Advanced Reporting Services

Sample manual - first two chapters

Wise Owl Training

TABLE OF CONTENTS (1 of 5)

1	GROUPING IN TABLES	Page
1.1	Grouping Basics	7
1.2	Grouping Rows in a Table	8
	Step 1 – Build a Basic Table	8
	Step 2 – Apply Grouping to the Detail Row	8
	Step 3 – Choose Which Field to Group By	9
	The End Result	9
1.3	Removing Groups	10
	Deleting Groups in the Grouping Pane	10
	Deleting Groups in a Table	10
1.4	Formatting a Grouped Table	11
	Adding Group Headers and Footers	11
	Rearranging Group Titles	12
	Adding Group Totals	13
1.5	Multiple Groups	14
	Adding a New Parent Group	14
	Adding a Child Group	14
1.6	Page Breaks	15
	Inserting Page Breaks between Groups	15
1.7	Collapsible Groups	16
	Step 1 - Hiding Detail Rows	16
	Step 2 - Setting Toggle Items	17

2	EXPRESSIONS	Page
2.1	Overview of Expressions	18
	Choosing to Create an Expression	18
	The Expression Builder Dialog Box	19
	Referring to Dataset Fields	19
	Referring to Report Items	20
	Inserting Functions	20
2.2	Ad-Hoc Table Columns	21
	Creating an Ad-Hoc Column Expression	21
2.3	Calculated Fields	22
	Adding a Calculated Field to a Dataset	22
	Using a Calculated Field	23
	Adding Calculated Fields to a Query	23
2.4	Working with Numbers	24
	Basic Arithmetic	24
	Numeric Functions	24
2.5	Conditional Functions	25
	The IIf Function	25
	The Switch Function	25
	Comparison Operators	26
	Logical Operators	26
2.6	Working with Text	27
	Concatenating Text	27
	Text Functions	27
2.7	Working with Dates	28
	Returning the Current Date	28
	Date Functions	28
	Intervals for Date Functions	29
	Formatting Dates	29
2.8	Aggregating Data	30
	Aggregate Functions	30
	Adding Aggregates the Quick Way	30
	Changing the Aggregate Function	31
	Specifying the Scope	31
2.9	Lookup Functions	32
	The Lookup Function	32
	The LookupSet Function	32
2.10	Built-In Fields	33
	Built-In Fields in the Expression Builder	33
	Built-In Fields in the Report Data Window	33
2.11	Placeholders	34
	Creating a Placeholder	34

© Copyright 2024

TABLE OF CONTENTS (2 of 5)

3	ADVANCED EXPRESSIONS	Page
3.1	Scope	35
	Dataset Scope	35
	Data Region Scope	36
	Row and Group Scope	37
	Entering Aggregate Formulae	37
3.2	Row Numbers	38
	The RowNumber Function	38
	Numbering over Groups	38
	The Nothing Scope	39
	Alternate Row Colouring	39
3.3	Running Totals	40
3.4	Extra Functions	41

4	VARIABLES	Page
4.1	Using Variables	42
	The Need for Variables	42
	Creating a Variable	43
	Referencing Variables	43
4.2	How Variables are Calculated	44
4.3	Group Variables	45
	Creating a Group Variable	45
	Referring to Group Variables in Expressions	46

5	EMBEDDING CODE IN REPORTS	Page
5.1	Overview – Ways to Reuse Code	47
	Our Example	47
	Strategies to Use	47
	Writing your Expression in SQL	48
	Writing an Expression in SSRS	48
5.2	Creating an Embedded Function	49
	Step 1 - Designing your Function (Data Types)	49
	Step 2 – Writing your Function	50
	Step 3 – Embedding your Code in a Report	51
	Step 4 – Referencing your Code	51
5.3	Other Ways to Write Code	52
	Writing Code within RDL	52
	Writing Code within a Class Library	52

6	SQL SERVER DATABASE ACCESS	Page
6.1	Server Explorer / Management Studio	54
	Comparison of SSMS and Server Explorer	54
6.2	Accessing Server Explorer	55
6.3	Accessing Management Studio (SSMS)	56
	Starting to Use Management Studio	56
	Object Explorer	57
	Useful Start-up Options	57

7	VIEWS	Page
7.1	The Need for Views	58
7.2	Creating Views using the Designer	59
	Starting the Designer	59
	Choosing Columns	59
	Sorting and Filtering	60
	Adding Grouping	60
	Executing a View	61
	Saving and Closing Views	62
	Seeing your View in Object Explorer	62
	Running a View	63
	Changing a View	63
7.3	Scripting Views	64
	Creating a New View	64
	Changing an Open View in Script	65
	Changing a View's Script from Object Explorer	65
7.4	Switching between the Designer and Scripting	66
7.5	Using Views in Datasets	67
7.6	Using Views to Rename Columns	68
7.7	Pros and Cons of Views	69
	Advantages of Views	69
	Disadvantages of Views	69

© Copyright 2024

TABLE OF CONTENTS (3 of 5)

8	PARAMETERS	Page
8.1	Introduction to Parameters	70
	Displaying the Parameters Pane	70
8.2	Report Parameters	71
	Creating a Report Parameter	71
	Using a Report Parameter in a Filter	72
	Parameter Data Types	72
	Dealing with No Rows	73
	Displaying Parameter Values in the Report	73
8.3	Query Parameters	74
	Writing a Query using Parameters	74
8.4	Organising Parameters	75
	Changing the Order of Parameters	75
	Using the Parameters Pane	75
8.5	Default Values	76
	Typing in a Default Value	76
	Calculating Default Values	76
	Getting Default Values from a Dataset	77
8.6	Null Values	78
	Allowing Null Values	78
	Dealing with Nulls in Filters	78
8.7	Drop Down Lists	79
	Manually Entering Available Values	79
	Using a Dataset to Populate a List	80
8.8	Multi-Value Drop Down Lists	81
	Allowing Multiple Values	81
	Using Multiple Values in Filters	81
	Using Multiple Values in Queries	82
	Displaying Multiple Values in a Report	82
8.9	Cascading Drop Down Lists	83
	Creating Cascading Drop Down Lists	83
	Using Cascading Drop Down Lists	85
8.10	Conditional Formatting with Parameters	86
	Parameters in Formatting Expressions	86

9	STORED PROCEDURES	Page
9.1	Introduction to Stored Procedures	87
9.2	Creating Stored Procedures	88
	Typing in a Stored Procedure	88
	Executing the Script to Create your Stored Procedure	88
	Viewing your Stored Procedure	89
	Basing a Report on your Stored Procedure	89
9.3	Altering a Stored Procedure	90
	Altering an Open Stored Procedure	90
	Altering a Procedure in a Database	90
	The Need to Refresh Fields	91
9.4	Executing Stored Procedures	92
	Refreshing your Local Cache	92
	Altering and Executing a Stored Procedure Together	93
	Selecting a Stored Procedure Name to Run It	93
9.5	Renaming and Deleting Stored Procedures	94
	Renaming/Deleting a Procedure with the Menu	94
	Deleting a Procedure in Script	94
	Renaming a Procedure in Script	94
9.6	The Cheat's Stored Procedure	95

© Copyright 2024

TABLE OF CONTENTS (4 of 5)

10	PARAMETERS IN STORED PROCEDURES	Page
10.1	Overview	96
	Syntax of Parameters	96
	Parameters and SSRS Reports	96
10.2	Creating Procedures with Parameters	97
	Step 1 – Specifying the Parameters	97
	Step 2 – Coding the Parameters	97
	Step 3 – Referencing the Parameters	98
	Using Text Wildcards with Parameters	98
10.3	Testing Procedures using Parameters	99
	Passing Values to Parameters in Order	99
	Testing a Procedure using Parameter Names	100
	Right-clicking to Test a Procedure	100
10.4	Stored Procedure Parameters in SSRS Reports	101
10.5	Allowing Null Values	102
	How Null Values are Passed from SSRS	102
	The Workaround – Trap Nulls in your WHERE Condition	103
10.6	Parameter Data Types (Numbers)	104
	Integer Data Types	104
	Decimal and Numeric Types	104
	The Float Type	104
10.7	Parameter Data Types (Text)	105
	Types of Character Storage	105
	Variable Length Data Types	105
	Fixed Length Data Types	105
10.8	Parameter Data Types (Dates/Times)	106

11	STORED PROCEDURES AND DROPDOWNS	Page
11.1	Basic Dropdowns using Stored Procedures	107
	Step 1 – Creating the Two Datasets Needed	107
	Step 2 – Adding the Datasets	108
	Step 3 – Configuring the Dropdown Parameter	108
11.2	Cascading Dropdowns	109
	The Datasets and Parameters Needed	109
	The Three Stored Procedures Needed	110
11.3	Removing SELECT A VALUE	110
	How Dropdowns Look by Default	111
	How the Solution will Work	111
	Creating the Dataset with Additional Top Row	111
	Setting the Default Parameter Value	112

12	MULTIVALUE DROPDOWNS AND PROCEDURES	Page
12.1	Multivalue Parameters	113
	How Multivalue Parameters are Stored	113
12.2	Splitting Comma-Delimited Strings	114
	Creating a Table-Valued Function to Split the Parameter String	114
	Testing the Table-Valued Function	115
12.3	Creating a Multivalue Parameter	116
	Step 1 - Creating the Dropdown Dataset	116
	Step 2 - Creating the Main Report Dataset	116
	Step 3 – Adding the Datasets to the Report	117
	Step 4 – Configuring the Parameter Created	117
	Step 5 – Showing the Values Chosen	118
12.4	Coping with Multivalue Choices	119
	Intercepting SELECT ALL	119
	Setting a Maximum Number of Choices	120

13	DRILLTHROUGH REPORTS	Page
13.1	Overview of Drillthrough Reports	121
	The Example for this Chapter	121
13.2	Creating the Child Report	122
	Step 1 - Create the Dataset	122
	Step 2 - Hide the Parameter	122
	Step 3 - Design the Report	122
13.3	Creating the Parent Report	123
	Step 1 - Create the Dataset	123
	Step 2 - Design the Report	123
13.4	Creating Report Actions	124
	Step 1 - View the Action Properties	124
	Step 2 - Configure the Action to Open a Report	124
13.5	Using Drillthrough Reports	125
	Running the Parent Report	125
	Returning to the Parent Report	125
13.6	Multiple Parameters	126
	Creating Child Reports with Multiple Parameters	126
	Creating the Parent Report	126

© Copyright 2024

TABLE OF CONTENTS (5 of 5)

14	IMPROVING REPORT NAVIGATION	Page
14.1	Bookmarks	127
	Step 1 - Creating the Report	128
	Step 2 – Creating the Bookmarks to Jump to	129
	Step 3 - Creating the Links to Each Row	129
	Step 4 - Creating the Link Back to the Top	130
	Step 5 – Creating Tooltips	130
14.2	Navigation Maps	131
	Creating a Document Map	131
	Nested Document Maps	132
	Exporting Document Map Reports to Excel	133
	Setting Titles	133

15	DYNAMIC REPORTS (DATA)	Page
15.1	Dynamic Data Sources	134
15.2	Dynamic Datasets	135
	Step 1 - Creating the Base Report	135
	Step 2 - Creating the Parameter	136
	Step 3 – Make the Report Variable	136
15.3	Dynamic Stored Procedures	137
	Vary the Name of the Stored Procedure	137
	Passing a Value to a Stored Procedure Parameter	137

16	DYNAMIC REPORTS (DESIGN)	Page
16.1	Examples in this Chapter	138
16.2	The Underlying Principle	139
16.3	Example – Choosing the Grouping Field	140
	Step 1 – Create the Report Template	140
	Step 2 – Create the Parameter	141
	Step 3 - Choose the Grouping Field	142
	Step 4 – Make any References to the Fixed Field Dynamic	142
16.4	Example – Choosing the Row and Column Field for a Matrix	143
	Step 1 – Create the Report Template	143
	Step 2 – Create the First Dropdown Parameter	144
	Step 3 – Creating the Second Dropdown Parameter	145
	Step 4 – Showing the Second Parameter	145
	Step 5 – Changing the Grouping Fields	146
	Step 6 – Update other References	146
16.5	Example – Choosing the Data Field	147
	Creating the Dropdown Parameter	147
	The Data Field and Title Expressions	148
	The Number Formatting	148

© Copyright 2024

CHAPTER 1 - GROUPING IN TABLES

1.1 Grouping Basics

Applying *grouping* to a table allows you to organise the data into discrete sections. This can make it easier to display and print the data.

In a grouped table the detail rows appear in neatly organised sections. In this example we've grouped films by the country in which they were made.

Australia		
Title	**Release Date**	**Run Time**
Crocodile Dundee II	25/05/1988	112
Crocodile Dundee	26/09/1986	98
Happy Feet	17/11/2006	108
Mad Max	12/04/1979	93
Mad Max 2	24/12/1981	96
Mad Max Beyond Thunderdome	10/07/1985	107
Mad Max: Fury Road	07/05/2015	120
	Total	**734**

Brazil		
Title	**Release Date**	**Run Time**
City of God	30/08/2002	130
	Total	**130**

Canada		
Title	**Release Date**	**Run Time**
My Big Fat Greek Wedding	19/04/2002	95
Porky's	16/04/1982	98
	Total	**193**

Australia		
Action		
Title	**Release Date**	**Run Time**
Mad Max	12/04/1979	93
Mad Max 2	24/12/1981	96
Mad Max Beyond Thunderdome	10/07/1985	107
Mad Max: Fury Road	07/05/2015	120
Adventure		
Title	**Release Date**	**Run Time**
Crocodile Dundee II	25/05/1988	112
Animation		
Title	**Release Date**	**Run Time**
Happy Feet	17/11/2006	108
Comedy		
Title	**Release Date**	**Run Time**
Crocodile Dundee	26/09/1986	98
	Total	**734**

You can create multiple nested groups too. In this example we've grouped films by country and within each country we've grouped films by genre.

You can even create fancy collapsing and expanding groups to add a little interactivity to your reports.

⊞ Australia		
⊟ Brazil		
Title	**Release Date**	**Run Time**
City of God	30/08/2002	130
	Total	**130**
⊞ Canada		

Wise Owl's Hint

*You can also apply groups to columns in a table but this chapter focusses on grouping rows. The **Matrix** report item is a **Table** with pre-defined row and column groups and you'll learn about this in a later chapter.*

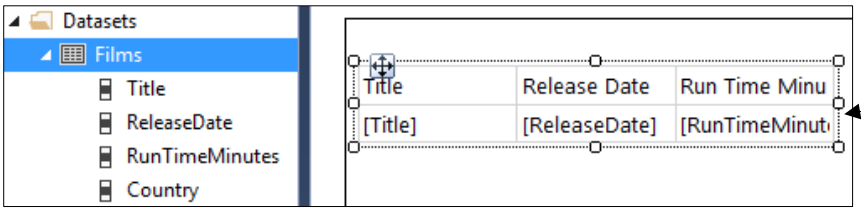
© Copyright 2024

1.2 Grouping Rows in a Table

There are several steps involved in creating a grouped table, as detailed in this section.

Step 1 – Build a Basic Table

Start by building a basic table containing the fields you want to see for each detail row.



Here we've added three fields to the table. We've deliberately not included the **Country** field as we're going to group by that field in the next step.

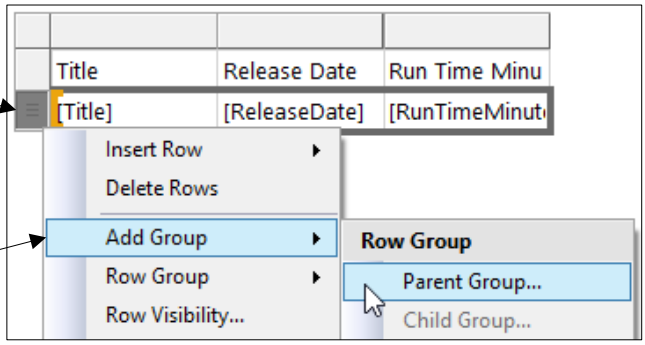
Step 2 – Apply Grouping to the Detail Row

You can choose to apply grouping to the detail row in a table using one of several techniques:

- Right-click on the detail row in the table.

a) Select a cell in the table and then right-click on the grey block to the left of the detail row (the one with the three horizontal lines).

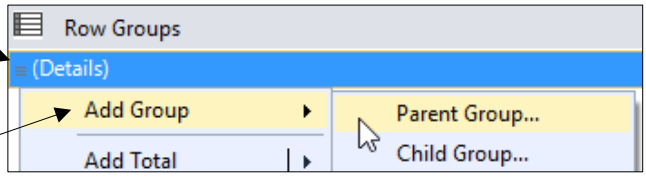
b) From the menu select **Add Group | Parent Group…**



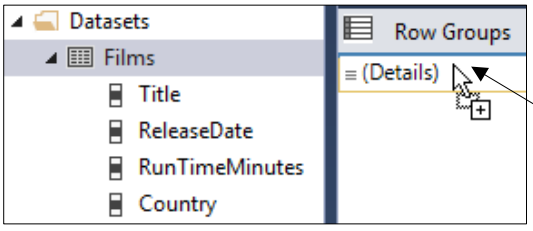
- Right-click on the detail row in the **Grouping Pane**.

a) Right-click the **(Details)** item in the **Row Groups** area of the **Grouping Pane**.

b) Select **Add Group | Parent Group…**



- Click and drag a field from the **Report Data** window to the **Grouping Pane**.

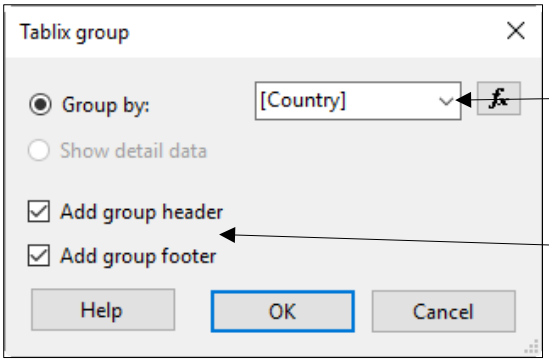


Click and drag the relevant field (here we're using the **Country** column) from the **Report Data** window onto the **(Details)** item in the **Grouping Pane**. A blue horizontal line will appear – it's important to release the mouse when the line sits above the **(Details)** item.

© Copyright 2024

Step 3 – Choose Which Field to Group By

If you used the right-click menu to add a group in Step 2 you'll need to complete a dialog box to specify which field you're grouping by.



Select the field you want to group by from this drop down list.

You can optionally add a group header and footer by ticking these boxes. This is something you can't do if you used the click and drag method in Step 2. You can see how to add a header and footer to a group manually later in this chapter.

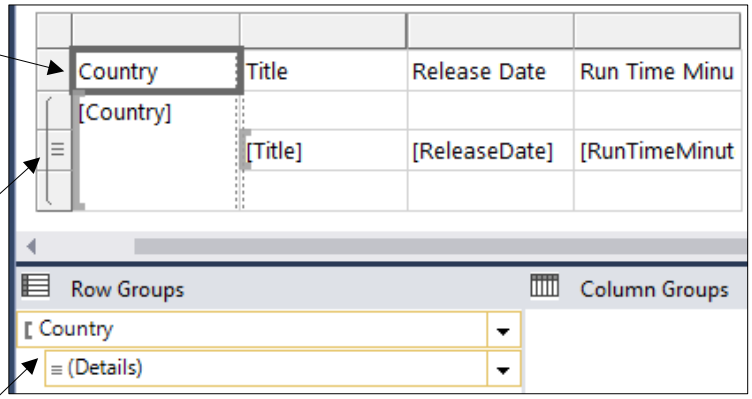
The End Result

When you've finished applying grouping to the table it should look somewhat different:

A new column containing the group field will be added to the left of the table.

The rows which belong to the group will be indicated by the **(** symbol on the left of the table. In this example we added a group header and footer so the group comprises three rows of the table in design view.

The grouping pane displays a **[** icon next to group fields. Items which belong to the group are indented to the right below the group item.



When you run the report the group region in the table will be repeated for each item belonging to the group. In this example the group is repeated for each country in the table.

Country	Title	Release Date	Run Time Minutes
Australia			
	Crocodile Dundee II	25/05/1988	112
	Crocodile Dundee	26/09/1986	98
	Happy Feet	17/11/2006	108
	Mad Max	12/04/1979	93
	Mad Max 2	24/12/1981	96
	Mad Max Beyond Thunderdome	10/07/1985	107
	Mad Max: Fury Road	07/05/2015	120
Brazil			
	City of God	30/08/2002	130

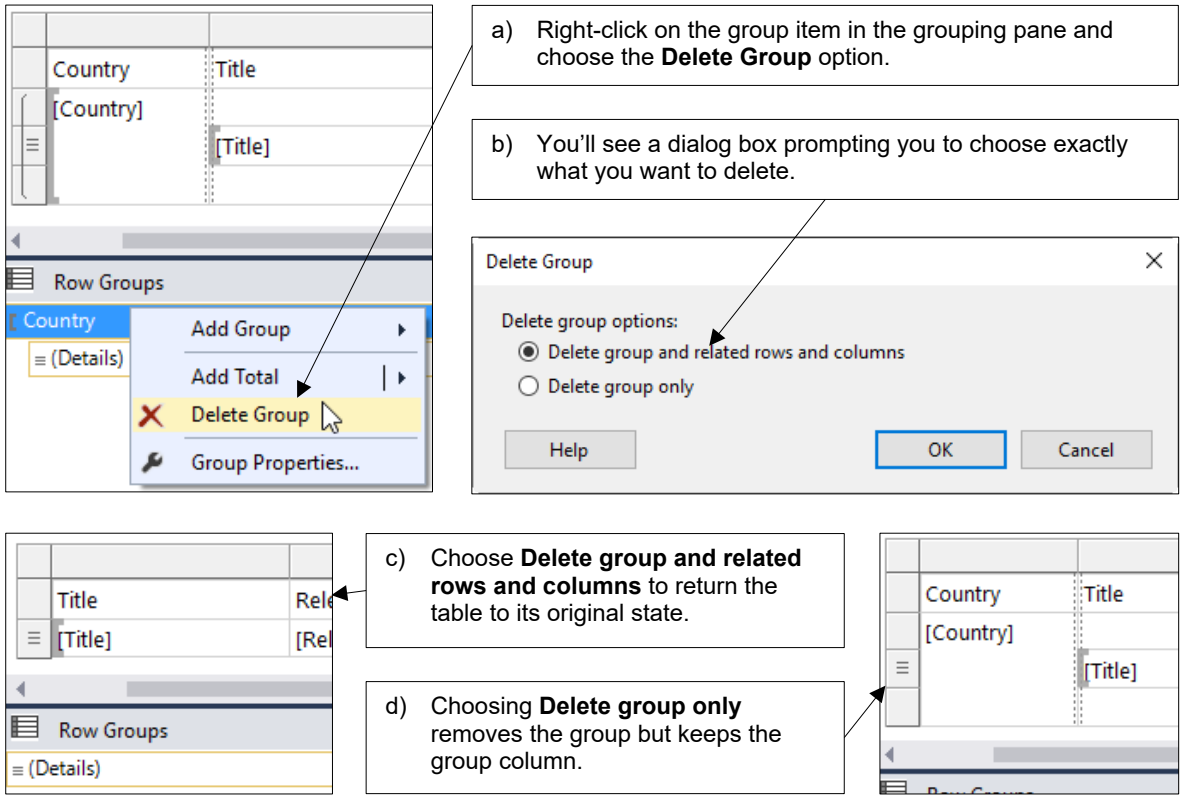
© Copyright 2024

1.3 Removing Groups

You can delete a group along with its associated rows to return your table to its original state.

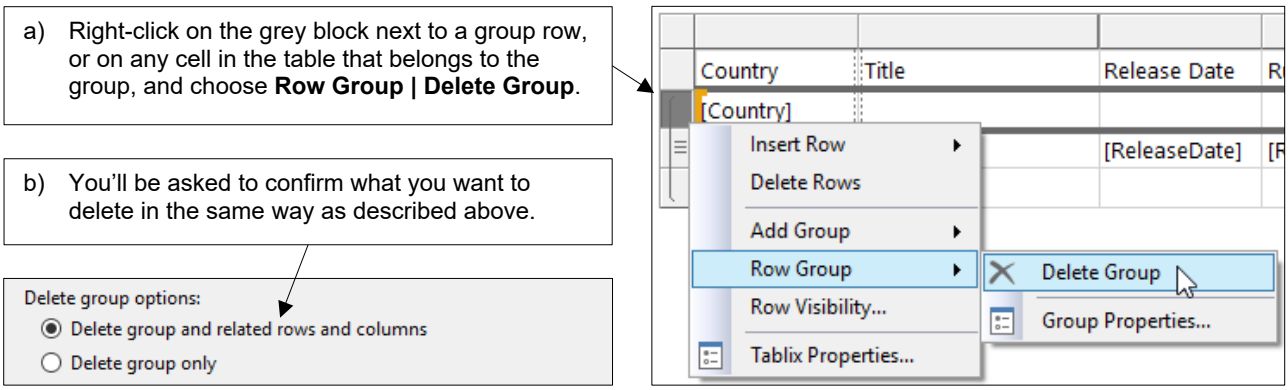
Deleting Groups in the Grouping Pane

The easiest way to delete a group is from the grouping pane.



Deleting Groups in a Table

You can also delete a group by right-clicking inside a table.



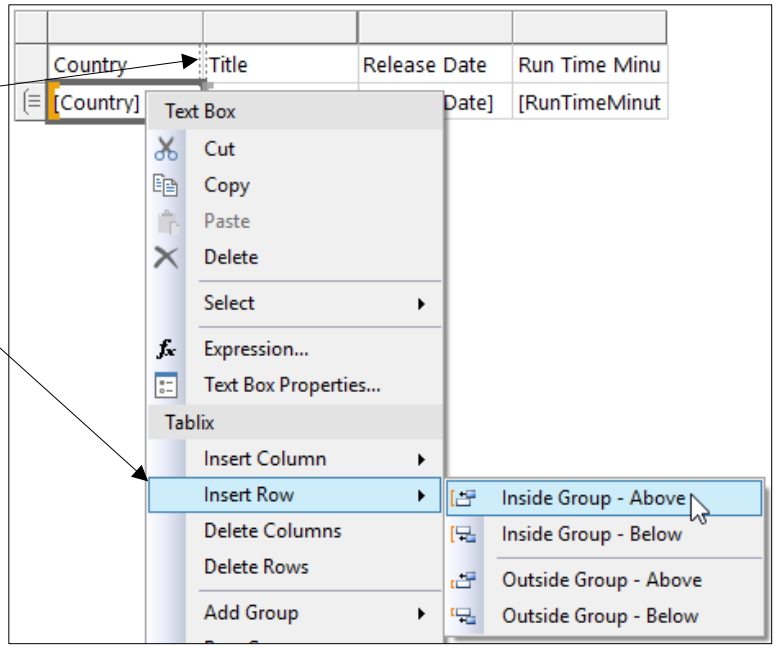
1.4 Formatting a Grouped Table

The default layout and formatting of a grouped table isn't particularly attractive. You can do several things to alter the default appearance, as explained in this section.

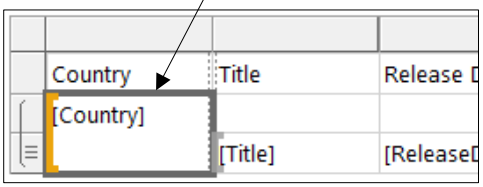
Adding Group Headers and Footers

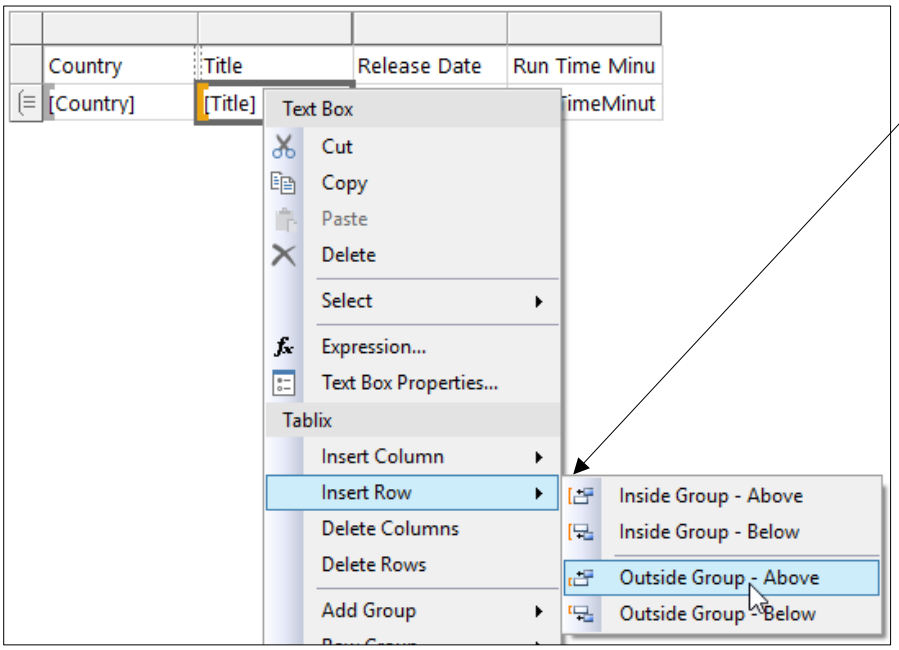
If you used the click and drag method to create a group, or you just forgot to tick the boxes on the dialog box when you created the group, you can add a header and footer to a group manually.

When you create a group a new column containing the group field will be added to the left of the table. This column is separated from the other columns with a double, dotted, vertical line.



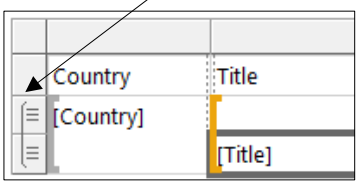
Right-click on a cell in the group to the left of the dotted vertical line and choose **Insert Row | Inside Group – Above** to create a group header. Choose **Below** to add group footer instead.





If you right-click to the right of the dotted line you must choose **Insert Row | Outside Group – Above** to create a group header.

If you had selected **Inside Group** in this case, your new row would appear within the **Details** group. This new row would appear for every row of data in the table, rather than once per group.

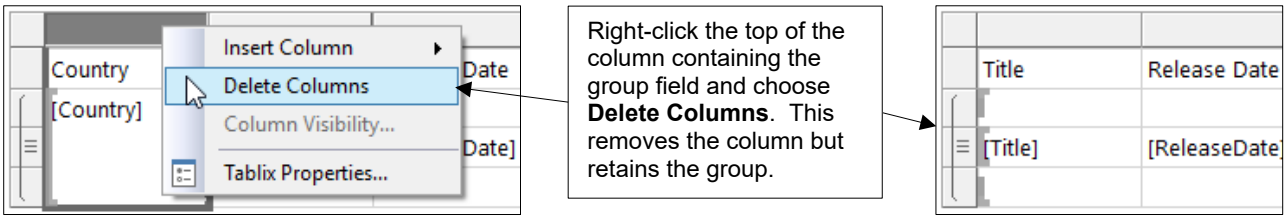


© Copyright 2024

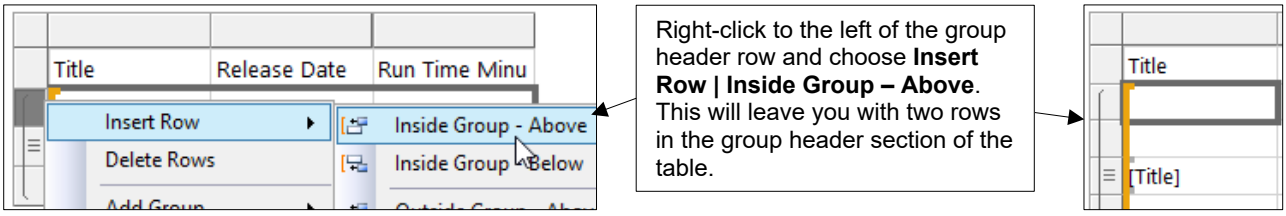
Rearranging Group Titles

By default the group title appears in a new column to the left of the grouped rows. You may find it looks better to move this title into a row which sits above the group. The steps to do this are:

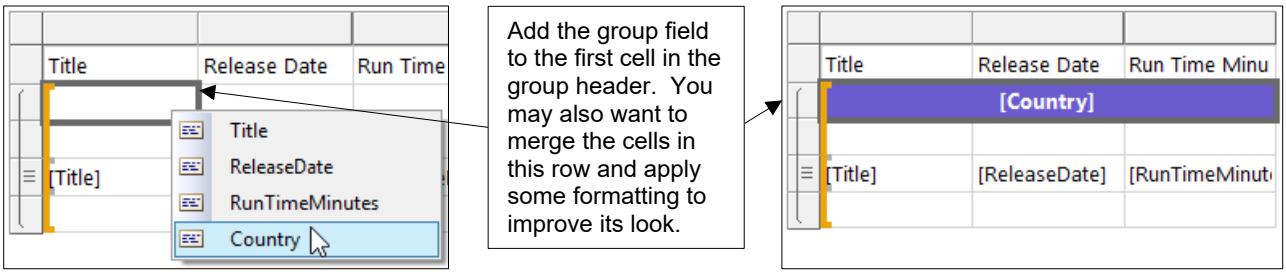
1) Delete the column which contains the group title:



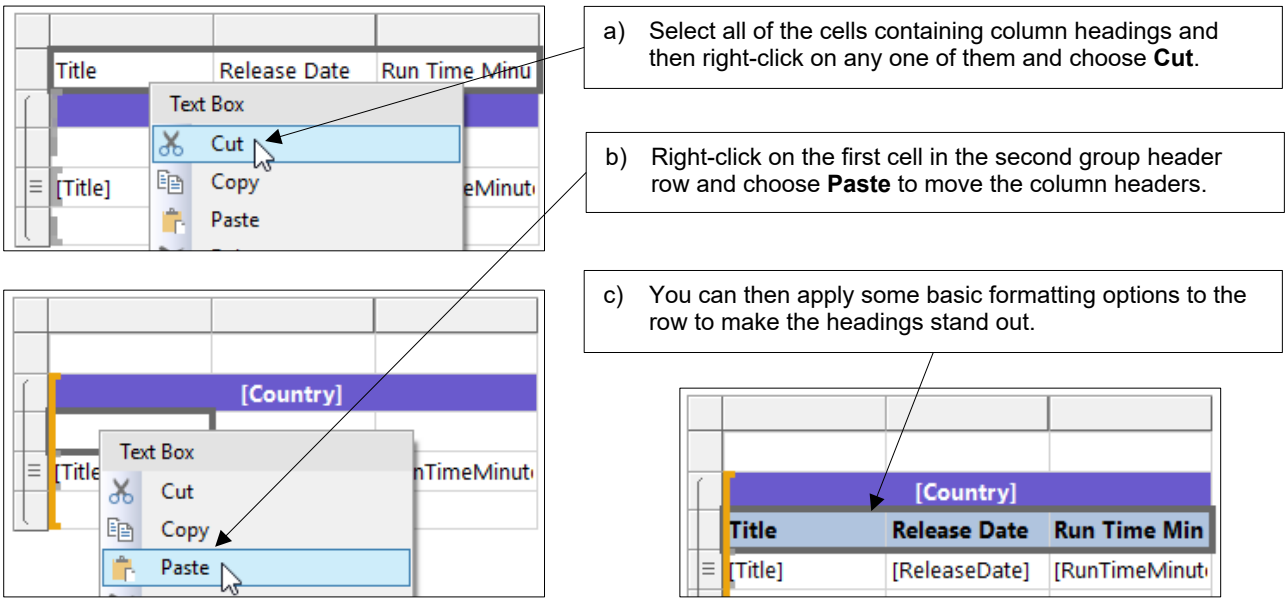
2) Insert an extra row for the group header:



3) Assign a field to the group header cell:



4) Move the column headings.

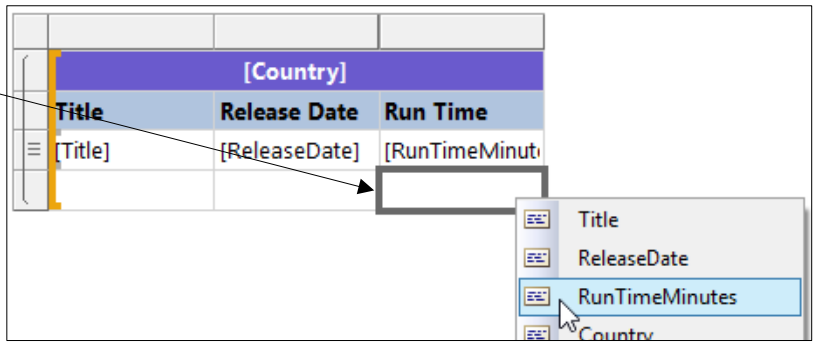


© Copyright 2024

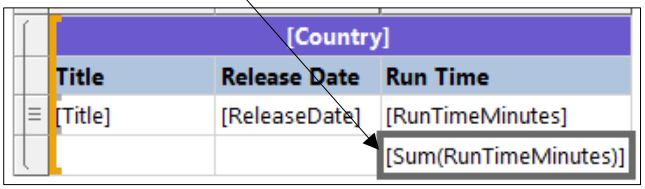
Adding Group Totals

You can add aggregated values to either a header or footer of a group to show totals, averages, etc:

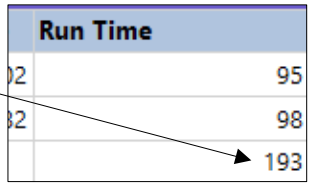
a) Assign a field to a cell in a header or footer row of a group. Here we're adding the **RunTimeMinutes** field to a cell in the footer row.



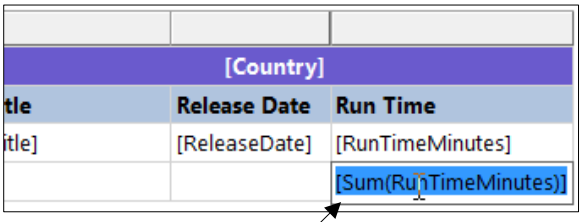
b) When you add a number field to a group header or footer it will create a sum of that field automatically.



c) The total will appear at the top or bottom of a group when you run the report.

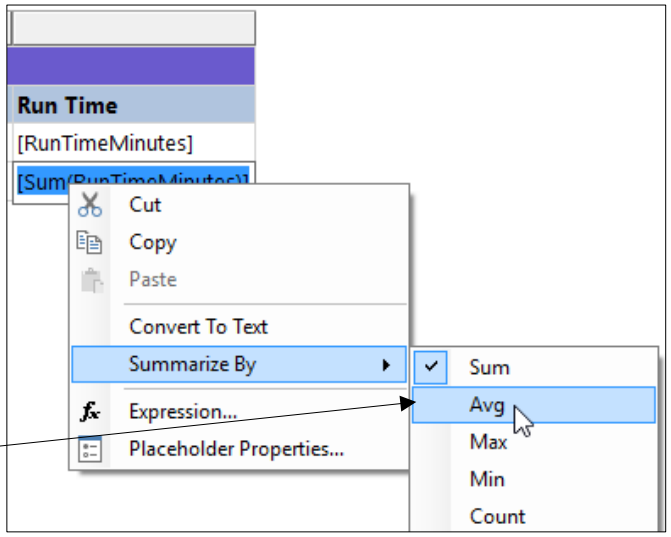


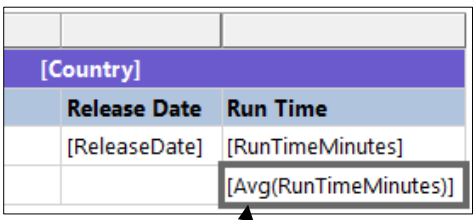
You can change the function applied to the field to create different aggregates.



a) Click on the text in the cell which contains the total expression – it will be highlighted in blue.

b) Right-click on the highlighted text and then select **Summarize By** followed by the function you want to apply to the field. Here we're choosing to show an average of the **RunTmieMinutes** field.





You may want to apply some formatting to make the results readable!

Title	Release Date	Run Time
Crocodile Dundee II	25/05/1988	112
Crocodile Dundee	26/09/1986	98
Happy Feet	17/11/2006	108
Mad Max	12/04/1979	93
Mad Max 2	24/12/1981	96
Mad Max Beyond Thunderdome	10/07/1985	107
Mad Max: Fury Road	07/05/2015	120
		104.857142857143

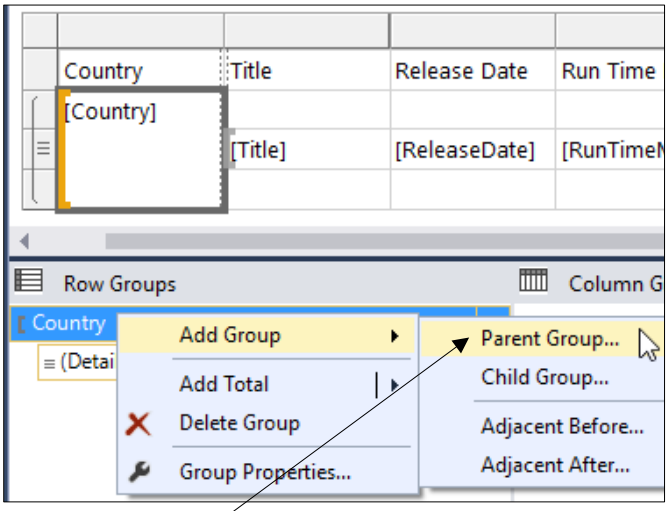
© Copyright 2024

1.5 Multiple Groups

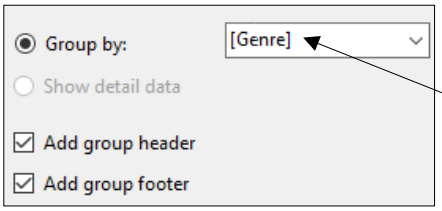
Once you've created one group, adding more is simple!

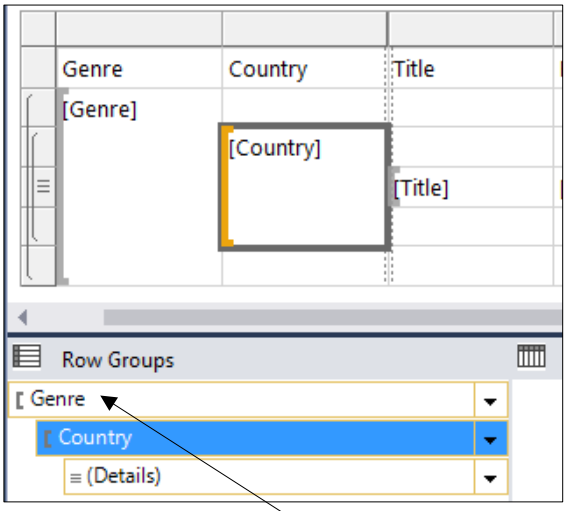
Adding a New Parent Group

You can add a parent group to an existing group in the same way you added the original group:



Right-click an existing group in the grouping pane and select **Add Group | Parent Group…** You'll need to complete the dialog box to say which field you want to group by.

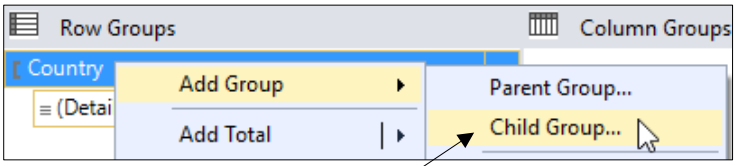




Here we're adding a group based on the **Genre** field. This field becomes a parent to the **Country** field, which is itself the parent to the **(Details)** item representing the individual films.

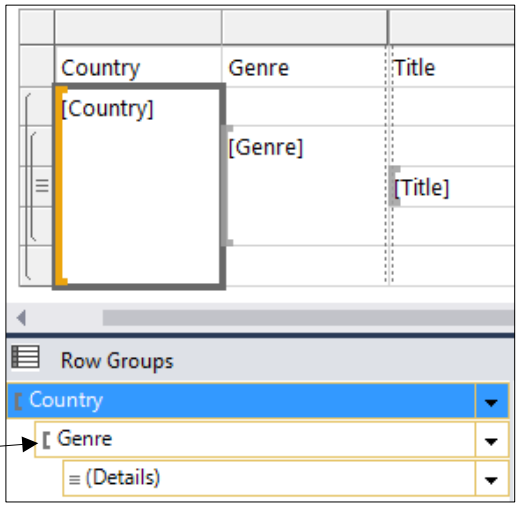
Adding a Child Group

You can add a *child group* to an existing group, as shown in the diagram below:



Right-click an existing group and choose **Add Group | Child Group…**. You'll then need to complete the dialog box in the same way as for a parent group.

The new group will be added as a subordinate of the original group. In this case **Country** becomes a parent of **Genre** which in turn becomes the parent of the **(Details)** item.

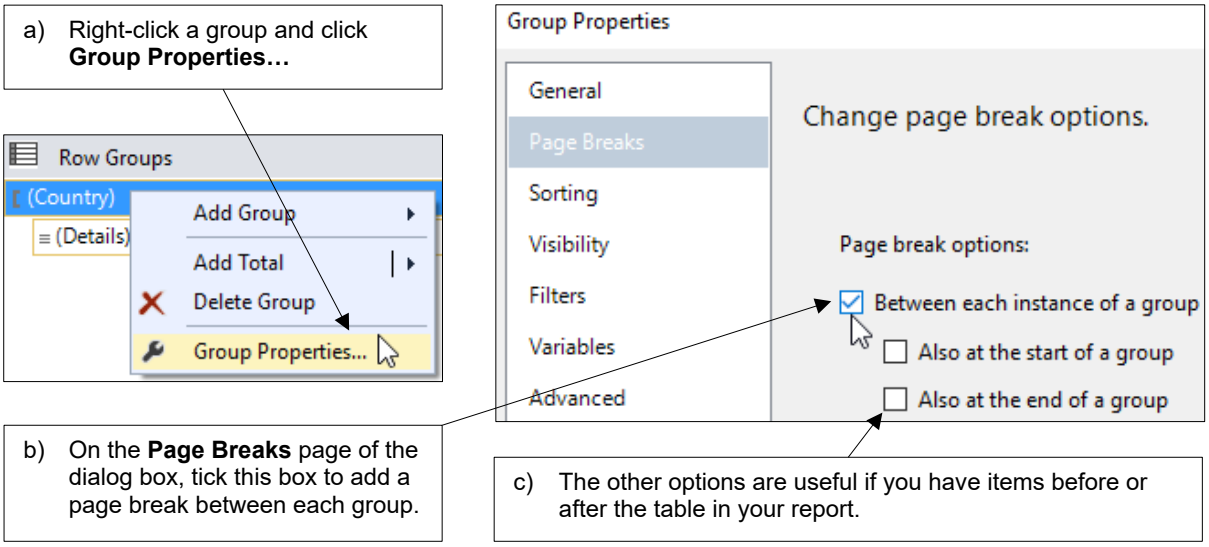


1.6 Page Breaks

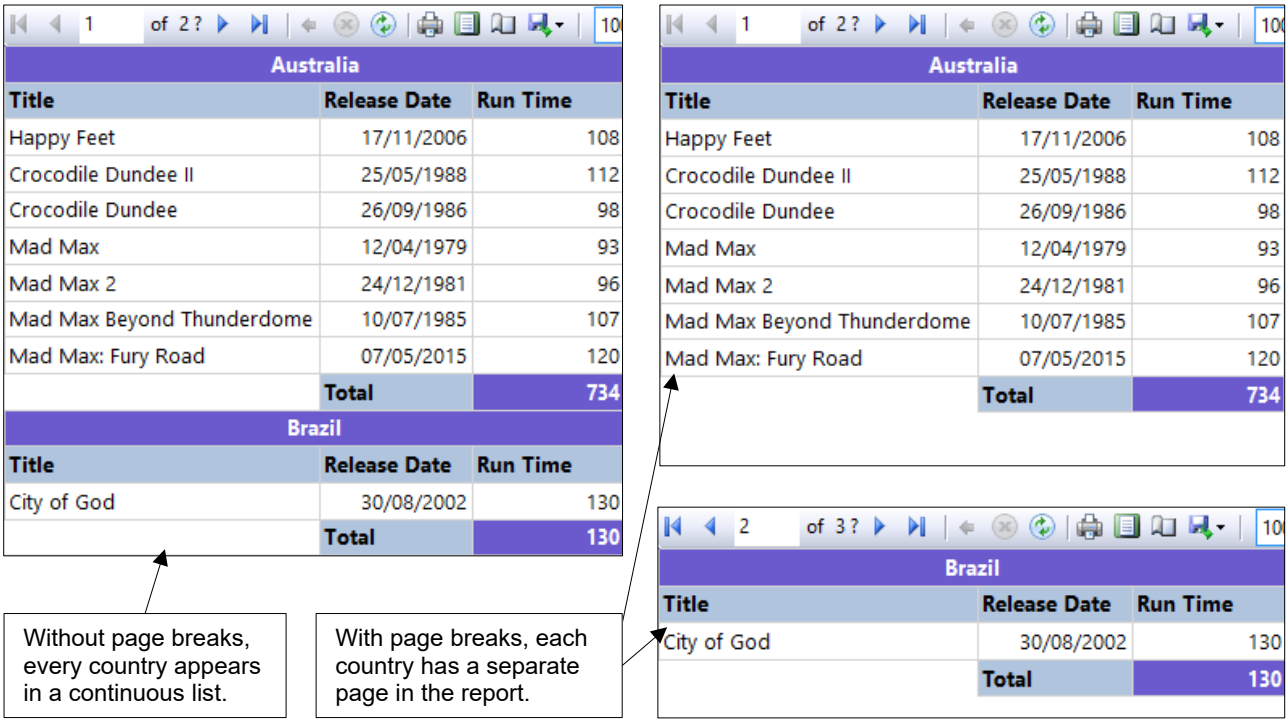
You may find it useful to have each group start on a new page, rather than simply following on immediately after the previous group in the table.

Inserting Page Breaks between Groups

You can add a page break between each group using the **Group Properties** dialog box.



You should notice a difference when you run the report:



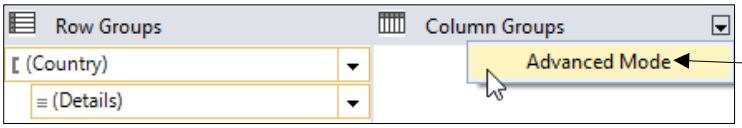
© Copyright 2024

1.7 Collapsible Groups

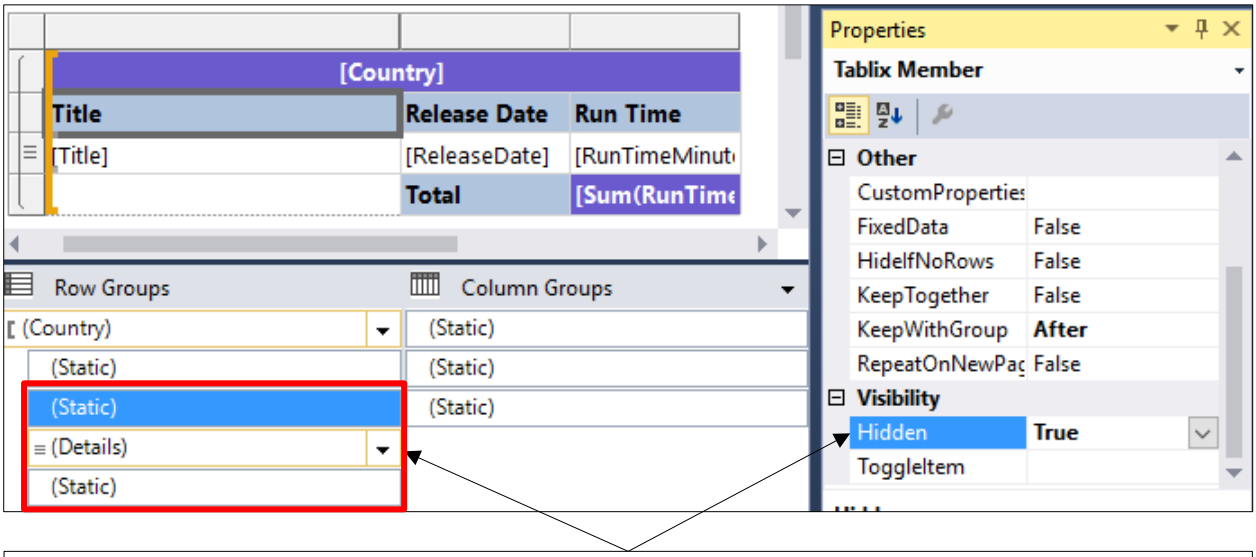
Creating collapsible groups allows you to show and hide the detail rows in a group when the report is running.

Step 1 - Hiding Detail Rows

The first step in creating a collapsible group is to ensure that the detail rows and any associated column headers and footers are hidden when the report loads.

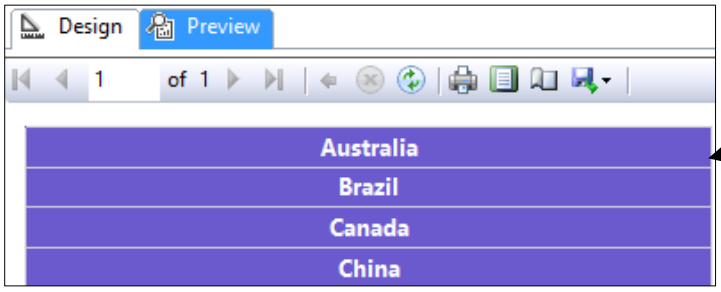


Make sure that the grouping pane is set to **Advanced Mode** by clicking the drop arrow in the top right and clicking here.



Select an item in the Grouping Pane that you want to hide and then change its **Hidden** property to **True** in the **Properties** window. For this example we would hide the **(Details)** item and the **(Static)** item immediately above and below it. You must do this separately for each item that you want to hide.

When you run the report you should only be able to see the items in the table that you didn't hide.



All of the detail rows and the column headers and group footer rows have been hidden.

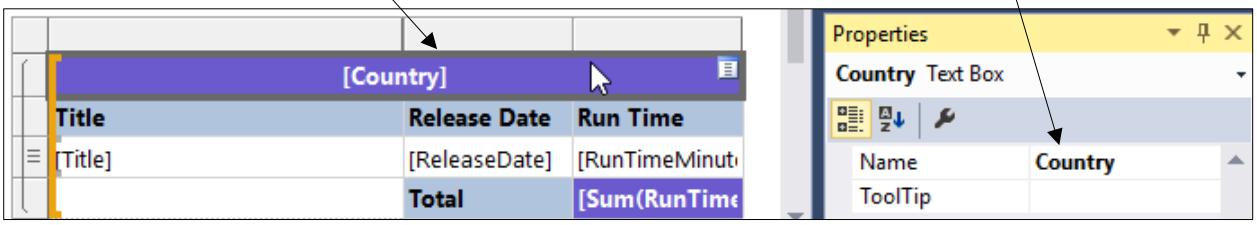
© Copyright 2024

Step 2 - Setting Toggle Items

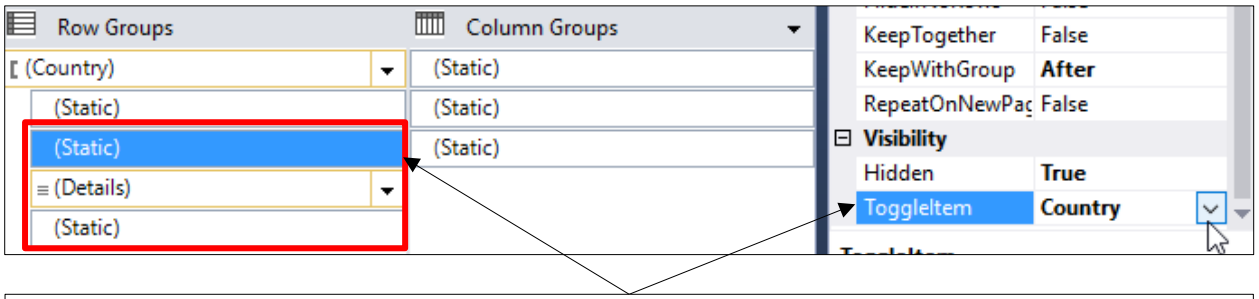
The *toggle item* is the report item that you will click on to show and hide the detail rows in each group. The first thing you need to do is find out the name of this item.

When we run the report we'd like to click on the cell containing the country name to show the details for that country. Start by selecting the cell which contains the **Country** field.

In the **Properties** window you'll see the name of the selected item at the top and also next to the **Name** property.



Having established the name of the item you'd like to click on, you can set this object to be a toggle item using the grouping pane and **Properties** window.

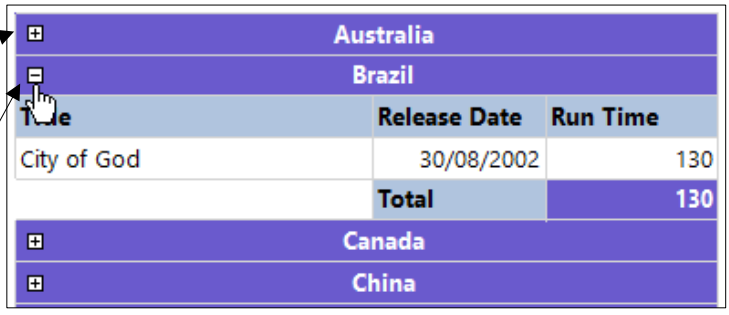


Select an item in the grouping pane whose visibility you want to control. You can then assign an item to toggle its visibility using the **ToggleItem** property in the **Properties** window. You need to do this separately for each item whose visibility you want to control. In our example we need to do this for three items.

When you run the report you should find that you can show and hide items in a group by clicking the + and – symbols which appear in the text box containing the country name.

Click a + symbol to expand a collapsed group.

Click a - symbol to collapse an expanded group.



© Copyright 2024

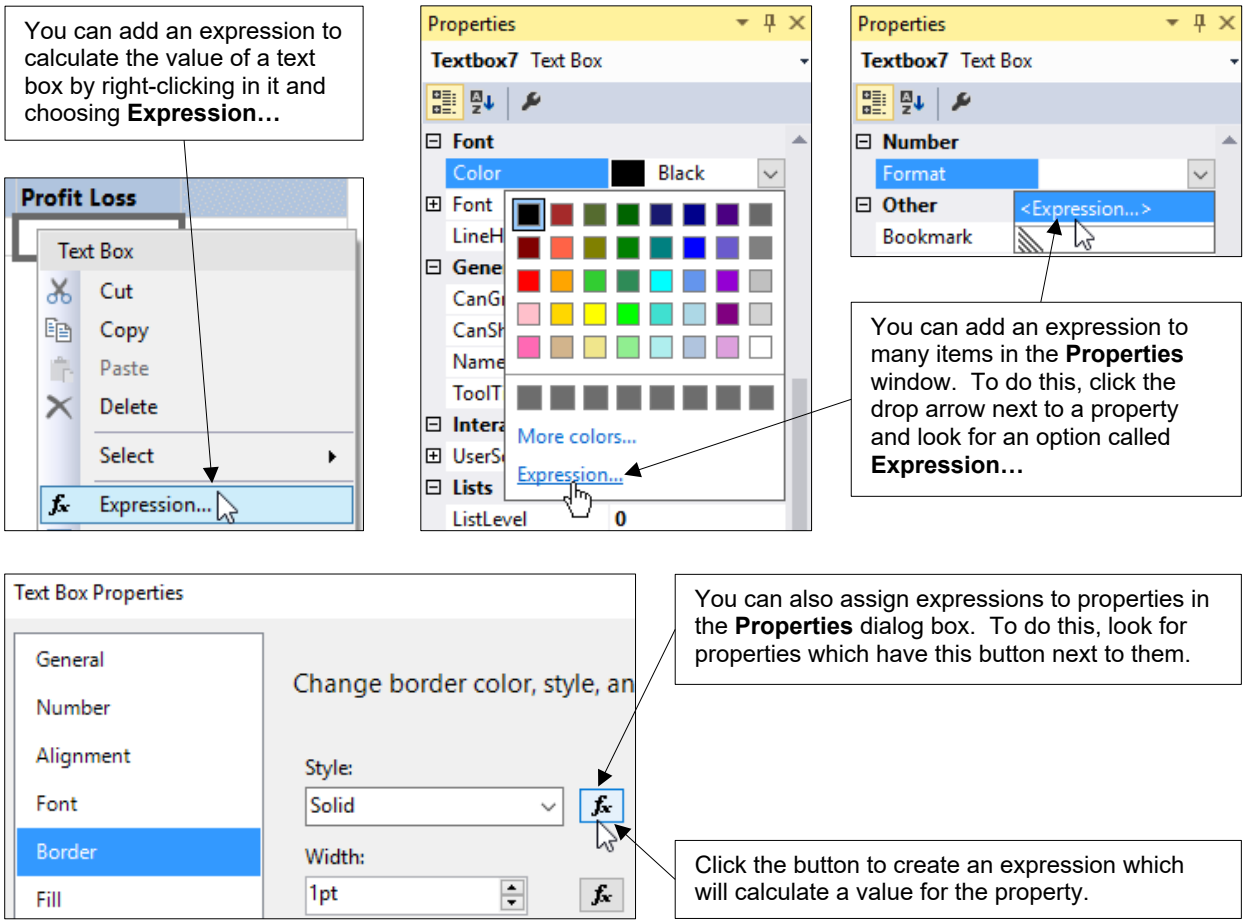
CHAPTER 2 - EXPRESSIONS

2.1 Overview of Expressions

Expressions are calculations that are evaluated when you run a report to determine some aspect of the report. They crop up almost everywhere in SSRS, as this chapter will show you!

Choosing to Create an Expression

You can add an expression to calculate a value in a variety of places in a report.

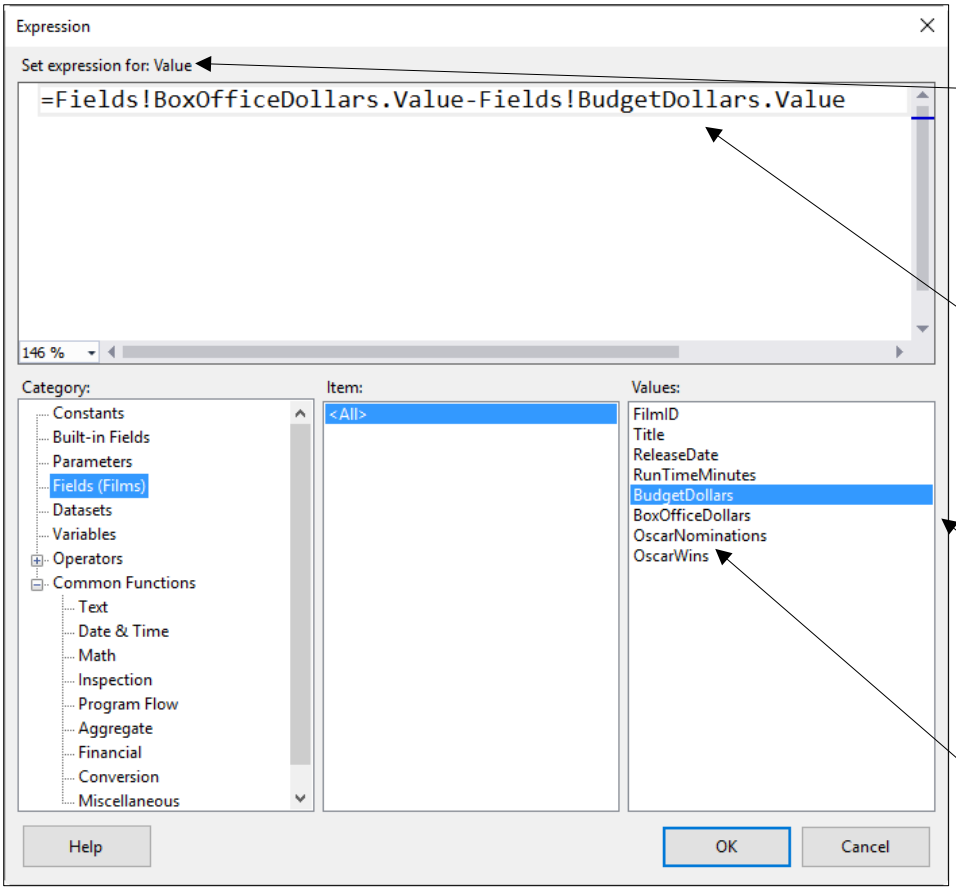


This chapter explains how to calculate new values which will appear in the report. The next chapter shows how to use expressions to change the appearance of a report.

© Copyright 2024

The Expression Builder Dialog Box

When you choose to create an expression you'll be presented with the *Expression Builder* dialog box. This provides you with help as you create your calculation.



This text provides you with a handy reminder of which property you're creating an expression for.

Use this area to create your expression using a combination of typing and selecting items from the panels in the bottom half of the dialog box.

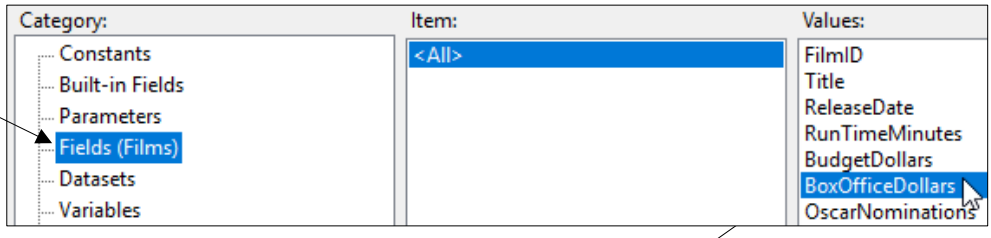
Use the options in the bottom half of the dialog box to insert items into your expression.

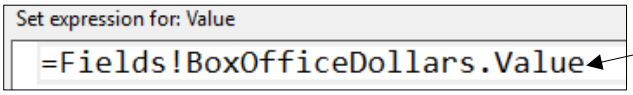
You can double-click on items in these panels to insert, for example, the names of functions and references to fields in a dataset.

Referring to Dataset Fields

Many of your expressions will refer to the value of fields in a dataset. You can insert references to fields easily using the **Expression Builder**.

In the bottom half of the dialog box, click the **Fields** category to see a list of fields in the dataset you're using.



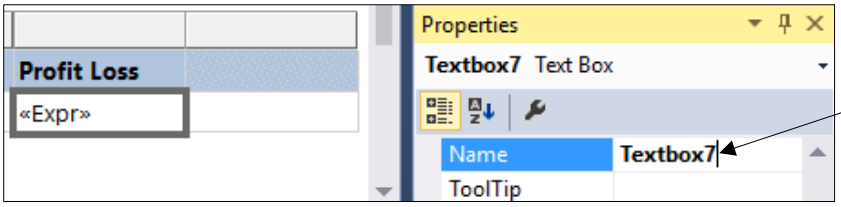


Double-click on a field to insert a reference into your expression. You can also simply type this in.

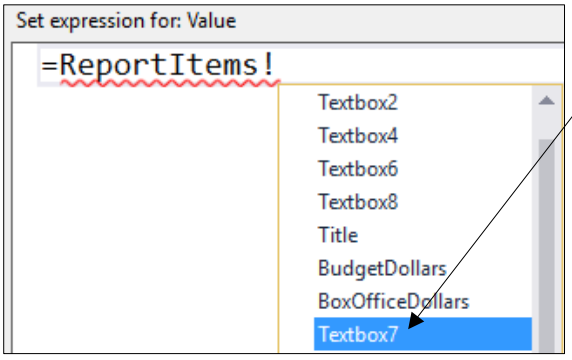
© Copyright 2024

Referring to Report Items

You'll sometimes need to refer to an item in the report, for example, to return the value of a textbox. The **Expression Builder** doesn't list report items so you'll need to do some typing.

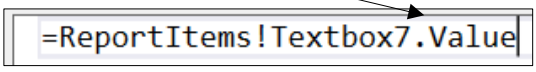


Each item in a report has a name which you'll need to know if you want to refer to it in an expression. You can select the item and look in the **Properties** window to see or change the item's name.



In the **Expression Builder**, type in **=ReportItems!** to see a list of the items in the report. Find the name of the item you want to reference and double-click or highlight it and press Tab to insert the name.

To complete the reference, enter a full stop and then refer to a property of the item. In this case it's **Value**.

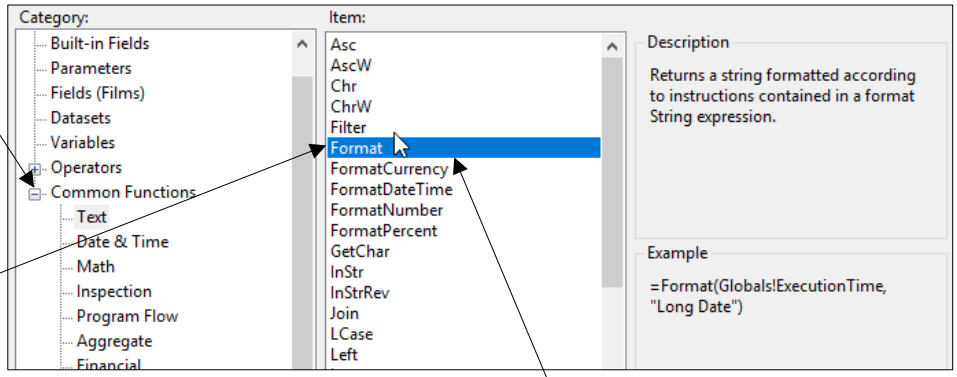


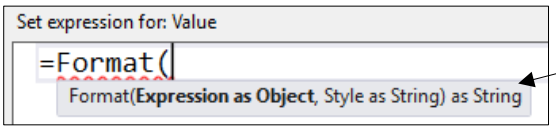
Inserting Functions

You can use a variety of functions in your expressions. Rather than trying to remember the names of all the available functions, you can use the **Expression Builder** to insert them.

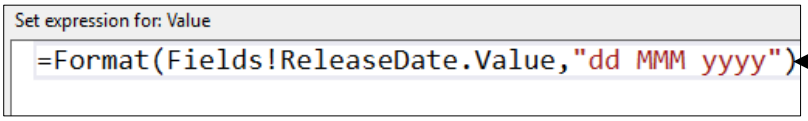
a) Expand **Common Functions** and click the category of functions you need.

b) Click a function name to see an example and description on the right of the box.





c) Double-click a function name to insert it into the expression. A tooltip appears to show you what arguments you must pass into the function's parameters.



d) Fill in the required arguments for the function and complete it by closing the parentheses.

© Copyright 2024

2.2 Ad-Hoc Table Columns

You can use *ad-hoc* expressions to create new calculated columns in a table. The table in the diagram below has two columns whose values are calculated using ad-hoc expressions.

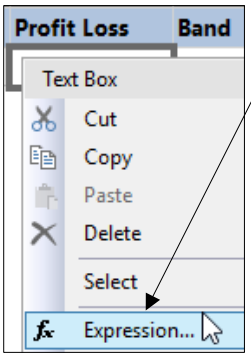
Title	Budget	Box Office	Profit Loss	Band
Jurassic Park	63000000	1029939903	966939903	Profit
Spider-Man	140000000	821708551	681708551	Profit
King Kong	207000000	550500000	343500000	Profit
Superman Returns	204000000	391081192	187081192	Profit
Titanic	200000000	2186772302	1986772302	Profit
Evan Almighty	175000000	173418781	-1581219	Loss

The **Profit Loss** column is calculated by subtracting the value of the **Budget** field from that of the **Box Office** field.

The **Band** column is calculated using an **IIf** function to determine if the value of the **Profit Loss** item is less than **0**.

Creating an Ad-Hoc Column Expression

You can calculate the value of a text box in a table by adding an expression directly to it.



Right-click the text box and choose **Expression…** to open the **Expression Builder**.

Set expression for: Value

```
=Fields!BoxOfficeDollars.Value-Fields!BudgetDollars.Value
```

Build the first expression by subtracting the value of the **BudgetDollars** field from that of the **BoxOfficeDollars** field. The final expression should look like this.

Once you've created the first expression you can create a new column which refers to it. You'll need to know the name of the text box which contains the first expression to do this.

Here we're using an **IIf** function (more on this later in the chapter) to check if the value of the text box containing the first expression has a value of less than **0**.

Set expression for: Value

```
=IIf(ReportItems!Textbox7.Value<0,"Loss","Profit")
```

If the condition is met, the function returns the word **Loss**, otherwise it returns the word **Profit**.

The examples on this page would be more useful as calculated fields, as described in the next section. Ad-hoc column expressions are still useful however, as some functions in SSRS can't be used in a calculated field.

© Copyright 2024

2.3 Calculated Fields

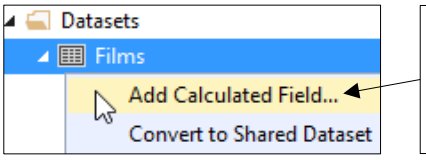
A *calculated field* is an expression which creates a new field in a dataset. You can use calculated fields in the same way as built-in fields to, for instance, sort and filter items in the dataset.

Calculated fields are generally more useful than ad-hoc calculated columns as they're easier to reference in subsequent expressions and you can reuse them in multiple report items.

Adding a Calculated Field to a Dataset

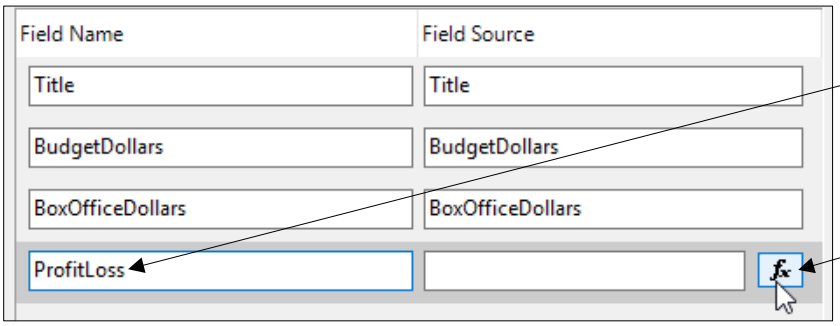
Adding a calculated field takes a little more effort than adding an ad-hoc column expression. Here are the steps you'll need to follow:

1) Choose to add a calculated field to a dataset.



Right-click on a dataset in the **Report Data** window and click **Add Calculated Field…**.

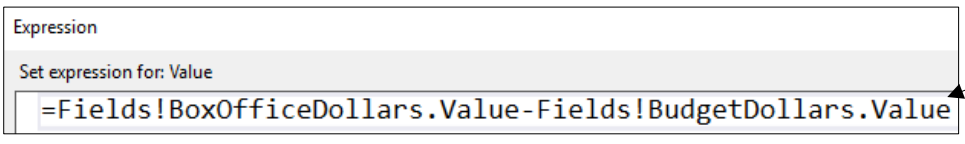
2) Provide a name for the field and launch the **Expression Builder**.



a) Type a name for the field without using spaces.

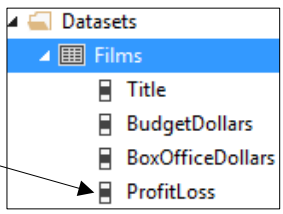
b) Click here to open the **Expression Builder** dialog box.

3) Create the expression.



Build an expression in the usual way.

When you've clicked OK a couple of times, your new field will appear in the dataset field list.

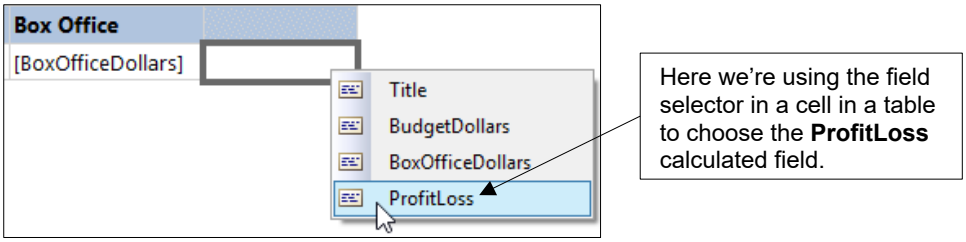


Calculated fields are listed in the dataset in the same way as other fields.

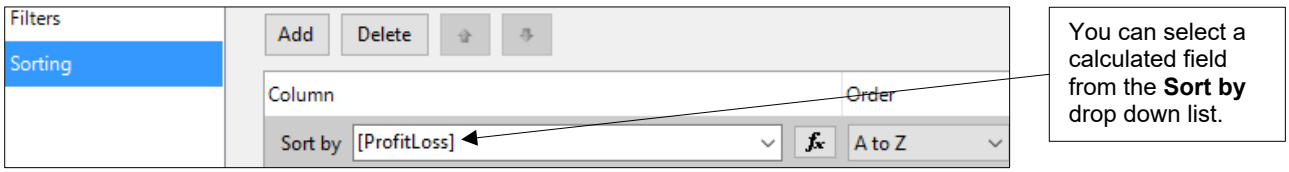
Using a Calculated Field

Once you've created a calculated field you can use it in the same way as any other field. For example, you can:

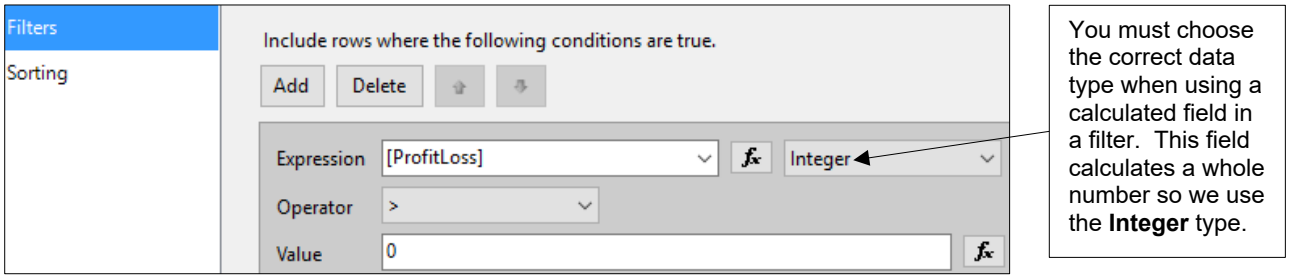
- Add the field to a report item such as a table.



- Apply sorting to a report item.

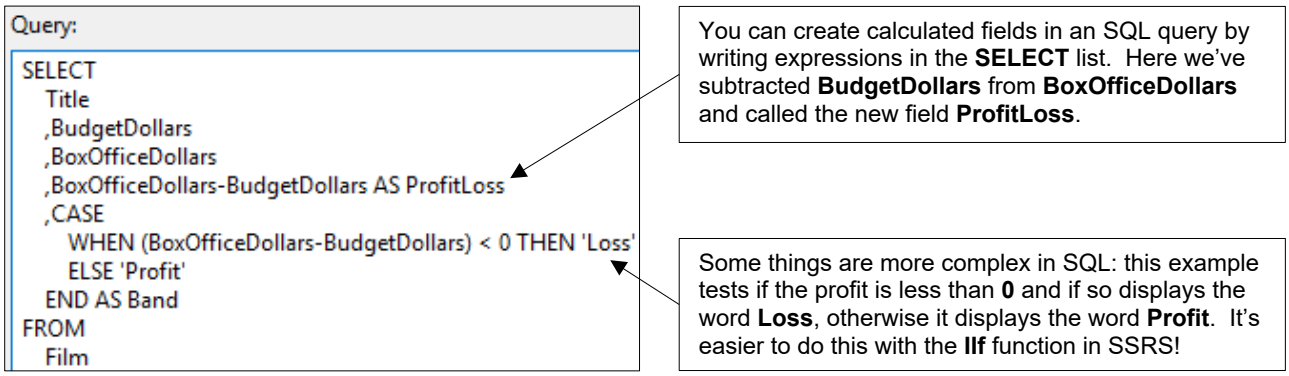


- Apply filtering to a dataset or report item.



Adding Calculated Fields to a Query

If you know how to write SQL you may prefer to add calculated columns to a dataset's query.



© Copyright 2024

2.4 Working with Numbers

Some of the simplest expressions you'll create will involve numbers. This page describes a few of the basic things you can do to manipulate numeric data in SSRS.

Basic Arithmetic

You can perform arithmetic on numbers by placing an *operator* between them. The table below lists the arithmetic operators in SSRS:

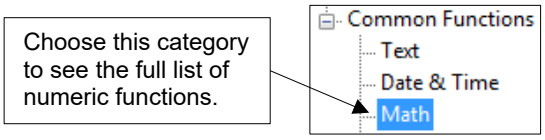
Operator	What it does	Example	Result
+	Adds two values together.	3 + 2	5
-	Subtracts one number from another	3 – 2	1
*	Multiplies two numbers together.	3 * 2	6
/	Divides one number by another and returns a decimal number.	3 / 2	1.5
\	Divides one number by another and returns a whole number.	3 \ 2	1
^	Raises one number to the power of another.	2 ^ 3	8
Mod	Returns the remainder of dividing one number by another.	3 Mod 2	1

Numeric Functions

You can use several numeric functions to manipulate numbers in interesting ways. You can see a summary of some useful numeric functions in the table below:

Function	What it returns	Example	Answer
`Abs( number )`	The absolute value of a number.	`Abs(-5)`	5
`Ceiling( number )`	The next largest whole number.	`Ceiling(3.147)`	4
`Floor( number )`	The next smallest whole number.	`Floor(3.147)`	3
`Int( number )`	The integer portion of a number. For negative numbers it returns the next lowest whole number.	`Int(9.81)` `Int(-9.81)`	9 -10
`Round( number, [number of decimal places])`	A number rounded to a whole number or an optional number of decimal places.	`Round(3.147)` `Round(3.147, 2)`	3 3.15
`Sqrt( number )`	The square root of a number.	`Sqrt(16)`	4

You'll find all of the functions shown above, along with many others, in the **Math** category of functions in the Expression Builder.



© Copyright 2024

2.5 Conditional Functions

Conditional functions are those which return a different answer depending on the result of evaluating a logical test.

The IIf Function

The *IIf* function allows you to test a condition and provide a different answer depending on whether or not it has been met. The basic syntax of the **IIf** function is:

```
=IIf(Condition_To_Test, Answer_If_True, Answer_If_False)
```

The slightly harsh example below tests whether a film has won any Oscars and, if so, describes it as a **Winner**, otherwise it calls the film a **Loser**.

Set expression for: Value

```
=IIf(Fields!OscarWins.Value>=1,"Winner","Loser")
```

The results of this expression are shown on the right. As we suspected: wearing your underpants outside your trousers does make you a loser.

Title	Oscars	Success
Jurassic Park	3	Winner
Spider-Man	0	Loser
King Kong	3	Winner
Superman Returns	0	Loser

The Switch Function

The *Switch* function is helpful for testing multiple conditions in the same expression. The basic syntax of the **Switch** function is:

```
=Switch(Condition1, Answer1[, Condition_n, Answer_n])
```

The example below categorises films based on their length in minutes.

Expressions with **Switch** tend to be quite long, so take advantage of the ability to use multiple lines in the Expression Builder.

```
=Switch(
    Fields!RunTimeMinutes.Value<100,"Short",
    Fields!RunTimeMinutes.Value<150,"Medium",
    Fields!RunTimeMinutes.Value<200,"Long",
    TRUE,"Epic")
```

Conditions are evaluated in the order you write them so, in this case, if a film falls into the **Short** category it can't belong to any of the other categories.

If a value fails all of the previous tests it must implicitly have a value greater than or equal to **200**. We could explicitly test for this but it's easier to enter the value **TRUE** (which would be the result of the test anyway).

© Copyright 2024

Comparison Operators

A *comparison operator* is the symbol that you place between the two values you're testing. The table below describes the comparison operators you can use in SSRS.

Operator	Description
=	Is equal to
>	Is greater than
>=	Is greater than or equal to

Operator	Description
<>	Is not equal to
<	Is less than
<=	Is less than or equal to

Logical Operators

A *logical operator* affects the way your conditional tests are evaluated. There are three main logical operators you're likely to encounter in SSRS:

Operator	Description	Example

AND	Combines two logical tests and returns **True** if both conditions are met.	`=IIf(`
		`  Fields!Profit.Value>0 AND Fields!OscarWins.Value>0,`
		`  "Success", "Failure")`

OR	Combines two logical tests and returns **True** if either condition is met.	`=IIf(`
		`  Fields!Profit.Value<0 OR Fields!OscarWins.Value=0,`
		`  "Failure", "Success")`

NOT	Returns the opposite of a logical value.	`=IIf(`
		`  NOT(Fields!Profit.Value>0 OR`
		`  Fields!OscarWins.Value>0),`
		`  "Disaster","Success")`

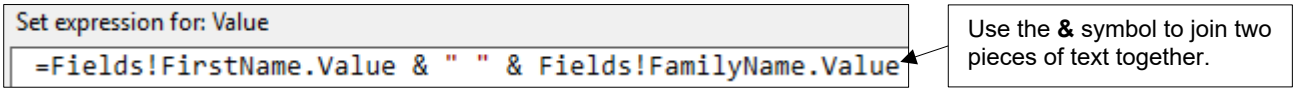
© Copyright 2024

2.6 Working with Text

You can manipulate text, or *strings*, in a variety of ways in SSRS. Bear in mind that most string operations in SSRS are case-sensitive.

Concatenating Text

Joining strings together, or *concatenating* them, is one of the more common operations you'll perform with strings.



Text Functions

The table below describes some of the useful text functions you'll find in SSRS:

Function	What it returns	Example	Answer
`InStr( string1, string2, [compare] )`	The position of the specified character in the string.	`=InStr( "Wise Owl", "w")`	7
`LCase( string )`	The lower case version of the string.	`=LCase("Wise Owl")`	wise owl
`Left( string, length )`	The specified number of characters from the left of the string.	`=Left("Wise Owl", 4)`	Wise
`Len( string )`	The number of characters in the string.	`=Len("Wise Owl")`	8
`LTrim( string )`	The string without any leading spaces.	`=LTrim(" Wise Owl")`	Wise Owl
`Mid( string, start, [length] )`	The specified number of characters from a given starting position in the string.	`=Mid( "Wise Owl", 6, 3)`	Owl
`Replace( string, find, replace, [start], [count], [compare] )`	The string with the specified characters replaced by a new string.	`=Replace( "Wise Owl", "w", "O", 1, -1, vbTextCompare )`	Oise OOl
`Right( string, length )`	The specified number of characters from the right of the string.	`=Right("Wise Owl", 3)`	Owl
`RTrim( string )`	The string without any trailing spaces.	`=RTrim("Wise Owl ")`	Wise Owl
`Trim( string )`	A string without leading or trailing spaces.	`=Trim(" Wise Owl ")`	Wise Owl
`UCase( string )`	The upper case version of the string.	`=UCase("Wise Owl")`	WISE OWL

*For functions with a **Compare** parameter, you can use **vbBinaryCompare** for case-sensitive searches and **vbTextCompare** to make the search case-insensitive.*

© Copyright 2024

2.7 Working with Dates

You can manipulate dates in a variety of ways in SSRS.

Returning the Current Date

Many date expressions rely on knowing what the current date is. The table below describes a few ways to get the current date.

Expression	What it does
`=Today()`	Returns the current date which is recalculated each time a new page is loaded.
`=Now()`	Gives the current date and time. This recalculates each time a new page loads.
`=Globals!ExecutionTime`	Returns the current date and time and is calculated when the report first loads.

Date Functions

You can use a variety of functions to work with dates in your report. The examples in the table below assume that today's date is Monday 19th June 2017.

Function	What it returns	Example	Answer
`DateAdd( interval, number, date )`	A date to which a specified time interval has been added.	`=DateAdd("d", 30, Today())`	6/19/2017
`DateDiff( interval, date1, date2)`	The number of specified intervals between two dates.	`=DateDiff("d", Today(), #12/25/2017#)`	189
`DateSerial( year, month, day )`	A complete date from the three separate values.	`=DateSerial( 2017, 7, 16)`	7/16/2017
`DateValue( date )`	A date value based on the provided date, which is usually a string.	`=DateValue( "18 July 2017")`	7/18/2017
`Day( date )`	The day of the month of the date.	`=Day(Today())`	19
`Month( date )`	The month number of the date.	`=Month(Today())`	7
`MonthName( number )`	The name of the numbered month.	`=MonthName(6)`	June
`Weekday( date, [firstdayofweek] )`	The number of the day of the week.	`=Weekday(Today(), vbSunday)`	2
`WeekdayName( number, [abbreviate], [firstdayofweek] )`	The name of the numbered weekday.	`=WeekdayName(2, False, vbSunday)`	Monday
`Year( date )`	The year of the provided date.	`=Year(Today())`	2017

© Copyright 2024

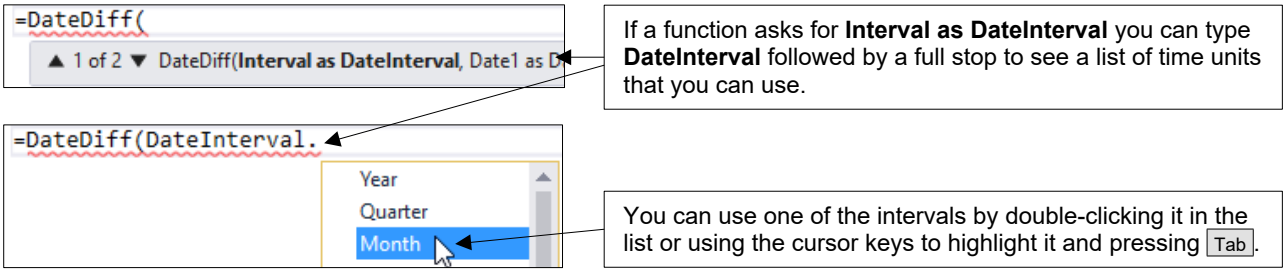
Intervals for Date Functions

Several of the date functions have an **Interval** parameter. The table below lists the string values you can pass as arguments to this parameter:

Interval	Description
"yyyy"	Year
"q"	Quarter
"m"	Month
"y"	Day of year
"d"	Day of month

Interval	Description
"w"	Weekday
"ww"	Week
"h"	Hour
"n"	Minute
"s"	Second

Rather than passing a string to the **Interval** parameter, you can use one of the **DateInterval** constants.

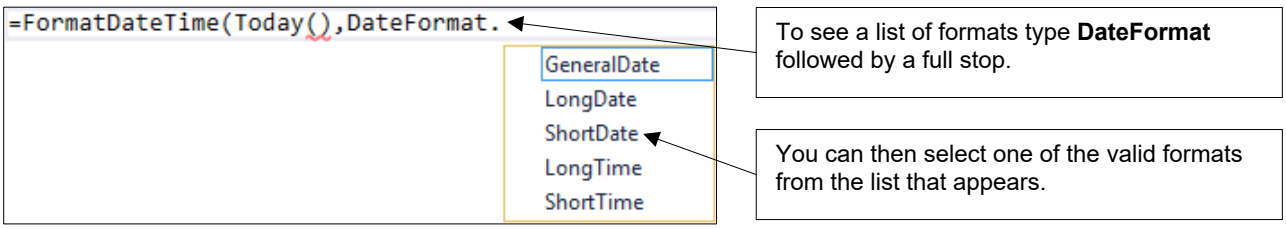


Formatting Dates

You can use functions to calculate a formatted version of your date values. There are two functions that can help you with this, as shown in the table below:

Function	Example	Result
Format(*value*, *[format]*, *[firstdayofweek]*, *[firstweekofyear]*)	Format(Today(), "dddd, d MMMM yyyy")	Monday, 19 June 2017
FormatDateTime(*date*, *[namedformat]*)	FormatDateTime(Today(), DateFormat.LongDate)	19 June 2017

There are several named formats that you can use with the **FormatDateTime** function:



© Copyright 2024

2.8 Aggregating Data

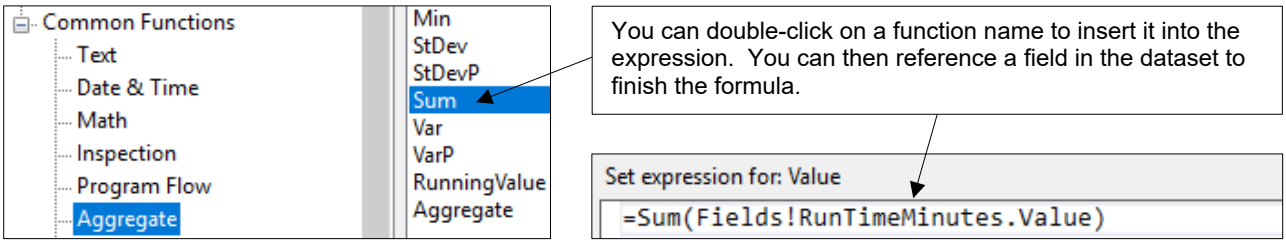
An *aggregate* function is one which usually operates on several rows of data rather than just one. You'll commonly find these functions used in table group footers, matrices and charts.

Aggregate Functions

You can see a list of some common aggregate functions in the table below:

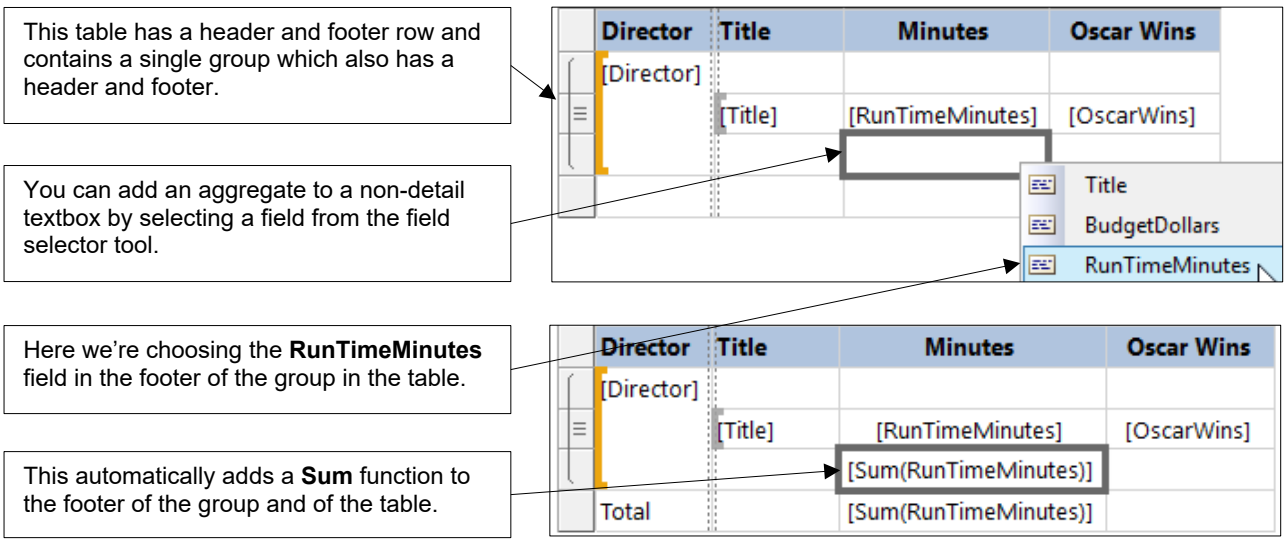
Function	What it returns	Example
`Sum( Field )`	The total of a numeric field.	`=Sum(Fields!OscarWins.Value)`
`Avg( Field )`	The average of a numeric field.	`=Avg(Fields!RunTimeMinutes.Value)`
`Min( Field )`	The lowest value of a field.	`=Min(Fields!BudgetDollars.Value)`
`Max( Field )`	The highest value of a field.	`=Max(Fields!BoxOfficeDollars.Value)`
`Count( Field )`	The number of non-null values in a field.	`=Count(Fields!FilmID.Value)`

You'll find these functions, along with several others, in the **Aggregate** category of the **Common Functions** list in the **Expression Builder**.



Adding Aggregates the Quick Way

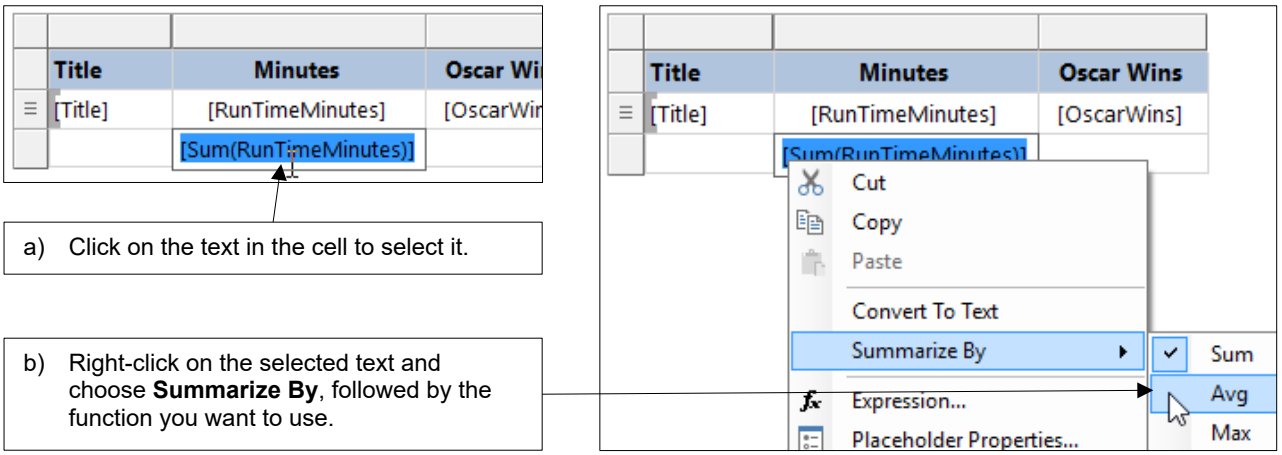
You can quickly add an aggregate function to a tablix report item by adding a field to cell which doesn't belong to a **Detail** row.



© Copyright 2024

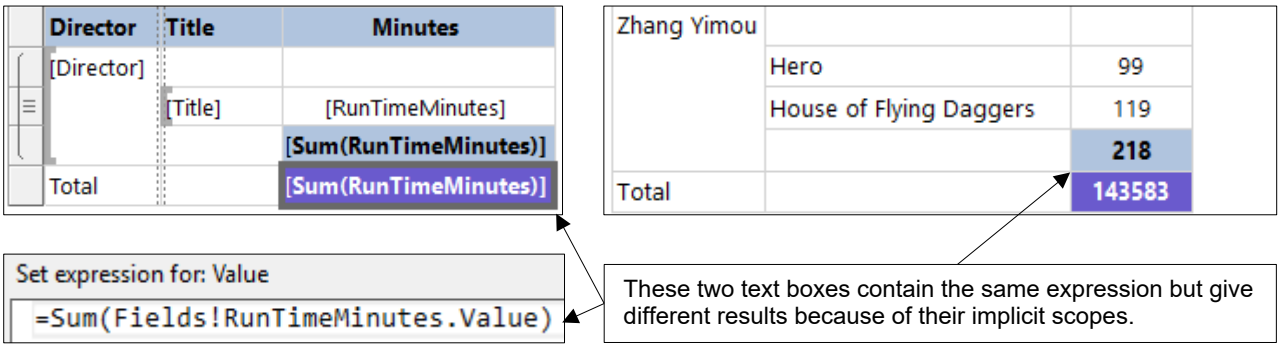
Changing the Aggregate Function

When you add an aggregate using the quick technique shown on the previous page, SSRS automatically applies the **Sum** function if you choose a numeric field. You can easily change this:

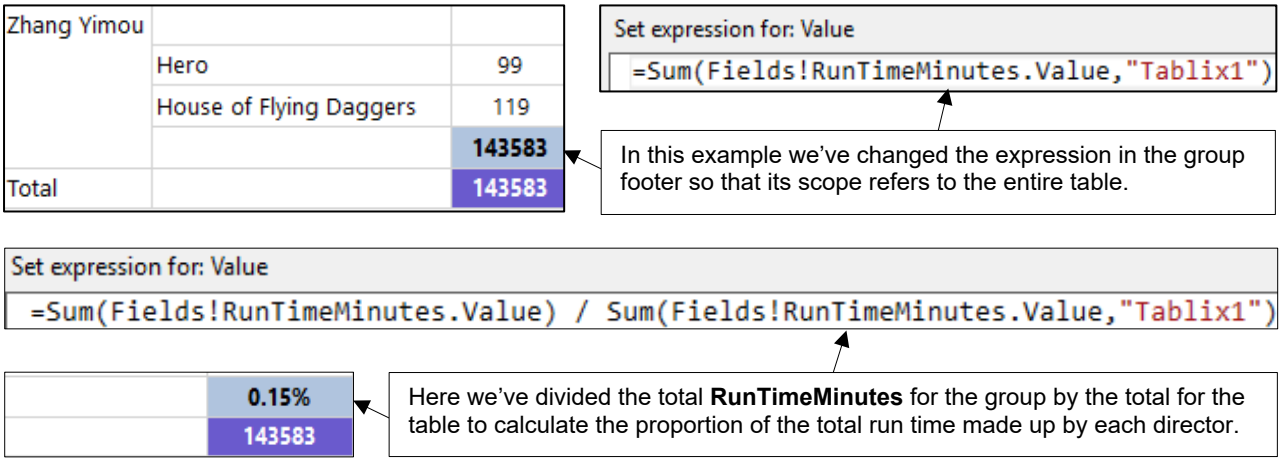


Specifying the Scope

The range of rows that an aggregate function operates on is referred to as the *scope*. The scope is applied automatically based on where the aggregate function is placed.



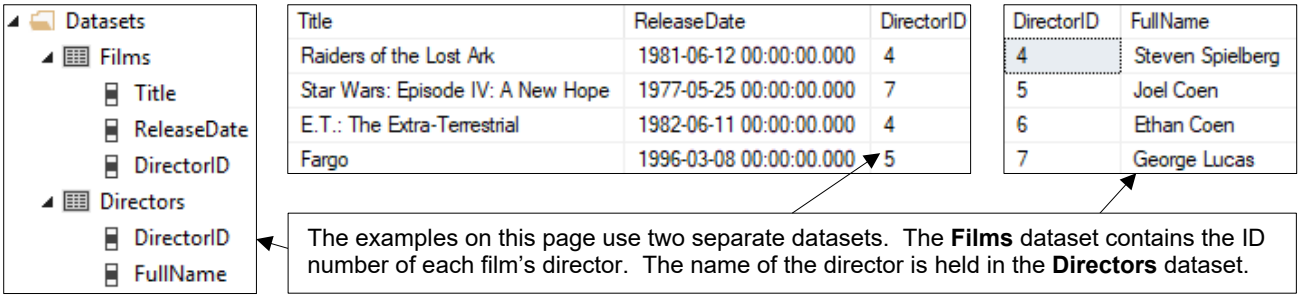
You can alter the scope by passing the name of a data region to an aggregate function. A data region name can be the name of a dataset, tablix report item or group.



© Copyright 2024

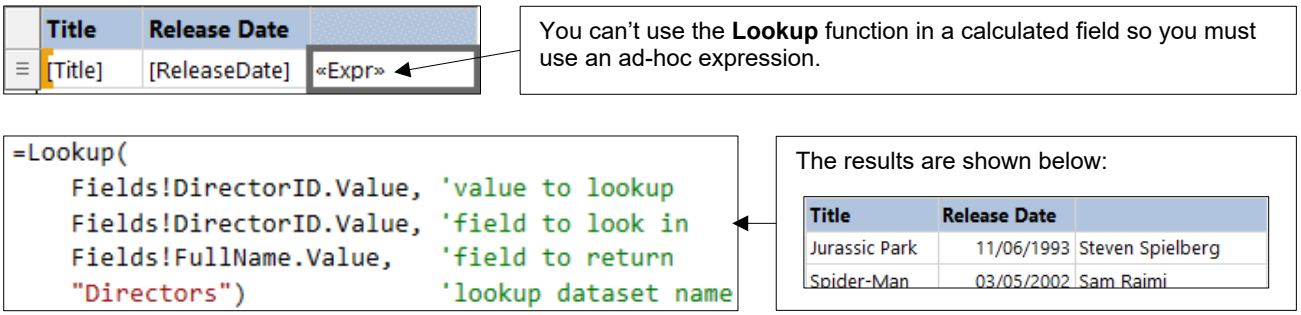
2.9 Lookup Functions

SSRS has two *lookup* functions which allow you to match rows in different datasets. This is useful when the datasets use different data sources and couldn't be joined in a single dataset.



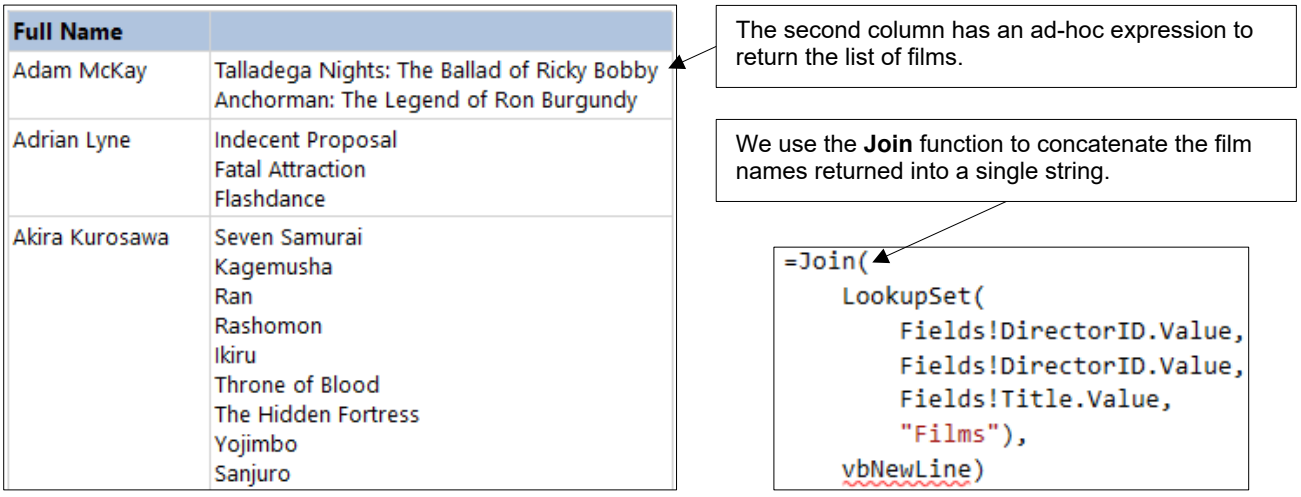
The Lookup Function

You can use the **Lookup** function to return a single match for the item you're looking up. In the example below we're attempting to return the name of the single director for each film.



The LookupSet Function

The **LookupSet** function is very similar to the **Lookup** function except that it returns multiple values. In the example below we're returning all of the films made by each director.



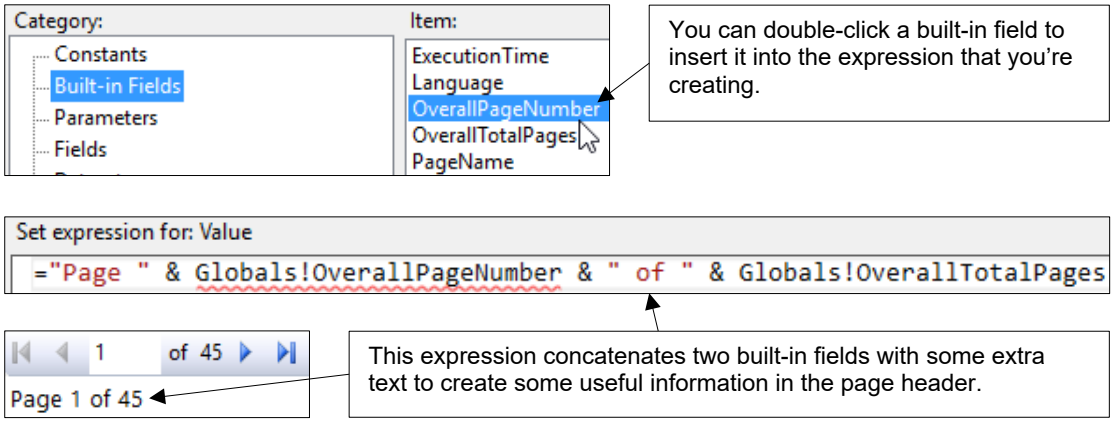
© Copyright 2024

2.10 Built-In Fields

Reporting Services provides you with a set of global values referred to as *built-in* fields. You can use these to create expressions in your reports.

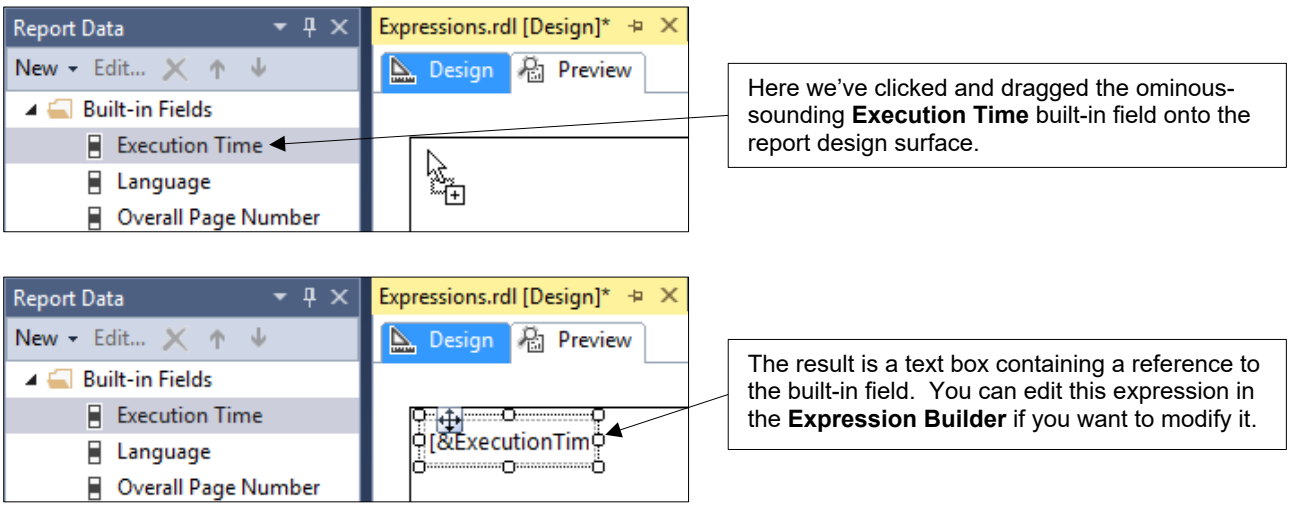
Built-In Fields in the Expression Builder

You can find a category containing the built-in fields in the **Expression Builder**.



Built-In Fields in the Report Data Window

You can drag built-in fields from the **Report Data** window in much the same way as dragging dataset fields.



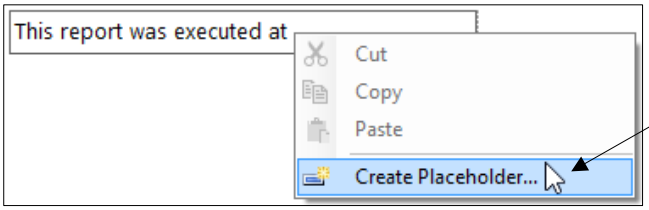
2.11 Placeholders

As an alternative to concatenating long, complex expressions, you can use a *placeholder* to make life easier.

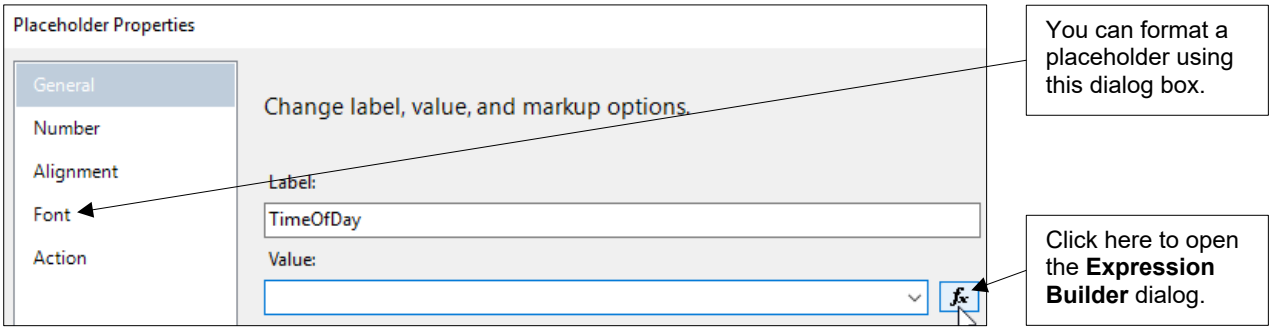
Creating a Placeholder

You can insert a placeholder into a textbox alongside any other text that you want to display. To do this:

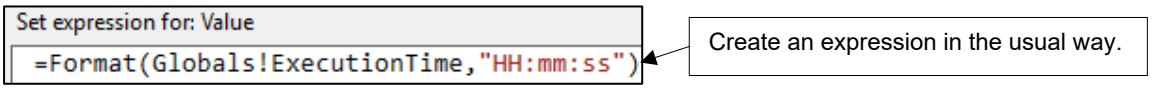
1) Right-click inside the text box at the position you'd like to create your placeholder.



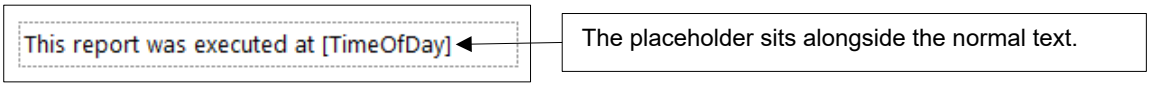
2) Enter a label which describes the placeholder and then use the button shown below to open the **Expression Builder**.



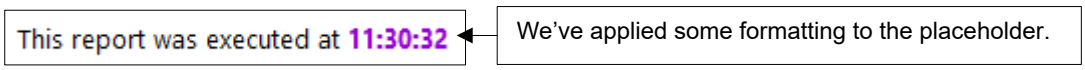
3) Create an expression to calculate the value of the placeholder.



4) Click OK twice to insert the placeholder into the text box.



When your report runs, the placeholder will be calculated and display the result.



© Copyright 2024

What we do!

		Basic training	Advanced training	Systems / consultancy
Office	Microsoft Excel			
	VBA macros			
	Office Scripts			
	Microsoft Access			
Power BI, etc	Power BI and DAX			
	Power Apps			
	Power Automate (both)			
SQL Server	SQL			
	Reporting Services			
	Report Builder			
	Integration Services			
	Analysis Services			
Coding and AI	Visual C#			
	VB programming			
	AI tools			
	Python			

Wise Owl Training

Wise Owl
Training