?

Page 2

Which of the following models have you owned?

Page 3

How would you rate the car(s) you have owned from this brand?

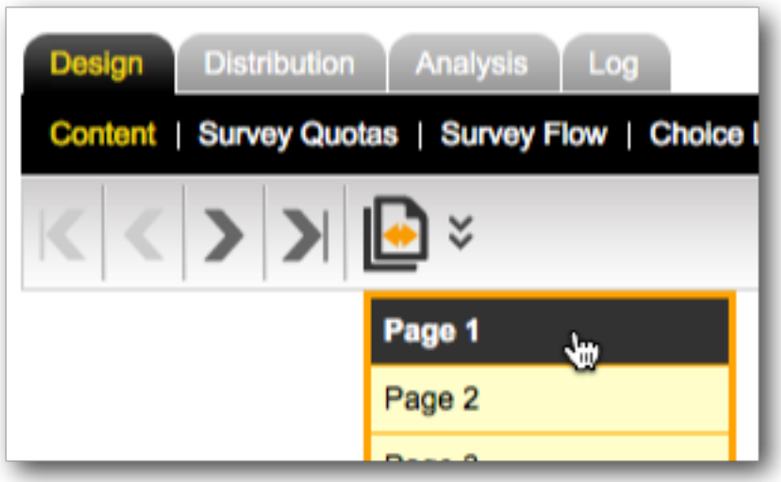
Clearly pages 2 and 3 need to be hidden if the respondent has never owned a car from the brand. Page 3 will also need to be hidden if none of the models listed on page 2 were owned by the respondent.

Before we set up survey flow, there is one piece of "housekeeping" we will do to make the setup of flows easier.

Flow Control and page looping both refer to pages as part of their setup. The time has come to name our pages to make them easier to distinguish.

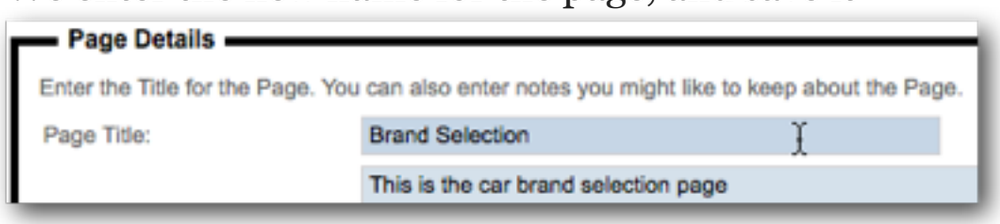
Renaming pages is simply a matter of editing each page as follows:

1. We ensure we are on the page we want to name



2. We click the *Edit button* to edit the page details

3. We enter the new name for the page, and save it



4. The new page name will be visible at the top of the page

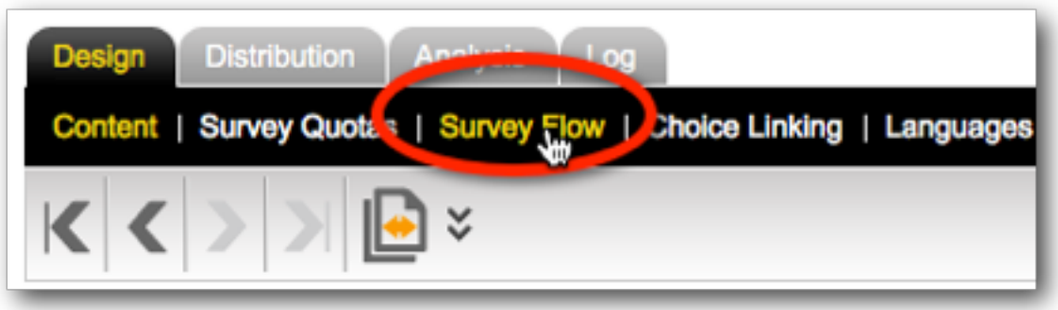
We will change all our page names as follows:

Page 1	Brand Selection
Page 2	Brand Opinion
Page 3	Models Owned
Page 4	Model Opinion

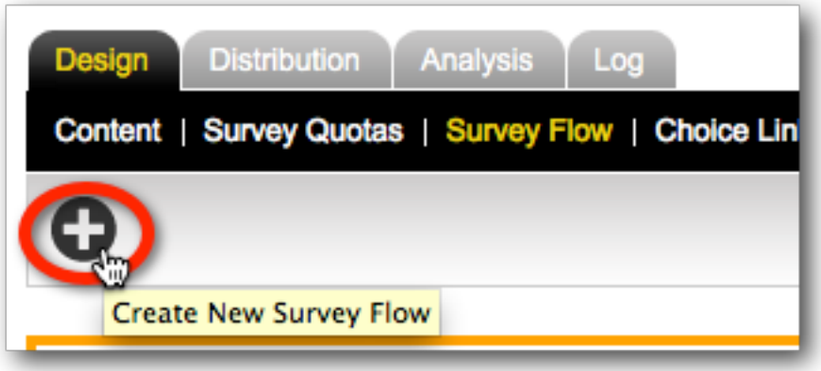
We are now ready to set up our flows. Flow control can all be set from the *Survey Flow* menu in Web Survey Creator.

We can to add each flow by:

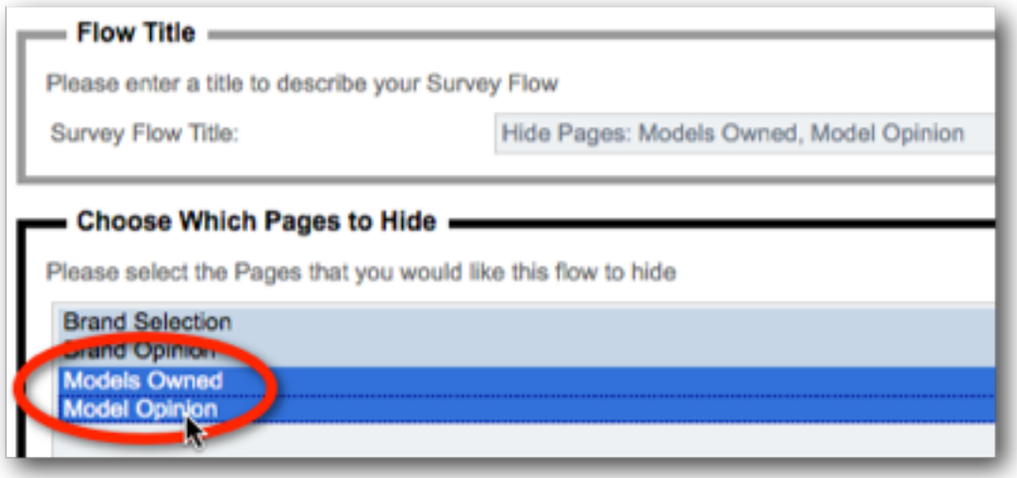
1. Clicking on the *Survey Flow* menu



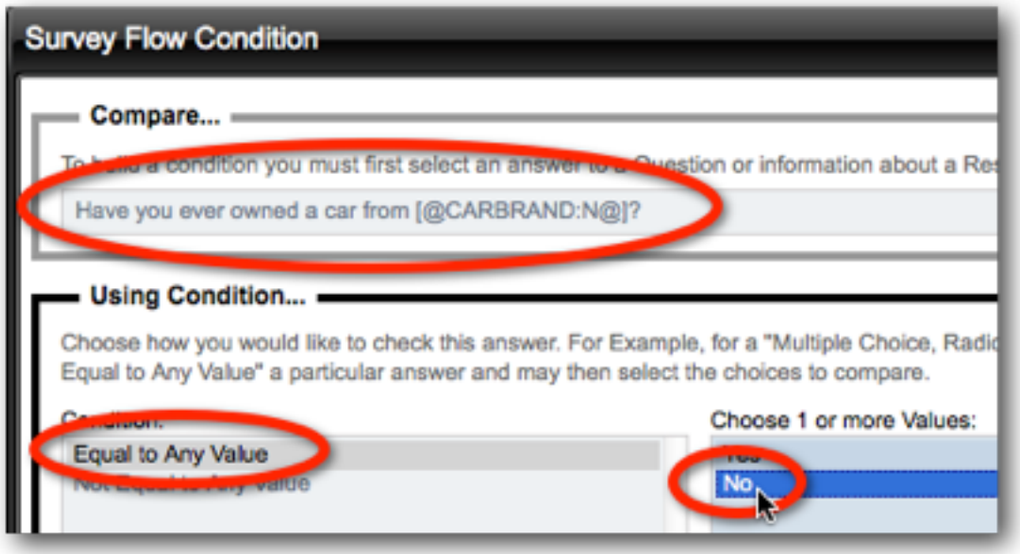
2. Clicking the *New Survey Flow button* on the toolbar



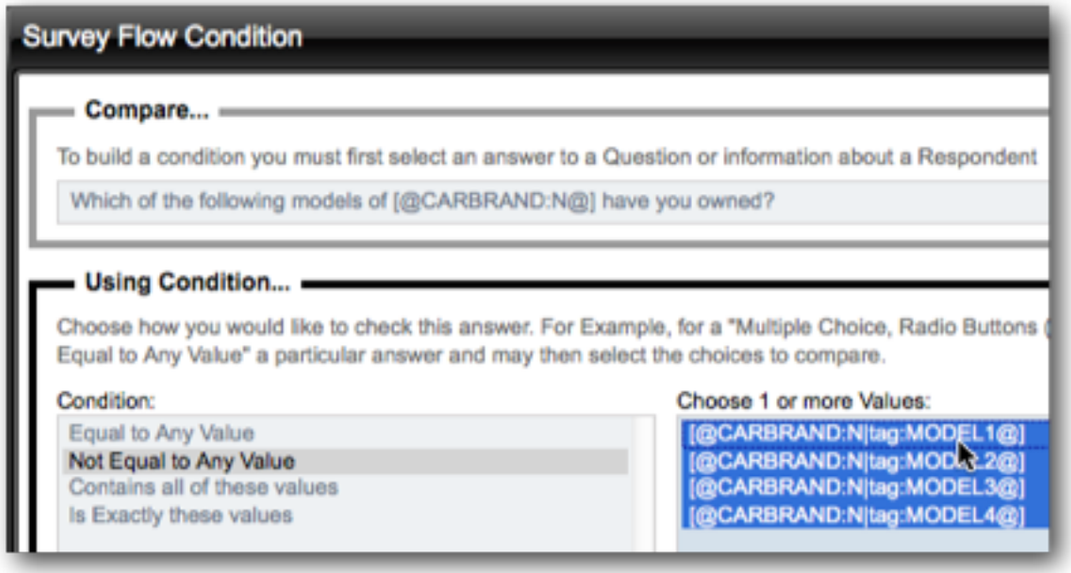
3. Choosing the pages we want to hide for the flow



4. Choosing the rule to use to hide the pages chosen



The flow relating to models can be created in the same way. The rule we would apply for hiding the "Model Opinion" page would be to check if none of the models were selected.

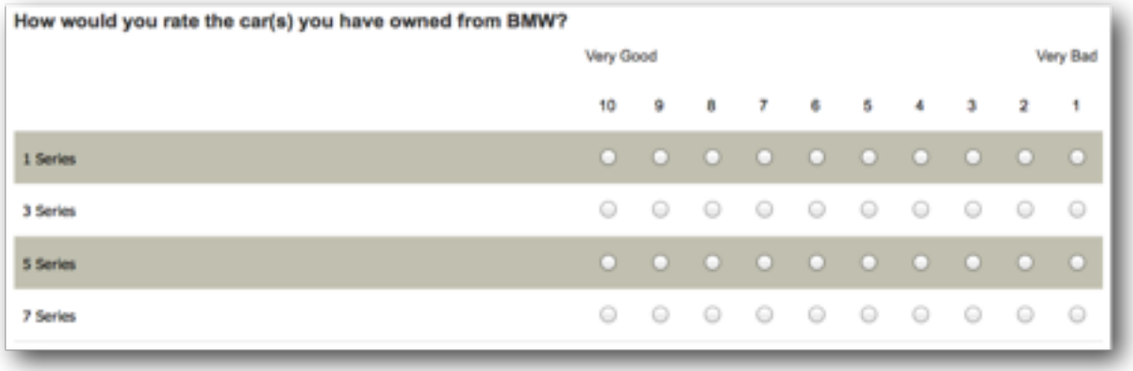


Choice Linking

Choice linking is used to only display appropriate choices in a question based on choices made earlier in the survey. The best way to understand this is to consider our example.

We have a question about models that can look as follows:

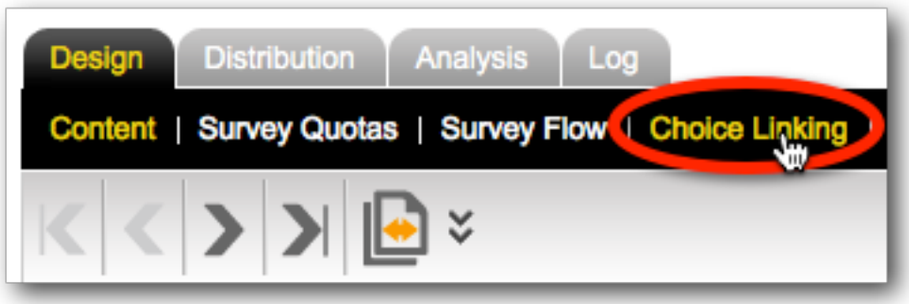
If we answer that we have owned a 1-Series and a 3-Series, it doesn't make sense for the next question to be:



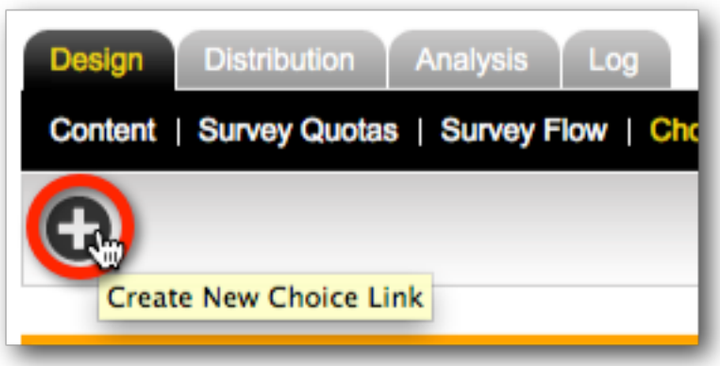
It doesn't make sense to ask about 5-Series and 7-Series models, because I have indicated I have never owned them. This is where **Choice Linking** is used - we only want to show rows in the matrix for the models that have actually been chosen.

We set up choice linking by:

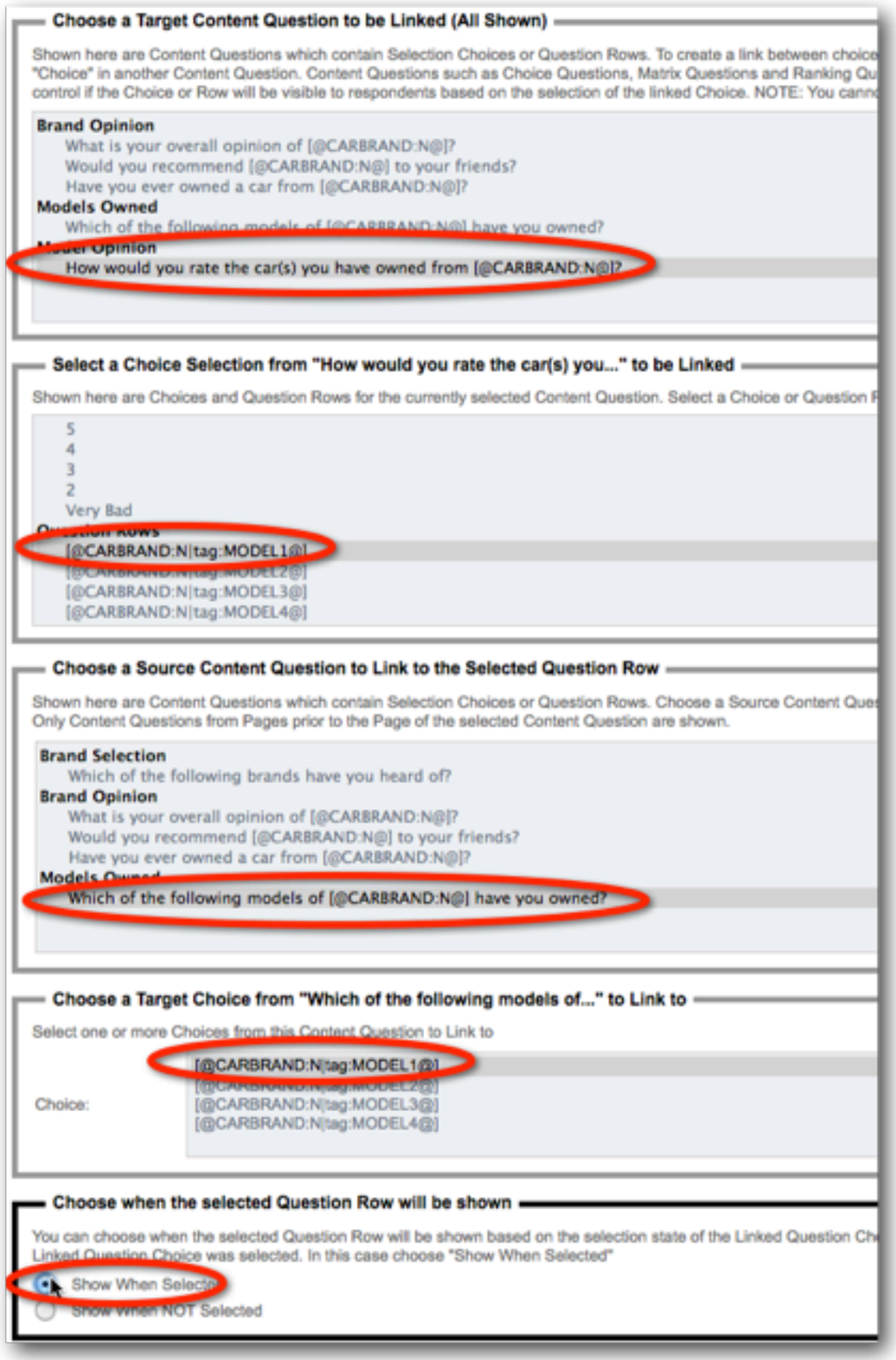
1. Clicking on the *Choice Linking* menu



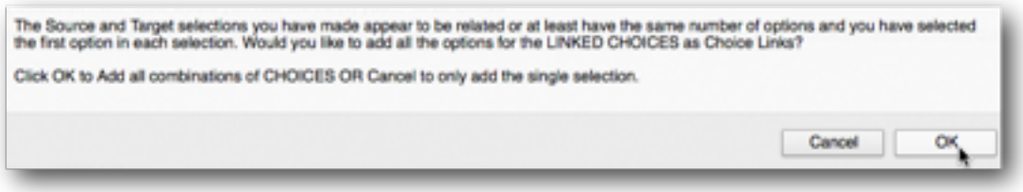
2. Clicking the *New Choice Link* button



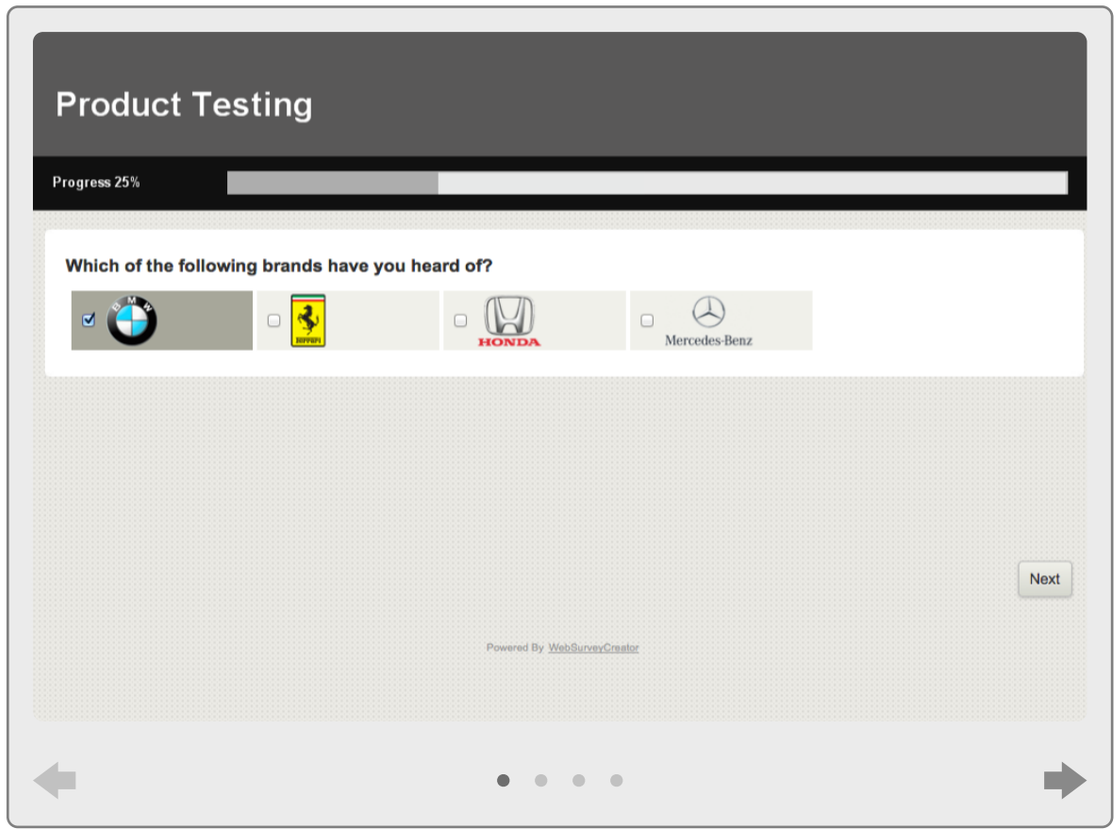
3. Set up the Choice Link



4. Accept the assistance to complete choice links for other choices



We now have a four page survey with all the content we need to create our Page Loops.



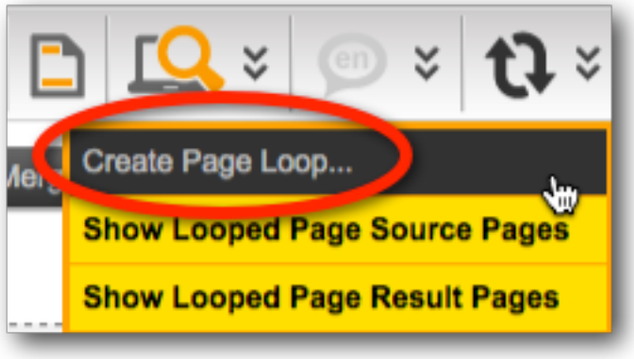
Building the Page Loops

Setting up Page Loops is all about preparation. Everything we have done so far in this section provides an indication of the sort of preparation that is necessary.

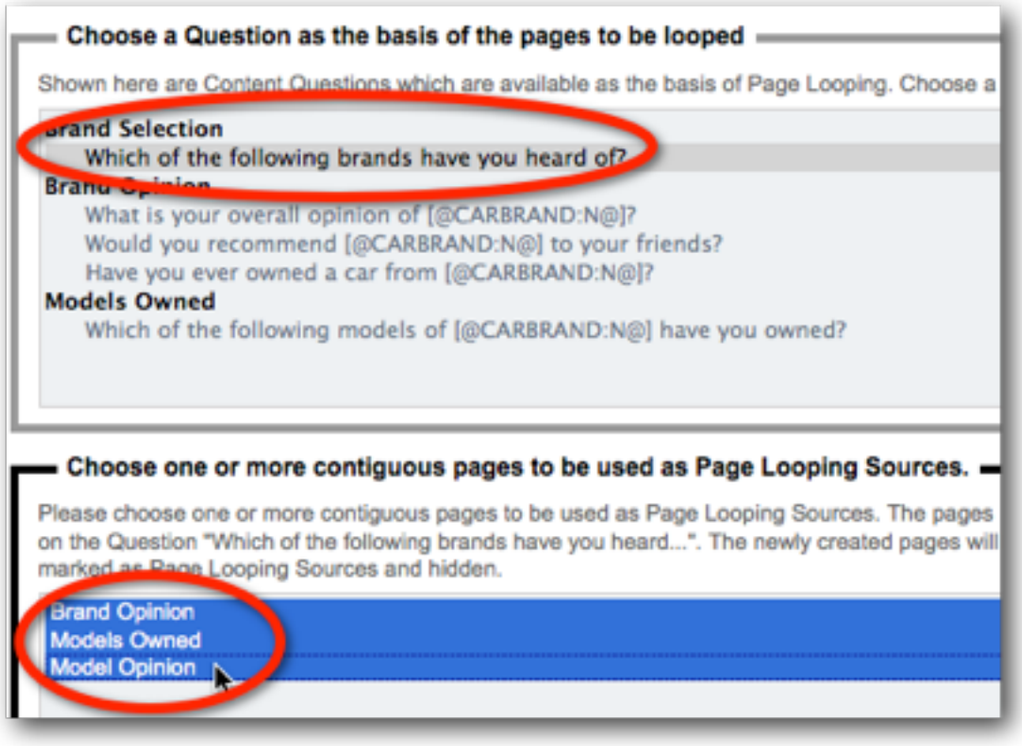
Setting up of Page Loops themselves is always the last thing you should do in your design. Page Looping is effectively a massive copy operation, so anything you have missed prior to building the loops will be missed in every single loop. Check and double check your work!

Building Page Loops is a very straightforward process. To build them we:

1. Click on the *Page Loop toolbar button* and choose *Create Page Loop* from the drop-down menu

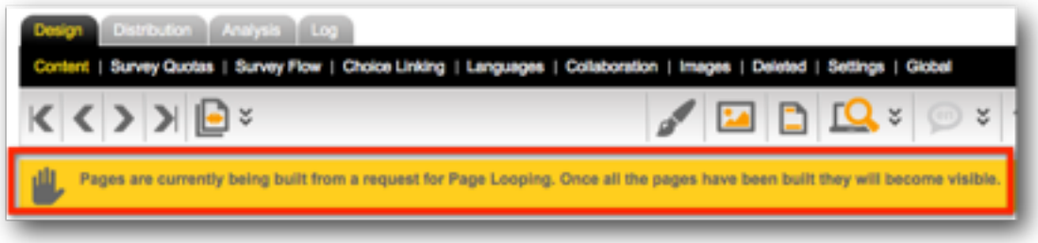


2. Choose the *Brand Question* as the basis for the loop, and the three brand pages for looping

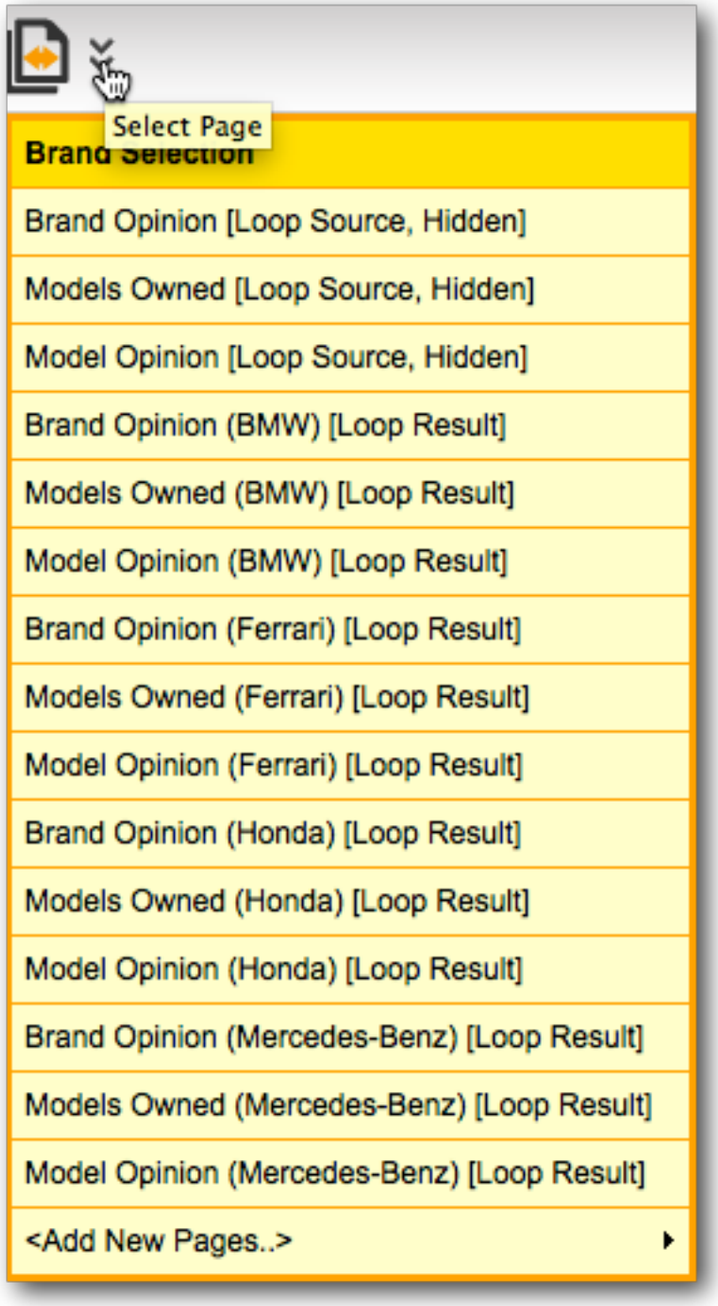


3. Click the *Create New Page Loop* button

4. We will be returned immediately to the Design tab - the looped pages will be created in the background. A warning message will show indicating that the looped pages are not yet created



5. Once the process is completed, the warning will disappear, and we will be left with our looped pages in our design



Page Looping - What just happened?!

It is important to understand exactly what happens when we use page looping. Let's consider what happened to the four pages we had in our "pre-looped" survey.

Brand Selection Page

This page remained exactly as it was - it was not looped itself. Other pages were looped based on what the brand question on this page had as choices.

Original Opinion Pages

Our original pages are still in the survey. They formed the basis of all the looped pages. They are exactly like they always were, except:

1. They are now flagged as "Looped Source" pages
2. They are now hidden

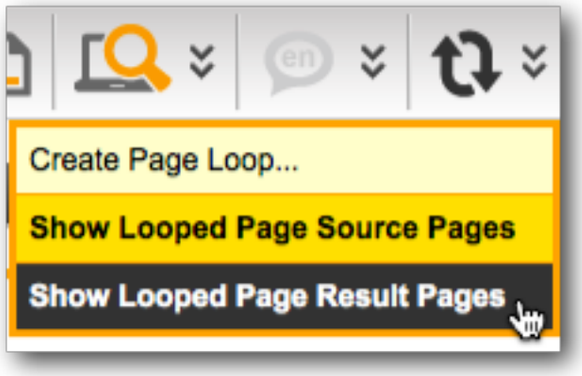
These pages will never be shown to a respondent. They are kept in the survey mainly as a backup - if you ever need to delete your looped pages, and recreate them, you will want to recreate them from these original pages.

Looped Opinion Pages

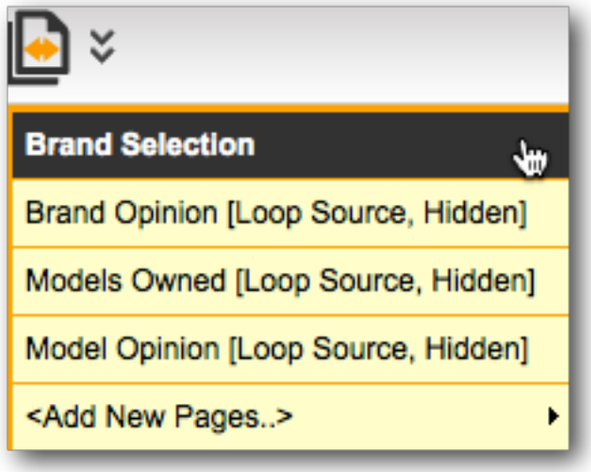
These are the pages that have been created for each of the choices in the source question. They use the original name of the page as part of the page name (another good reason we named these pages properly) together with the value of each of the choices that the pages have been copied for.

What do I do with all these pages?

It's not hard to see that the number of pages shown in a survey can grow very quickly if page loops based on a large number of choices are set up. This can get unwieldy very quickly. Fortunately you can turn Loop Source Pages and Loop Result Pages off very easily from the *Page Loop toolbar button*.



If we turned off Page Loop Result Pages, our list of pages looks very familiar to the pre-looping design.



Note that you can not do certain functions, like Page merging or moving, unless all pages are shown.

Page Looping DOs and DON'Ts

DO ensure you have set up all flows, piping and choice linking in the pages that will be included in the loop

DON'T modify loop result pages unless you have a specific reason for doing so. Any changes will ONLY be made on that one specific page

DON'T add or remove looped pages in a piecemeal fashion if you can avoid it.

DO delete all loop result pages and rebuild the entire page loop if there are changes to the source question (assuming there are no responses)

CHAPTER 5

Scripting

Providing the ability to write script in a Web Survey gives a level of flexibility that is simply not possible using built-in functionality.

Scripting provides a “do anything” capability to surveys that takes the customizability of your Web Surveys to a whole new level.

SECTION 1

Survey Scripting

IN THIS SECTION

Scripting in Contact Profiler

Regardless of what capabilities exist in a Web Survey tool, the most complex market research surveys always need "something else". There is no way to prepare for every possible situation, so the alternative is to provide a really customizable scripting language that allows the survey designer to create any logic they need, and manipulate existing questions and data.

The Premium and Ultimate MR versions of Web Survey Creator provide scripting capabilities using the Javascript client-side scripting language.

USING JAVASCRIPT

The key to any good scripting system is a strong scripting language. Fortunately all modern browsers support JavaScript.

"JavaScript was formalized in the ECMAScript language standard and is primarily used in the form of client-side JavaScript, implemented as part of a Web browser in order to provide enhanced user interfaces and dynamic websites. This enables programmatic access to computational objects within a host environment." *- Wikipedia*

Hooking into Events

Javascript is a client-side scripting tool, so any script that is run has to be initiated when a particular “event” occurs. There are five events that can have custom JavaScript executed.

1. When the Survey Page has Loaded
2. When the Next or Submit Buttons are tested for their Visibility
3. Before the Survey Page is Validated
4. When the Next or Submit Buttons are Pressed
5. When the Previous Button is Pressed

Scripting Objects

When a custom JavaScript is executed you will have access to two objects. These objects allow you access to the questions that are exposed on the current page and additional help methods that can help you to perform various tasks.

1. args
2. wscScripting

args

args contains a single item isValid that can be used to set the status of an event. This is particularly relevant for confirming to the event engine that you wish to continue the current process. For example, you must set the value to true on Next or Previous Button events or those processes halted and will not continue.

Property:	**isValid**
Return Value:	boolean - Is the current process Valid

Example:	var isOkay = true;
	if (isOkay) {
	  // All my changes allow me to continue
	  args.isValid = true;
	}

wscScripting

The following methods available in the wscScripting object. Some methods contain a method and an identical method post-fixed with the number 2. These methods are used where the question has two (2) choice ranges.

For example, Dual Range Matrix questions consist of a Primary Range and a Secondary Range.

In these circumstances the Secondary Range can be utilized by using the methods with a post-fix of 2. For example. getChoiceByValue2(question, value). In this document methods post-fixed with a number of 2 will be documented only in their primary method. Each method explanation will denote if the method has a second range capability.

A listing of the methods available for the wscScripting object are shown below. Detailed explanations are provided in the next section.

- clearValidation(question)
- derankChoice(question, choice)

- deselectChoice(question, choice)
- deselectChoice2(question, choice)
- deselectChoiceByValue(question, value)
- deselectChoiceByValue2(question, value)
- deselectMatrixChoice(question, choice, row)
- disableQuestion(question)
- enableQuestion(question)
- getABTesting()
- getBrowserData()
- getChoiceByTagValue(question, tagName, value)
- getChoiceByTagValue2(question, tagName, value)
- getChoiceByValue(question, value)
- getChoiceByValue2(question, value)
- getDateStringFromDate(date)
- getDirection()
- getDisplayType()
- getDistribution()
- getElementById(id)
- getEventData(name)
- getLanguageId()
- getQuestionByDataPipingCode(dataPipingCode)
- getQuestionByIdentity(identity)
- getQuotaByCode(code)
- getQuotaByIdentity(identity)
- getRecallCode()
- getRowByTagValue(question, tagName, value)
- getSelectedChoices(question)
- getSelectedChoices2(question)
- getSelectedMatrixChoices(question, row)
- getSelectedMatrixChoices2(question, row)
- getSelectedRanks(question)
- getSubstringLeft(str, n)
- getSubstringRight(str, n)
- getTrimString(str)
- getValidators(question)
- getValue(question)

- hideElement(element)
- isAnyChoiceSelected(question, choices)
- isAnyChoiceSelected2(question, choices)
- isAnyChoiceSelectedByValue(question, values)
- isAnyChoiceSelectedByValue2(question, values)
- isAnyMatrixChoiceSelected(question, choices, row)
- isAnyMatrixChoiceSelected2(question, choices, row)
- isAnyMatrixChoiceSelectedByValue(question, values, row)
- isAnyMatrixChoiceSelectedByValue2(question, values, row)
- isChoiceSelected(question, choice)
- isChoiceSelected2(question, choice)
- isChoiceSelectedByValue(question, value)
- isChoiceSelectedByValue2(question, value)
- isMatrixChoiceSelected(question, choice, row)
- isMatrixChoiceSelected2(question, choice, row)
- isMatrixChoiceSelectedByValue(question, value, row)
- isMatrixChoiceSelectedByValue2(question, value, row)
- rankChoice(question, choice, rank)
- selectChoice(question, choice)
- selectChoice2(question, choice)
- selectChoiceByValue(question, value)
- selectChoiceByValue2(question, value)
- selectMatrixChoice(question, choice, row)
- selectMatrixChoice2(question, choice, row)
- selectMatrixChoiceByValue(question, value, row)
- selectMatrixChoiceByValue2(question, value, row)
- setEventData(name, value)
- setValidation(question, text)
- setValue(question, value)
- showElement(element)

Function Reference

Method:	**clearValidation(question)**
Parameters:	question object - object of the question
Return Value:	Nil

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	wscScripting.clearValidation(question);
	}

Method:	**derankChoice(question, choice)**
Parameters:	question object - object of the question choice object - object of a choice
Return Value:	Nil

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var choice = wscScripting.getChoiceByValue(question, 1);
	if (choice) {
	wscScripting.derankChoice(question, choice);
	}
	}

Method:	**deselectChoice(question, choice)** **second range**
Parameters:	question object - object of the question choice object - object of a choice
Return Value:	boolean - confirmation that the choice was deselected

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var choice = wscScripting.getChoiceByValue(question, 1);
	if (choice) {
	var isUnselected = wscScripting.deselectChoice(question,
	choice);
	}
	}

Method:	**deselectChoiceByValue(question, number)** **second range**
Parameters:	question object - object of the question number - value of the question choice to check
Return Value:	boolean - confirmation that the choice was deselected

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var isUnselected = wscScripting.deselectChoiceByValue(question, 1);
	}

Method:	**deselectMatrixChoice(question, choice, row)** **second range**
Parameters:	question object - object of the question choice object - object of a choice row object - object of a row
Return Value:	boolean - confirmation that the choice was deselected

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var row = wscScripting.getRowByTagValue(question, 'bankcode',
	'AMER');
	if (row) {
	var choice = wscScripting.getChoiceByValue(question, 1);
	if (choice) {
	var isUnselected =
	wscScripting.deselectMatrixChoice(question, choice, row);
	}
	}

Method:	**disableQuestion(question)** **second range**
Parameters:	question object - object of the question
Return Value:	boolean - confirmation that the choice was disabled

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var isDisabled = wscScripting.disabledQuestion(question);
	}

Method:	**enableQuestion(question)** **second range**
Parameters:	question object - object of the question
Return Value:	boolean - confirmation that the choice was enabled

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var isDisabled = wscScripting.enabledQuestion(question);
	}

Method:	**getABTesting()**
Parameters:	Nil
Return Value:	integer – Value (range 1..100) of for use by AB Testing

Example:	var ABTest = wscScripting.getABTesting();
	if (ABTest <= 50) { // Split 50:50
	}

Method:	**getBrowserData()**
Parameters:	Nil
Return Value:	object - Browser Object o.browser = Browser Name o.version = Browser Version o.OS = Operating System

Example:	var browser = wscScripting.getBrowserData();
	if (browser.OS == 'Windows') { // The respondent is on a Windows computer
	}

Method:	**getChoiceByTagValue(question, tagName, value)** **second range**
Parameters:	question object - object of the question string - name of the tag to search for string - value of the tag being searched
Return Value:	object or undefined

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var choice = wscScripting.getChoiceByTagValue(question, 'position', 'Manager');
	if (choice) {
	// I can do something with this choice
	}
	}

Method:	**getChoiceByValue(question, value)** **second range**
Parameters:	question object - object of the question number - value of the question choice to retrieve
Return Value:	object or undefined

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var choice = wscScripting.getChoiceByValue(question, 1);
	if (choice) {
	// I can do something with this choice
	}
	}

Method:	**getDateStringFromDate(date)**
Parameters:	Date = Value of Date type to be converted to a string in format usable by WSC
Return Value:	string = Newly created string in format of YYYY.MM.DD.HH.mm

Example:	var newDate = new Date();
	var newString = wscScripting.getDateStringFromDate(newDate);
	// newString contains today's date
	// e.g. 2012.01.31.16.24

Method:	**getDirection()**
Parameters:	Nil
Return Value:	string containing:- ltr = Left to Right rtl = Right to Left e.g. Arabic

Example:	var direction = wscScripting.getLanguageId();
	if (direction == 'rtl') {
	// This is a survey using a RTL language
	}

Method:	**getDisplayType()**
Parameters:	Nil
Return Value:	string containing:- standard = Standard Display tablet = Tablet Computer e.g. iPad mobile = Mobile Phone / Cellular Phone e.g. iPhone

Example:	var display = wscScripting.getDisplayType();
	if (display == 'tablet') {
	// This is a tablet based display
	}

Method:	**getDistribution()**
Parameters:	Nil
Return Value:	object or undefined

Example:	var object = wscScripting.getDistribution();
	if (object) {
	// I can do something with this object
	}

Method:	**getElementById(id)**
Parameters:	string - Identity of an Html Element
Return Value:	object or undefined

Example:	var object = wscScripting.getElementById('mycontrol');
	if (object) {
	// I can do something with this element
	}

Method:	**getEventData(name)**
Parameters:	string - Identity of an item of data temporarily stored for later use on the current page only
Return Value:	value or undefined

Example:	var object = wscScripting.getEventData('myvalue');
	if (object) {
	// I can do something with this value
	// Value contains the text 'Hello World!'
	}

Method:	**getLanguageId()**
Parameters:	Nil
Return Value:	string - Two Character Language Code

Example:	var language = wscScripting.getLanguageId();
	if (language == 'fr') {
	// This is a survey using French Language
	}

Method:	**getQuestionByDataPipingCode(dataPipingCode)**
Parameters:	string - Data Piping Code of a Question - Must be a WSC Data Piping Code. If the question is not on the current page then you should use a Data Piping ShortCut to include the question on the current page
Return Value:	object or undefined

Example:	// Using a data piping code
	var object = wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (object) {
	// I can do something with this question
	}
	
	// Using a data piping symbol if the Question is not on the same page
	// The data piping symbol with the code #data# is required to
	// tell the system to have the question available
	var object2 =
	wscScripting.getQuestionByDataPipingCode('[@mydatapipingcode#data#@]');
	if (object2) {
	// I can do something with this question
	}

Method:	**getQuestionByIdentity(identity)**
Parameters:	string - Identity of a Question - Must be a WSC internal identity
Return Value:	object or undefined

Example:	var object = wscScripting.getQuestionByIdentity('61ce3764-1288-e111-8eae-0019b9c4ecf3');
	if (object) {
	// I can do something with this question
	}

Method:	**getQuotaByCode(code)**
Parameters:	string – Code of the Quota
Return Value:	quota object – object of the quota
Example:	var oQuota = wscScripting.getQuotaByCode('GENDER');

Method:	**getQuotaByIdentity(identity)**
Parameters:	string - Identity of a Quota - Must be a WSC internal identity
Return Value:	quota object – object of the quota

Example:	var object = wscScripting.getQuotaByIdentity('61ce3764-1288-e111-8eae-0019b9c4ecf3');
	if (object) {
	// I can do something with this quota
	}

Method:	**getRecallCode()**
Parameters:	Nil
Return Value:	string - Unique Code which identifies the response
Example:	var recallCode = wscScripting.getRecallCode();

Method:	**getRowByTagValue(question, tagName, value)**
Parameters:	question object - object of the question string - name of the tag to search for string - value of the tag being searched
Return Value:	object or undefined

Method:	**getSelectedChoices(question)** **second range**
Parameters:	question object - object of the question
Return Value:	array or undefined

Example:	var question = wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var selectedChoices = wscScripting.getSelectedChoices(question);
	if (selectedChoices) {
	// I can do something with this array
	}
	}

Method:	**getSelectedMatrixChoices(question, row)** **second range**
Parameters:	question object - object of the question row object - object of a row
Return Value:	array or undefined

Example:	var question = wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var row = wscScripting.getRowByTagValue(question, 'bankcode', 'AMER');
	if (row) {
	var selectedChoices = wscScripting.getSelectedMatrixChoices(question, row);
	if (selectedChoices) {
	// I can do something with this array
	}
	}
	}

Method:	**getSelectedRanks(question)**
Parameters:	question object - object of the question
Return Value:	array or undefined

Example:	var question = wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var ranks = wscScripting. getSelectedRanks(question);
	}

Method:	**getSubstringLeft(string, number)**
Parameters:	string = Base string from which a new string will be extracted number = Number of Characters from the Left side of the string to be extracted
Return Value:	string = Newly extracted string
Example:	var newString = wscScripting.getSubstringLeft('Hello World!', 5); // newString contains 'Hello'

Method:	**getSubstringRight(string, number)**
Parameters:	string = Base string from which a new string will be extracted number = Number of Characters from the Right side of the string to be extracted
Return Value:	string = Newly extracted string
Example:	var newString = wscScripting.getSubstringLeft('Hello World!', 5); // newString contains 'orld!'

Method:	**getTrimString(string)**
Parameters:	string = Base string from which a new string will be created with spaces at either end of the string removed
Return Value:	string = Newly created string
Example:	var newString = wscScripting.getTrimString(' Hello World! '); // newString contains 'Hello World!'

Method:	**getValidators(question)**
Parameters:	question object - object of the question
Return Value:	array

Example:	Var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	  var validators = wscScripting.getValidators(question);
	  if (validators) {
	  // I can do something with this array of validators
	  }
	}

Method:	**getValue(question)**
Parameters:	question object - object of the question
Return Value:	value or undefined dependent on the question type value only suitable for SingleText, MultipleText, DemographicEmail, DemographicPhone, Number, Slider and DateTime Questions

Example:	Var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	  var value = wscScripting.getValue(question);
	}

Method:	**hideElement(string)**
Parameters:	string = id of an Html control to hide
Return Value:	boolean - confirmation that the control was hidden
Example:	wscScripting.hideElement('mydiv');

Method:	**isAnyQuestionChoiceSelected(question, choices)**
Parameters:	question object - object of the question array - array of choice objects to check
Return Value:	boolean - confirmation that the choice is selected

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question && question.choices) {
	  // Make an array of just 1 choice
	  var arrayChoices = new Array();
	  arrayChoices.push(question.choices[0]);
	  var isSelected = wscScripting.isAnyQuestionChoiceSelected(question, arrayChoices);
	}

Method:	**isAnyQuestionChoiceSelectedByValue(question, values)** **second range**
Parameters:	question object - object of the question array - array of number values to check
Return Value:	boolean - confirmation that the choice is selected

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question && question.choices) {
	// Make an array of values
	var arrayChoices = new Array(1, 2, 3);
	var isSelected =
	wscScripting.isAnyQuestionChoiceSelectedByValue(question, arrayChoices);
	}

Method:	**isAnyQuestionMatrixChoiceSelected(question, choices)** **second range**
Parameters:	question object - object of the question array - array of choice objects to check row object - object of the row
Return Value:	boolean - confirmation that the choice is selected

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question && question.choices) {
	var row = wscScripting.getRowByTagValue(question, 'bankcode',
	'AMER');
	if (row) {
	// Make an array of just 1 choice
	var arrayChoices = new Array();
	arrayChoices.push(question.choices[0]);
	var isSelected =
	wscScripting.isAnyQuestionChoiceSelected(question, arrayChoices);
	}
	}

Method:	**isAnyQuestionMatrixChoiceSelectedByValue(question, values, row)** **second range**
Parameters:	question object - object of the question array - array of number values to check row object - object of the row
Return Value:	boolean - confirmation that the choice is selected

Method:	**isQuestionChoiceSelected(question, choice)** **second range**
Parameters:	question object - object of the question choice object - object of a choice
Return Value:	boolean - confirmation that the choice is selected

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var choice = wscScripting.getChoiceByValue(question, 1);
	if (choice) {
	var isSelected =
	wscScripting.isQuestionChoiceSelected(question, choice);
	}
	}

Method:	**isQuestionChoiceSelectedByValue(question, value)** **second range**
Parameters:	question object - object of the question number - value of the question choice to check
Return Value:	boolean - confirmation that the choice is selected

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var isSelected =
	wscScripting.isQuestionChoiceSelectedByValue(question, 1);
	}

Method:	**isQuestionMatrixChoiceSelected(question, choice, row)** **second range**
Parameters:	question object - object of the question choice object - object of a choice row object - object of the row
Return Value:	boolean - confirmation that the choice is selected

Method:	**isQuestionMatrixChoiceSelectedByValue(question, value, row)** **second range**
Parameters:	question object - object of the question number - value of the question choice to check row object - object of the row
Return Value:	boolean - confirmation that the choice is selected

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var row = wscScripting.getRowByTagValue(question, 'bankcode',
	'AMER');
	if (row) {
	var isSelected =
	wscScripting.isQuestionChoiceSelectedByValue(question, 1, row);
	}
	}

Method:	**selectChoice(question, choice)** **second range**
Parameters:	question object - object of the question choice object - object of a choice
Return Value:	boolean - confirmation that the choice was selected

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var choice = wscScripting.getChoiceByValue(question, 1);
	if (choice) {
	var isSelected = wscScripting.selectChoice(question, choice);
	}
	}

Method:	**selectChoiceByValue(question, number)** **second range**
Parameters:	question object - object of the question number - value of the question choice to check
Return Value:	boolean - confirmation that the choice was selected

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var isSelected = wscScripting.selectChoiceByValue(question, 1);
	}

Method:	**selectMatrixChoice(question, choice, row)** **second range**
Parameters:	question object - object of the question choice object - object of a choice row object - object of a row
Return Value:	boolean - confirmation that the choice was selected

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	var row = wscScripting.getRowByTagValue(question, 'bankcode',
	'AMER');
	var choice = wscScripting.getChoiceByValue(question, 1);
	if (row && choice) {
	var isSelected = wscScripting.selectMatrixChoice(question,
	choice, row);
	}
	}

Method:	**setEventData(name, value)**
Parameters:	string - Identity of an item of data temporarily stored for later use on the current page only object - Value of an item of data temporarily stored for later use on the current page only
Return Value:	boolean - confirmation that the value was correctly added
Example:	wscScripting.setEventData('myvalue', 'Hello World!');

Method:	**setValidation(question, text)**
Parameters:	question object - object of the question string - text of the validation message
Return Value:	Nil

Example:	var question =
	wscScripting.getQuestionByDataPipingCode('mydatapipingcode');
	if (question) {
	wscScripting.setValidation(question, 'Something doesnt make sense!');
	}

Method:	**setValue(question, value)**
Parameters:	question object - object of the question value - type dependent on question type value only suitable for SingleText, MultipleText, DemographicEmail, DemographicPhone, Number, Slider and DateTime Questions
Return Value:	Nil

Example:	`var question = wscScripting.getQuestionByDataPipingCode('mydatapipingcode');`
	`if (question) {`
	`  wscScripting.setValue(question, 'Hello World!');`
	`}`

Method:	**showElement(string)**
Parameters:	string = id of an Html control to show
Return Value:	boolean - confirmation that the control was shown
Example:	`wscScripting.showElement('mydiv');`

Additional Objects

The following objects exist and have the properties as described with type and name.

Note: You do not have the ability to affect the rendering of a standard question by altering a property.

SurveyQuestion

- *string* **addressType**
- *string* **allRankedText**
- *array* [surveychoice] **choices**
- *array* [surveychoice] **choices2**
- *string* **clearText**
- *string* **containerName**
- *string* **dataPipingCode**
- *number* **defaultValue**
- *string* **fieldWidth1**
- *string* **fieldWidth2**
- *string* **formatType**
- *string* **gridHeadingFormat**
- *number* **gridTotal**
- *string* **identity**
- *number* **increment**
- *number* **interval**
- *boolean* **isCommentsEnabledByDefault**
- *boolean* **isHeadingTextVertical**
- *boolean* **isLargeComments**
- *boolean* **isLength**
- *boolean* **isMandatory**

- *boolean* **isPivot**
- *boolean* **isQuestionOnPage**
- *boolean* **isResetAllowed**
- *boolean* **isSpecify**
- *string* **javascriptBodyName**
- *string* **listDirection**
- *string* **listType**
- *number* **maxIncrement**
- *number* **maxValue**
- *number* **minValue**
- *string* **noneRankedText**
- *number* **numberGrids**
- *string* **popupType**
- *string* **primaryRangeTitle**
- *string* **questionNumber**
- *string* **rankedText**
- *number* **repeatRows**
- *string* **resetText**
- *string* **rowHeight1**
- *string* **rowHeight2**
- *array* [surveyrow] **rows**
- *number* **scaleIncrement**
- *string* **secondaryRangeTitle**
- *string* **text**
- *string* **textPosition**
- *string* **type**
- *string* **unRankedText**

SurveyChoice

- *number* **grid**
- *string* **identity**
- *string* **imageHeight**
- *string* **imageToolTip**
- *string* **imageUrl**
- *string* **imageWidth**
- *boolean* **isComments**
- *boolean* **isDefault**

- *boolean* **isExclusive**
- *boolean* **isPegged**
- *string* **labelText**
- *string* **numberPostText**
- *string* **numberPreText**
- *array* [surveychoicetag] **tags**
- *string* **text**
- *number* **value**

SurveyChoiceTag

- *string* **identity**
- *string* **name**
- *string* **text**

SurveyRow

- *string* **identity**
- *string* **imageHeight**
- *string* **imageToolTip**
- *string* **imageUrl**
- *string* **imageWidth**
- *array* [surveyrowtag] **tags**
- *string* **text**

SurveyHierarchicalListItem

- *string* **identity**
- *string* **description**
- *string* parent **Identity**

SurveyRowTag

- *string* **identity**
- *string* **name**
- *string* **text**

SurveyQuota

- *string* **code**
- *string* **identity**
- *bool* **isPriority**
- *int* **numberLimit**
- *int* **numberAllowed**
- *int* **numberResponded**
- *int* **numberOverflow**
- *string* **title**

SurveyDistribution

- *string* **identity**
- *array* [surveydistributiontag] **tags**
- *string* **title**

SurveyDistributionTag

- *string* **identity**
- *string* **name**
- *string* **text**

Browser

- *string* **browser**
- *string* **OS**
- *string* **version**

Chapter 6

On-Premise Software

The On-Premise release of Web Survey Creator provides the ultimate control over data and processing capacity.

SECTION 1

WSC On-Premise Version

A discussion of Web Survey Creator for Market Research wouldn't be complete without a brief mention of the On-Premise version. This release is the highest version of the software, featuring:

- Unlimited Surveys
- Unlimited Responses
- Unlimited Users

While the hosted version of Web Survey Creator runs on our servers, the on-premise version can either be:

- Run on your own servers
- Run as an isolated installation on our servers

On-Premise Exclusive Features

The On-Premise version of Web-Survey Creator has some exclusive features as follows:

- Exclusive database & processing (no sharing)
- New User Signup Capabilities
    - Quick signup for new users
    - Domain name based control of signup
    - User auto verification
    - Online chat support for defined supported users
- User Groups for Collaboration
- Setting of sending details for Emails

A/B testing Random Number

Whenever a response is commences in Web Survey Creator, a random number is calculated and placed on the response. This number can be used for a number of things, however the primary use is to provide a consistent “A/B testing random number” that can be used by flow control. Pages can be hidden based on the value of the testing number (a number between 1 and 100).

Related Glossary Terms

Drag related terms here

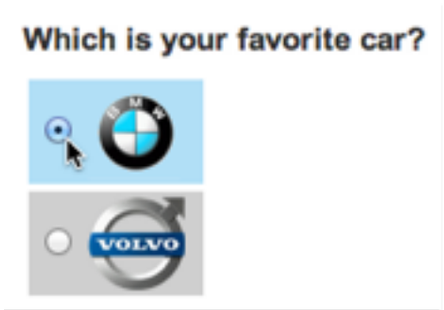
Index Find Term

Chapter 1 - Dealing with Bias

Choice Linking

Choice linking refers to the ability to hide a choice in a question based on whether a choice has been selected in another question. A classic use of choice linking would be:

The respondent's favorite car brand has to come from the list of those cars owned.



The cars that weren't chosen are not shown - they are hidden using choice linking.

Related Glossary Terms

Drag related terms here

Index Find Term

Chapter 4 - Product Testing Example

Choice Tags

Choice tags are pieces on information on choices that go beyond just the choice name. For example, a choice for a car brand question could be “BMW. Choice tags can be used to store additional information about the choice, such as the history of the company, or models available.

Related Glossary Terms

Drag related terms here

Index Find Term

Data Piping

Data piping allows details to be piped from earlier questions in a survey to later questions.

Related Glossary Terms

Drag related terms here

Index Find Term

Quota Builder

The quota builder uses a wizard-style interface to take you step-by-step through the creation of quotas based on multiple questions. For example, the quota builder can take these questions:

Gender
- Male
- Female

Age
- 18 to 30
- 31 to 60
- 61 and older

and build the following quotas:

Male - 18 to 30
Male - 31 to 60
Male - 61 and older
Female - 18 to 30
Female - 31 to 60
Female - 61 and older

Related Glossary Terms

Drag related terms here

Index Find Term

Chapter 2 - Quota Management Example

Terminate Pages

A Terminate Page ends the survey. It will have a submit button on it, rather than a next button, even if it appears early in the survey. Terminate pages are generally used to Quota Out and Screen Out respondents who are not needed in the survey.

Related Glossary Terms

Drag related terms here

Index Find Term

Chapter 2 - Quota Management Explained