**MAIN SSRS NOTES and EXPRESSIONS**

**Using Built-in Fields**

Display  Report Execution Time in a textbox:

="Report Execution Time: " & Globals!ExecutionTime

Display Page No in a textbox:  
="Page " & Globals!PageNumber & " of " & Globals!TotalPages

**Date & Time Functions**

The **Today()** function provides the current date. The expression  =Today() can be used in a text box to display the date on the report, or in a parameter to filter data based on the current date. This function will return date in M/d/yyyy 12:00:00 AM format. You can use **Format** function to display required format. Some of the popular date formats are given below:

|  |  |
| --- | --- |
| **Expression** | **Output** |
| =FORMAT(Today(),"M/d/yy") | 8/23/10 |
| =FORMAT(Today(),"MM-dd-yyyy") | 08-23-2010 |
| =FORMAT(Today(),"MMM-dd-yyyy") | Aug-23-2010 |
| =FORMAT(Today(),"MMMM dd, yyyy") | August 23, 2010 |
| =FORMAT(DateField,"MMM dd, yyyy hh:mm:ss") | Aug 23, 2010 01:43:33 |
| =FORMAT(DateField,"MMM dd, yyyy HH:mm:ss") | Aug 23, 2010 13:43:33 |
| =FORMAT(DateField,"MMM dd, yyyy HH:mm:ss.fff") | Aug 23, 2010 13:43:33.587 |
| =FORMAT(DateField,"MMM dd, yyyy hh:mm:ss tt") | Aug 23, 2010 01:43:33 PM |

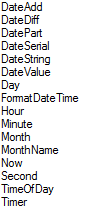
Note: FormatDateTime function can also be used to format the date field e.g. =FormatDateTime(Fields!BirthDate.Value, DateFormat.ShortDate)

**DateAdd** - Returns a Date value containing a date and time value to which a specified time interval has been added. this function can be used in an expression to add/substract time(day, month, year, sec etc.) from given date field:   
=DateAdd(DateInterval.Month, 6, Parameters!StartDate.Value)

**DateDiff** - Returns a Long value specifying the number of time intervals between two Date values.   
=DateDiff("yyyy",Fields!BirthDate.Value,Today())

**DatePart** - Returns an Integer value containing the specified component of a given Date value.   
=DatePart("q",Fields!BirthDate.Value,0,0)  
  
=DatePart(DateInterval.Quarter,Fields!BirthDate.Value, FirstDayOfWeek.System, FirstWeekOfYear.System)

There are many other Date &Time functions which can be used expression:

[](http://3.bp.blogspot.com/_2t15yG-FxQA/THLkyjKEDuI/AAAAAAAAA44/nicEZ332Pvc/s1600/1.png)

**String Functions**

• Combine more than one field by using concatenation operators and Visual Basic constants. The following expression returns two fields, each on a separate line in the same text box:

=Fields!FirstName.Value & vbCrLf & Fields!LastName.Value

•Format dates and numbers in a string with the Format function.

=Format(Parameters!StartDate.Value, "M/D") & " through " & Format(Parameters!EndDate.Value, "M/D")

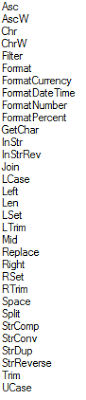
•The **Right**, **Len**, and **InStr** functions are useful for returning a substring, for example, trimming DOMAIN\username to just the user name. The following expression returns the part of the string to the right of a backslash (\) character from a parameter named User:

=Right(Parameters!User.Value, Len(Parameters!User.Value) - InStr(Parameters!User.Value, "\"))

The following expression results in the same value as the previous one, using members of the .NET Framework System.String class instead of Visual Basic functions:   
=User!UserID.Substring(User!UserID.IndexOf("\")+1, User!UserID.Length-User!UserID.IndexOf("\")-1)

• **Join** - Display the selected values from a multivalue parameter  
=Join(Parameters!MyParameter.Value,",")

•The **Regex** functions from the .NET Framework System.Text.RegularExpressions are useful for changing the format of existing strings, for example, formatting a telephone number. The following expression uses the Replace function to change the format of a ten-digit telephone number in a field from "nnn-nnn-nnnn" to "(nnn) nnn-nnnn":   
=System.Text.RegularExpressions.Regex.Replace(Fields!Phone.Value, "(\d{3})[ -.]\*(\d{3})[ -.]\*(\d{4})", "($1) $2-$3")

There are many other function which can be used in expression as shown below:  
[](http://4.bp.blogspot.com/_2t15yG-FxQA/THLlp_lUagI/AAAAAAAAA5A/vayybAvgOoI/s1600/1.png)

**Conversion Functions**

You can use Visual Basic functions to convert a field from the one data type to a different data type.

The following expression converts the constant 100 to type Decimal in order to compare it to a Transact-SQL money data type in the Value field for a filter expression: =CDec(100)

* The following expression displays the number of values selected for the multivalue parameter MyParameter: =CStr(Parameters!MyParameter.Count)

**Decision Functions**

The **IIF** function returns one of two values depending on whether the expression is true or false. The following expression uses the iif function to return a Boolean value of True if the value of Total exceeds 100. Otherwise it returns False:

=IIF(Fields!Total.Value > 100, True, False)

Use multiple IIF functions (nested IIFs) to return one of three values depending on the value of PercentComplete. The following expression can be placed in the fill color of a text box to change the background color depending on the value in the text box.

=IIF(Fields!PercentComplete.Value >= 10, "Green", IIF(Fields!PercentComplete.Value >= 1, "Blue", "Red"))

A different way to get the same functionality uses the **Switch** function. The Switch function is useful when you have three or more conditions to test. The Switch function returns the value associated with the first expression in a series that evaluates to true:

=Switch(Fields!PercentComplete.Value >= 10, "Green", Fields!PercentComplete.Value > 1, "Blue", Fields!PercentComplete.Value = 1, "Yellow", Fields!PercentComplete.Value <= 0, "Red",)

A third way to get the same functionality uses the **Choose** function. The Choose function uses the first parameter as an index to one of the remaining function parameters. The first parameter must be an integer. If the background color of a text box in a table is set to this expression, the value of MyIndex controls the color.

=Choose(Fields!MyIndex.Value,"Red","Green","Yellow")

Check the value of the PurchaseDate field and return "Red" if it is more than a week old, and "Blue" otherwise. This expression can be used to control the Color property of a text box in a report item:

=IIF(DateDiff("d",Fields!PurchaseDate.Value, Now())>7,"Red","Blue")

Posted by Hari Sharma at [5:07 PM](http://sql-bi-dev.blogspot.com/2010/09/ssrs-expressions.html) [http://img1.blogblog.com/img/icon18_email.gif](https://www.blogger.com/email-post.g?blogID=7924929528826389309&postID=1355622785320931385)[http://img2.blogblog.com/img/icon18_edit_allbkg.gif](https://www.blogger.com/post-edit.g?blogID=7924929528826389309&postID=1355622785320931385&from=pencil)

[Email This](https://www.blogger.com/share-post.g?blogID=7924929528826389309&postID=1355622785320931385&target=email)[BlogThis!](https://www.blogger.com/share-post.g?blogID=7924929528826389309&postID=1355622785320931385&target=blog)[Share to Twitter](https://www.blogger.com/share-post.g?blogID=7924929528826389309&postID=1355622785320931385&target=twitter)[Share to Facebook](https://www.blogger.com/share-post.g?blogID=7924929528826389309&postID=1355622785320931385&target=facebook)[Share to Pinterest](https://www.blogger.com/share-post.g?blogID=7924929528826389309&postID=1355622785320931385&target=pinterest)

Labels: [Conditional Formatting](http://sql-bi-dev.blogspot.com/search/label/Conditional%20Formatting), [SSRS](http://sql-bi-dev.blogspot.com/search/label/SSRS), [SSRS Expressions](http://sql-bi-dev.blogspot.com/search/label/SSRS%20Expressions)

|  |  |
| --- | --- |
| Reactions: |  |



* Sample disaster recovery plan
* Followers
* Return
* Web Services
* Services
* Small business ideas
* Resume Format
* Shares

PreviousDOJ   =IIF(IsNothing(Previous(Fields!DOJ.Value)),"Joined First",Previous(Fields!DOJ.Value))   
GroupByDesignation  =Runningvalue(Fields!Designation.Value,countdistinct,nothing)  
  
RowNumber  =RowNumber("DataSet1")  
  
RunningValue  =RunningValue(Fields!Salary.Value, Sum, Nothing)  
  
Count(EmployeeID)  =Count(Fields!EmployeeId.Value)  
  
TotalRowsCount  =CountRows("DataSet1")  
  
CountDistinct(Designation)  =CountDistinct(Fields!Designation.Value,"DataSet1")

**PRAGMATIC WORKS CHEAT SHEET**

SSRS Expression Cheat Sheet

|  |  |
| --- | --- |
| **Problems** | **Expression** |
| Return first day of current Week  (ex. Default Start Date parameter to return WTD) | **Expression on the parameter default value:**  =DateAdd("d",-DatePart(DateInterval.WeekDay,Today,0,0)+1,Today)  **Expression Output Example:** 11/7/2010 12:00:00 AM |
| Return first day of current Month   (ex. Default Start Date parameter to return MTD) | **Expression on the parameter default value:** =DateAdd("d",-(Day(today)-1), Today)  or  =DateSerial( year(today()), month(today()), 1)  **Expression Output Example:**11/1/2010 12:00:00 AM |
| Return first day of current year  ex. Default Start Date parameter to return YTD) | **Expression on the parameter default value:**  =DateAdd("d",-DatePart(DateInterval.DayOfYear,Today,0,0)+1,Today)  **Expression Output Example:** 1/1/2010 12:00:00 AM |
| Return period over period  (ex. Default date parameters to a rolling year) | **Expression on the parameter default value:**   **Week over Week**  =DateAdd("ww",-1, Today)   **Month over Month**  =DateAdd("m",-1,Today)   **Year over Year**  =DateAdd("yyyy",-1, Today)   **Expression Output Example:10/9/2010 12:00:00 AM** |
| Return current month name | **Expression in Text Box:**  =MonthName(Month(Today()))   Expression Output Example:November |
| Uppercase fields | **Expression in Text Box:**  =UCASE(Fields!FieldName.Value)   **Expression Output Example:**NOVEMBER |
| Convert text to proper case  (ex. 1st letter in each word is uppercase) | **Expression in Text Box:**  =StrConv(Fields!FieldName.Value, VbStrConv.ProperCase) |
| Replace NULL with another value | **Expression in Text Box:**  =iif(Fields!FieldName.Value = nothing, "No Value",Fields! FieldName.Value) |
| Alternating row color (Banding effect) | **BackgroundColor property on Text Box:**  =iif(RowNumber(Nothing) Mod 2 = 0, "Silver", "White") |
| Handling division by zero | **Expression in Text Box:**   =iif(Fields!DenominatorField.Value = 0, 0, Fields!NumeratorField.Value/   iif(Fields!DenominatorField.Value = 0, 1, Fields! DenominatorField.Value)) |
| security number) | **Expression in Text Box:**   =Replace(Fields!EmailAddress.Value,"-","") |

**SQL SERVER CENTRAL**

### *****EXPRESSIONS*****

A lot of these expressions would come in very handy when you setup report parameters. I would also suggest reading up on DateAdd, DatePart and other functions used in these expressions for more information. Note: all code are highlighted in blue.

1. Lets start with the easiest of the expressions i.e. To get Today’s date   
=Today

2. Get Date +/- a number of days - The following code will get Today's date -3 days. e.g. If today is the 08/12/2009, it should return 05/12/2009. You can change the -3 to any number and you should get the current date +/- the number specified. The DateAdd function basically adds an interval to a date. Here we add the interval -3 to Today's date.  
**=DateAdd("d", -3, Today)**

3. Get First Day of the Month - This is an extension of the pervious expression, it can be divided into two parts the -1.0 \* DatePart("D", Today) + 1 returns the number of days from the first day of the month -1 in negative. i.e. If today is the 08/12/2009 it would return the value -7 which is then passed to the function DateAdd to get the first day of the month e.g. 01/12/2009  
=DateAdd("D", -1.0 \* DatePart("D", Today) + 1, Today)

4. Get Last Day of the Month - This is almost the same as the last expression, the only difference is we don't add the +1 which gives us the last day of the last month then pass that date to another DateAdd function to add an extra month.  
=DateAdd("m", 1, DateAdd("d", -1.0 \* DatePart("d", Today), Today))

5. Get the First Day of the Last Month - This is another variation of the above expressions  
=DateAdd("D", -1.0 \* DatePart("D", Today) + 1, DateAdd("m", -1, Today))

6. Get the Name of the day - This gets the name for the day that was passed for example if its 08/12/2009 it would display "Tuesday"  
=WeekdayName(DatePart("w", Today))

7. Check what day it is and display date accordingly (This checks to see if today is a Monday and if it is true then displays today -3 days else displays today -1 day)  
=IIF(WeekdayName(DatePart("w", Today))="Monday",DateAdd("d", -3, Today),DateAdd("d", -1, Today))

8. Formatting Dates

|  |  |  |
| --- | --- | --- |
| Get Year - Will display only the year | **=Format(Today,"yyyy")** | Years as 1900-9999 |
| Get Month (without 0 suffix) | **=Format(Today,"M")** | Months as 1-12 |
| Get Month (with 0 suffix) | **=Format(Today,"MM")** | Months as 01-12 |
| Get Month by short Name | **=Format(Today,"MMM")** | Months as Jan-Dec |
| Get Month by Name | **=Format(Today,"MMMM")** | Months as January-December |
| Get Date (without 0 suffix) | **=Format(Today,"d")** | Days as 1-31 |
| Get Date (with 0 suffix) | **=Format(Today,"dd")** | Days as 01-31 |
| Get Date by short Name | **=Format(Today,"ddd")** | Days as Sun-Sat |
| Get Date by Name | **=Format(Today,"dddd")** | Days as Sunday-Saturday |
| Get Date in dd/mm/yyyy hh:mm:ss tt | **=Format(cdate("01/31/2009 16:00:00"),"dd MM yyyy hh:mm:ss tt")** | 31/01/2009 08:00:00 PM |

9. Convert String to Date - Following example combines string and integer values to form a valid date format  
=CDate( "1/" & Parameters!StartMonth.Value & "/" & Parameters!StartYear.Value)

10. SWITCH statement - An alternative to IIF/CASE (Returns a value depending on which condition is true)  
=SWITCH(WeekdayName(Fields!Date.Value) = "Monday","Blue",  
WeekdayName(Fields!Date.Value) = "Tuesday","Green",  
WeekdayName(Fields!Date.Value) = "Wednesday","Red")

11. Format Numbers as Currency - The result for the following will be $1000.00  
=FormatCurrency(1000)

12. Convert integer values to string  
= CStr(123123)

### *****CUSTOM CODE FUNCTIONS*****

You will find that not all your requirements can be easily accomplished with functions available in SSRS. To accomplish this we can use the custom code option to write our own vb.net functions in the report and then reference them from the report. You will have to copy this code to the code section of the report properties (Click on the Report Menu and select Report Properties, in the dialog box that pops up select the Code tab and paste these functions)

1. Get Time in hh:mm:ss from seconds - In scenarios where you get the output from the database in seconds but you want the results to be displayed in the format hh:mm:ss you can use this function to pass the seconds value to the function and it will return a string value in the format of hh:mm:ss. Sample Usage of the Function from the report =**Code.GetDurationAsString(10000)**. This should return 02:46:40

Insert the code below into the Code Section under the Report Properties.

Code Description: The function below accepts values in double and returns value as string. We declare 3 variables one each to store the hours, minutes and seconds values. Then use the following formula to get the hours Math.Floor(total / 3600) and then use the following formula Math.Floor((total - (hour \* 3600)) / 60) to get the minutes and finally use the following Math.Floor(total - (hour \* 3600) - (min \* 60)) for the seconds. We then declare a variable retThis to store the string value that we would be returned, then put the hours, mins and second values together and pad them with 0 accordingly and store it in the retThis variable and finally return the calculated retThis value

Function GetDurationAsString(ByVal total As Double) As String  
Dim hour As Integer  
hour = Math.Floor(total / 3600)  
Dim min As Integer  
min = Math.Floor((total - (hour \* 3600)) / 60)  
Dim sec As Integer  
sec = Math.Floor(total - (hour \* 3600) - (min \* 60))  
Dim retThis As String  
If hour > 999 Then  
retThis = Microsoft.VisualBasic.Right("0000" + RTrim(CType(hour, String)), 4) & ":" & Microsoft.VisualBasic.Right("00" & RTrim(CType(min, String)), 2) & ":" & Microsoft.VisualBasic.Right("00" & RTrim(CType(sec, String)), 2)  
ElseIf hour > 99 Then  
retThis = Microsoft.VisualBasic.Right("000" + RTrim(CType(hour, String)), 3) & ":" & Microsoft.VisualBasic.Right("00" + RTrim(CType(min, String)), 2) & ":" & Microsoft.VisualBasic.Right("00" & RTrim(CType(sec, String)), 2)  
Else  
retThis = Microsoft.VisualBasic.Right("00" + RTrim(CType(hour, String)), 2) & ":" & Microsoft.VisualBasic.Right("00" + RTrim(CType(min, String)), 2) & ":" & Microsoft.VisualBasic.Right("00" & RTrim(CType(sec, String)), 2)  
End If  
Return retThis  
End Function

2. Get time in seconds from hh:mm:ss - This function is the reverse of the previous function where you pass the string values in hh:mm:ss and it will return the values in seconds. Sample Usage of the Function from the report =Code.GetTimeinSec(12:40:45)

Insert the code in blue below into the Code Section under the Report Properties.

Code Description: The function GetTimeinSec accepts string values in hh:mm:ss format. This value is then broken up into 3 parts i.e. hours,mins and seconds using the Substring function. Each of the part is converted to integer using the Convert.ToInt32 function. We then calculate the seconds of each of the part i.e. hh \* 360 to get the seconds from the hour, mm \* 60 to get the seconds from the minutes and finally the ss for the seconds. All the calculated parts are then added up together and returned back.

Function GetTimeinSec(ByVal textVal As String) As Integer  
Dim Total As String = ""  
Total = Convert.ToInt32(textVal.Substring(0, 2)) \* 360 + Convert.ToInt32(textVal.Substring(3, 2)) \* 60 + Convert.ToInt32(textVal.Substring(6, 2))  
Return Convert.ToInt32(Total)  
End Function

3. Check for Divide by Zero - You will find that when setting up reports especially with calculated columns in tables there a situations where you get a divide by zero error . You can use this function to make sure that SSRS doesn't display the ugly #error. Sample Usage of the Function from the report:

=Code.isDividebyZero(1,0))

Insert the code in blue below into the Code Section under the Report Properties.

Code Description: The Function IsDividedbyZero accepts two values, the numerator and the denominator. The function checks to see if the denominator is 0 and if it is it returns a 0 else it returns the numerator / denominator value. This functions does the division and also checks if there is a divide by zero error.

Function IsDividedbyZero(ByVal Numerator As Double, ByVal Denominator As Double) As Double  
If Denominator = 0 Then  
IsDividedbyZero = 0  
Else  
IsDividedbyZero = Numerator / Denominator  
End If  
Return IsDividedbyZero  
End Function

4. Open a link in a new window and control the window properties - SSRS does not have an built-in option to open a link in a new browser window and neither does it have any option to control the size and properties of the window. This function uses a JavaScript workaround to achieve this functionality.

Sample Usage:=Code.NewWindows ("www.google.com.au","no","no","no"))

This will open a new window that opens the google website with the toolbars, menubar and statusbar hidden.

Insert the code in italics blue into the Code Section under the Report Properties.

Code Description: The function NewWindows accepts 4 parameters, the url, toolbars, menubar and statusbar. The url parameter is the url you want to open, the statusbar, menubar and statusbar accepts a value of yes, no or a blank value (if its blank its considered as yes). The values passed are combined with a javascript (window.open) function and returned back. Lookup javascript window.open on google to get more information. You can easily modify this function to add more functionality to it eg. window size etc.

Public Function NewWindows(ByVal url As String, ByVal toolbars As String, ByVal menubar As String, ByVal statusbar As String) As String  
If toolbars IsNothing Or toolbars = "" Then  
toolbars = "yes"  
End If  
If menubar IsNothing Or menubar = "" Then  
menubar = "yes"  
End If  
If statusbar IsNothing Or statusbar = "" Then  
statusbar = "yes"  
End If  
Return "javascript:void(window.open('http://" & url & "','\_blank','toolbar=" \_  
& toolbars & ",menubar=" & menubar & ",statusbar=" & statusbar & "'))"  
End Function

5. Display Popup Message on Click - This function gives you an option to display a message box from within the report items. For example you might want a popup when a user click on an image. You just add the following to the Jump to URL expression in image properties (right click the image and select properties - Select the Navigation Tab and select the radio button Jump to URL and copy the following). You can customize the message by passing report values. This opens a message box with the text "Please Contact Support on help@support.com".  
="javascript:void(alert('Please Contact Support on help@support.com'))"

### *****TIPS & TRICKS*****

I have included this third category as these dont quiet fit into the other 2 sections. This section covers functionality that's available in SSRS but might not be used as often or users just dont know about it.

1. Color alternate rows in table with a different color - This is very useful when there are a lot of rows with similar data and its hard to differentiate the rows. You have to set the table background color property with the following code.  
=iif(RowNumber(Nothing) Mod 2, "WhiteSmoke", "White")

2. To disable pagination - If you want to display the report in one single page you have to change change the report properties InteractiveHeight and InteractiveWidth to 0

3. Passing parameters to SSRS through the URL - If you plan to link to a report from your own website and if the report accepts parameters you can use the following sample to open the report pass the parameter values from the addressbar. This example passes 4 parameters Centre\_Contract, Emp\_ID,Start\_Date,End\_Date. Parameters start with rs:Command and each of the parameters are separated using & ).   
http://reporting.acme/ReportServer/Pages/ReportViewer.aspx?%2fUPA+Reports%2fUPASummary&rs:Command=Render&Centre\_Contract=21&Emp\_ID=11170&Start\_Date=2009-07-05&End\_Date=2009-07-06

4. Hide the parameters in a report - This is the same as the last option but if you add the following command (&rc:Parameters=false) at the end of the report url it will hide the parameters in the report. Very useful when you want to control what data can be viewed by which person especially from 3rd party applications.  
http://reporting.stellar/ReportServer/Pages/ReportViewer.aspx?%2fUPA+Reports%2fUPASummary&rs:Command=Render&Centre\_Contract=21&Emp\_ID=11170&Start\_Date=2009-07-05&End\_Date=2009-07-06&rc:Parameters=false

5. Dynamic DataSource - If you have a requirement for a report to be run against multiple database server you can change the datasource connection string property to be dynamic. You can write your own expression in the connection string option in the data source properties. The following will create a dynamic connection to the server depending on the report parameter (ServerName) selected. Note: You have to make sure that the login has permission on all the servers you would like to connect to.

="Data Source=" & Parameters!ServerName.Value & ";Initial Catalog=WSS\_Content"

## [Functions](javascript:void(0))

Many expressions in a report contain functions. You can format data, apply logic, and access report metadata using these functions. You can write expressions that use functions from the Microsoft Visual Basic run-time library, and from the [System.Convert](https://msdn.microsoft.com/en-us/library/system.convert.aspx) and [System.Math](https://msdn.microsoft.com/en-us/library/system.math.aspx) namespaces. You can add references to functions from other assemblies or custom code. You can also use classes from the Microsoft .NET Framework, including [System.Text.RegularExpressions](https://msdn.microsoft.com/en-us/library/system.text.regularexpressions.aspx).

### [Visual Basic Functions](javascript:void(0))

You can use Visual Basic functions to manipulate the data that is displayed in text boxes or that is used for parameters, properties, or other areas of the report. This section provides examples demonstrating some of these functions. For more information, see [Visual Basic Runtime Library Members](http://go.microsoft.com/fwlink/?LinkId=198941) on MSDN.

The .NET Framework provides many custom format options, for example, for specific date formats. For more information, see [Formatting Types](http://go.microsoft.com/fwlink/?LinkId=112024) on MSDN.

#### [Math Functions](javascript:void(0))

* The **Round** function is useful to round numbers to the nearest integer. The following expression rounds a 1.3 to 1:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_8379c82d-7280-4635-9cd5-ad40fe08490f');" \o "Copy to clipboard.)

= Round(1.3)

You can also write an expression to round a value to a multiple that you specify, similar to the **MRound** function in Excel. Multiply the value by a factor that creates an integer, round the number, and then divide by the same factor. For example, to round 1.3 to the nearest multiple of .2 (1.4), use the following expression:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_7429394c-066c-4a84-9a67-1b7fa8717a3e');)

= Round(1.3\*5)/5

#### [Date Functions](javascript:void(0))

* The **Today** function provides the current date. This expression can be used in a text box to display the date on the report, or in a parameter to filter data based on the current date.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_3e7256ee-3021-4488-a5bc-d15f6560f390');)

=Today()

* The **DateAdd** function is useful for supplying a range of dates based on a single parameter. The following expression provides a date that is six months after the date from a parameter named StartDate.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_19a6e22d-8752-4909-88c2-b13b632a508d');)

=DateAdd(DateInterval.Month, 6, Parameters!StartDate.Value)

* The **Year** function displays the year for a particular date. You can use this to group dates together or to display the year as a label for a set of dates. This expression provides the year for a given group of sales order dates. The **Month** function and other functions can also be used to manipulate dates. For more information, see the Visual Basic documentation.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_7f7d62f7-549f-4c36-9319-08b94ddd47e9');)

=Year(Fields!OrderDate.Value)

* You can combine functions in an expression to customize the format. The following expression changes the format of a date in the form month-day-year to month-week-week number. For example, 12/23/2009 to December Week 3:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_597a2ce4-65d2-4ed4-84be-7dc5a2055bd3');)

=Format(Fields!MyDate.Value, "MMMM") & " Week " &

(Int(DateDiff("d", DateSerial(Year(Fields!MyDate.Value),

Month(Fields!MyDate.Value),1), Fields!FullDateAlternateKey.Value)/7)+1).ToString

When used as a calculated field in a dataset, you can use this expression on a chart to aggregate values by week within each month.

* The following expression formats the SellStartDate value as MMM-YY. SellStartDate field is a datetime data type.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_fcba9c6a-9749-45ed-b920-9c9bb3d598f4');)

=FORMAT(Fields!SellStartDate.Value, "MMM-yy")

* The following expression formats the SellStartDate value as dd/MM/yyyy. The SellStartDate field is a datetime data type.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_80af0f6f-5e26-4fbd-9c86-2472ce57e9d9');)

=FORMAT(Fields!SellStartDate.Value, "dd/MM/yyyy")

* The **CDate** function converts the value to a date. The **Now** function returns a date value containing the current date and time according to your system. **DateDiff** returns a Long value specifying the number of time intervals between two Date values.

The following example displays the start date of the current year

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_e5144243-61fd-427c-a46c-62f52008fc5a');)

=DateAdd(DateInterval.Year,DateDiff(DateInterval.Year,CDate("01/01/1900"),Now()),CDate("01/01/1900"))

* The following example displays the start date for the previous month based on the current month.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_c1df7ae6-05fd-4279-9a2a-89331e33a783');)

=DateAdd(DateInterval.Month,DateDiff(DateInterval.Month,CDate("01/01/1900"),Now())-1,CDate("01/01/1900"))

* The following expression generates the interval years between SellStartDate and LastReceiptDate. These fields are in two different datasets, DataSet1 and DataSet2. The [First Function (Report Builder and SSRS)](https://msdn.microsoft.com/en-us/library/dd255270.aspx), which is an aggregate function, returns the first value of SellStartDate in DataSet1 and the first value of LastReceiptDate in DataSet2.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_84a0e614-0701-4771-8ed8-68ea812c6b2c');)

=DATEDIFF(“yyyy”, First(Fields!SellStartDate.Value, "DataSet1"), First(Fields!LastReceiptDate.Value, "DataSet2"))

* The **DatePart** function returns an Integer value containing the specified component of a given Date value.The following expression returns the year for the first value of the SellStartDate in DataSet1. The dataset scope is specified because there are multiple datasets in the report.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_0be29ac9-8346-4abc-a1e6-ec6d07199ab4');)

=Datepart("yyyy", First(Fields!SellStartDate.Value, "DataSet1"))

* The **DateSerial** function returns a Date value representing a specified year, month, and day, with the time information set to midnight. The following example displays the ending date for the prior month, based on the current month.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_8ce0f187-a401-4ca7-b37a-d92fc2958564');)

=DateSerial(Year(Now()), Month(Now()), "1").AddDays(-1)

* The following expressions display various dates based on a date parameter value selected by the user.

|  |  |
| --- | --- |
| **Example Description** | **Example** |
| Yesterday | =DateSerial(Year(Parameters!TodaysDate.Value),Month(Parameters!TodaysDate.Value),Day(Parameters!TodaysDate.Value)-1) |
| Two Days Ago | =DateSerial(Year(Parameters!TodaysDate.Value),Month(Parameters!TodaysDate.Value),Day(Parameters!TodaysDate.Value)-2) |
| One Month Ago | =DateSerial(Year(Parameters!TodaysDate.Value),Month(Parameters!TodaysDate.Value)-1,Day(Parameters!TodaysDate.Value)) |
| Two Months Ago | =DateSerial(Year(Parameters!TodaysDate.Value),Month(Parameters!TodaysDate.Value)-2,Day(Parameters!TodaysDate.Value)) |
| One Year Ago | =DateSerial(Year(Parameters!TodaysDate.Value)-1,Month(Parameters!TodaysDate.Value),Day(Parameters!TodaysDate.Value)) |
| Two Years Ago | =DateSerial(Year(Parameters!TodaysDate.Value)-2,Month(Parameters!TodaysDate.Value),Day(Parameters!TodaysDate.Value)) |

#### [String Functions](javascript:void(0))

* Combine more than one field by using concatenation operators and Visual Basic constants. The following expression returns two fields, each on a separate line in the same text box:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_13aa5596-5904-4e0b-846f-f1735cf86ae7');)

=Fields!FirstName.Value & vbCrLf & Fields!LastName.Value

* Format dates and numbers in a string with the **Format** function. The following expression displays values of the StartDate and EndDate parameters in long date format:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_924a81a9-a0af-4a24-9e76-cc7b24e6ba25');)

=Format(Parameters!StartDate.Value, "D") & " through " & Format(Parameters!EndDate.Value, "D")

If the text box contains only a date or number, you should use the **Format** property of the text box to apply formatting instead of the **Format** function within the text box.

* The **Right**, **Len**, and **InStr** functions are useful for returning a substring, for example, trimming DOMAIN\username to just the user name. The following expression returns the part of the string to the right of a backslash (\) character from a parameter named User:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_8805eb1a-26a7-4efc-8dd2-0500ea92cd31');)

=Right(Parameters!User.Value, Len(Parameters!User.Value) - InStr(Parameters!User.Value, "\"))

The following expression results in the same value as the previous one, using members of the .NET Framework [System.String](https://msdn.microsoft.com/en-us/library/system.string.aspx) class instead of Visual Basic functions:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_60020cbf-1552-47c4-a94e-8541d8fa5c33');)

=Parameters!User.Value.Substring(Parameters!User.Value.IndexOf("\")+1, Parameters!User.Value.Length-Parameters!User.Value.IndexOf("\")-1)

* Display the selected values from a multivalue parameter. The following example uses the **Join** function to concatenate the selected values of the parameter MySelection into a single string that can be set as an expression for the value of a text box in a report item:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_2e466eb6-2e78-4a95-b0a0-ce1dba0f9862');)

= Join(Parameters!MySelection.Value)

The following example does the same as the above example, as well as displays a text string prior to the list of selected values.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_f8076cb3-a6c7-42a4-9b6b-159939df99e0');)

=”Report for “ & JOIN(Parameters!MySelection.Value, “ & “)

* The **Regex** functions from the .NET Framework [System.Text.RegularExpressions](https://msdn.microsoft.com/en-us/library/system.text.regularexpressions.aspx) are useful for changing the format of existing strings, for example, formatting a telephone number. The following expression uses the **Replace** function to change the format of a ten-digit telephone number in a field from "nnn-nnn-nnnn" to "(nnn) nnn-nnnn":

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_9ab75e5f-5194-4acb-bce7-fcc25b6294f5');)

=System.Text.RegularExpressions.Regex.Replace(Fields!Phone.Value, "(\d{3})[ -.]\*(\d{3})[ -.]\*(\d{4})", "($1) $2-$3")

|  |
| --- |
| **System_CAPS_noteNote** |
| Verify that the value for Fields!Phone.Value has no extra spaces and is of type [String](https://msdn.microsoft.com/en-us/library/system.string.aspx). |

#### [Lookup](javascript:void(0))

* By specifying a key field, you can use the **Lookup** function to retrieve a value from a dataset for a one-to-one relationship, for example, a key-value pair. The following expression displays the product name from a dataset (“Product”), given the product identifier to match on:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_2425caee-a024-41fb-97d1-bfcb1658cc7a');)

=Lookup(Fields!PID.Value, Fields!ProductID.Value, Fields.ProductName.Value, "Product")

#### [LookupSet](javascript:void(0))

* By specifying a key field, you can use the **LookupSet** function to retrieve a set of values from a dataset for a one-to-many relationship. For example, a person can have multiple telephone numbers. In the following example, assume the dataset PhoneList contains a person identifier and a telephone number in each row. **LookupSet** returns an array of values. The following expression combines the return values into a single string and displays the list of telephone numbers for the person specified by ContactID:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_637965f9-73e1-4675-8b19-946cc4c334fb');)

=Join(LookupSet(Fields!ContactID.Value, Fields!PersonID.Value, Fields!PhoneNumber.Value, "PhoneList"),",")

#### [Conversion Functions](javascript:void(0))

You can use Visual Basic functions to convert a field from the one data type to a different data type. Conversion functions can be used to convert the default data type for a field to the data type needed for calculations or to combine text.

* The following expression converts the constant 500 to type Decimal in order to compare it to a Transact-SQL money data type in the Value field for a filter expression.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_4797fbac-e868-4f25-a1aa-8a722cc3cf55');)

=CDec(500)

* The following expression displays the number of values selected for the multivalue parameter MySelection.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_29a31e84-4dd1-4221-bd6e-af3c3cf7d488');)

=CStr(Parameters!MySelection.Count)

#### [Decision Functions](javascript:void(0))

* The **Iif** function returns one of two values depending on whether the expression is true or not. The following expression uses the **Iif** function to return a Boolean value of **True** if the value of LineTotal exceeds 100. Otherwise it returns **False**:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_cef0c402-a525-4dc8-8d96-dafd4f2c350b');)

=IIF(Fields!LineTotal.Value > 100, True, False)

* Use multiple **IIF** functions (also known as "nested IIFs") to return one of three values depending on the value of PctComplete. The following expression can be placed in the fill color of a text box to change the background color depending on the value in the text box.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_1ea29b19-367e-43fa-bf82-fd384e1e2729');)

=IIF(Fields!PctComplete.Value >= 10, "Green", IIF(Fields!PctComplete.Value >= 1, "Blue", "Red"))

Values greater than or equal to 10 display with a green background, between 1 and 9 display with a blue background, and less than 1 display with a red background.

* A different way to get the same functionality uses the **Switch** function. The **Switch** function is useful when you have three or more conditions to test. The **Switch** function returns the value associated with the first expression in a series that evaluates to true:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_340f933d-eec0-48b0-9aef-db4d33cf9c44');)

=Switch(Fields!PctComplete.Value >= 10, "Green", Fields!PctComplete.Value >= 1, "Blue", Fields!PctComplete.Value = 1, "Yellow", Fields!PctComplete.Value <= 0, "Red",)

Values greater than or equal to 10 display with a green background, between 1 and 9 display with a blue background, equal to 1 display with a yellow background, and 0 or less display with a red background.

* Test the value of the ImportantDate field and return "Red" if it is more than a week old, and "Blue" otherwise. This expression can be used to control the Color property of a text box in a report item:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_c422b8bc-b035-4d51-b20b-35fb6a082395');)

=IIF(DateDiff("d",Fields!ImportantDate.Value, Now())>7,"Red","Blue")

* Test the value of the PhoneNumber field and return "No Value" if it is **null** (**Nothing** in Visual Basic); otherwise return the phone number value. This expression can be used to control the value of a text box in a report item.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_b61979d1-416f-4e82-a1ae-3dd65d77e1f7');)

=IIF(Fields!PhoneNumber.Value Is Nothing,"No Value",Fields!PhoneNumber.Value)

* Test the value of the Department field and return either a subreport name or a **null** (**Nothing** in Visual Basic). This expression can be used for conditional drillthrough subreports.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_665a9b02-609c-4a56-a156-5eb2b596001e');)

=IIF(Fields!Department.Value = "Development", "EmployeeReport", Nothing)

* Test if a field value is null. This expression can be used to control the **Hidden** property of an image report item. In the following example, the image specified by the field [LargePhoto] is displayed only if the value of the field is not null.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_56edfe4f-44b3-4812-b6a7-385ff3c9b008');)

=IIF(IsNothing(Fields!LargePhoto.Value),True,False)

* The **MonthName** function returns a string value containing the name of the specified month. The following example displays NA in the Month field when the field contains the value of 0.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_335e8535-d6e9-44aa-9071-348f2cd9c62a');)

IIF(Fields!Month.Value=0,"NA",MonthName(IIF(Fields!Month.Value=0,1,Fields!Month.Value)))

### [Report Functions](javascript:void(0))

In an expression, you can add a reference to additional report functions that manipulate data in a report. This section provides examples for two of these functions. For more information about report functions and examples, see [Aggregate Functions Reference (Report Builder and SSRS)](https://msdn.microsoft.com/en-us/library/dd255275.aspx).

##### [Sum](javascript:void(0))

* The **Sum** function can total the values in a group or data region. This function can be useful in the header or footer of a group. The following expression displays the sum of data in the Order group or data region:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_b57065db-eedf-4335-a740-a875483a66bc');)

=Sum(Fields!LineTotal.Value, "Order")

* You can also use the **Sum** function for conditional aggregate calculations. For example, if a dataset has a field that is named State with possible values Not Started, Started, Finished, the following expression, when placed in a group header, calculates the aggregate sum for only the value Finished:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_aefa664f-f69a-4562-bfa1-8028c9b5120c');)

=Sum(IIF(Fields!State.Value = "Finished", 1, 0))

##### [RowNumber](javascript:void(0))

* The **RowNumber** function, when used in a text box within a data region, displays the row number for each instance of the text box in which the expression appears. This function can be useful to number rows in a table. It can also be useful for more complex tasks, such as providing page breaks based on number of rows. For more information, see [Page Breaks](https://msdn.microsoft.com/en-us/library/ms157328.aspx#PageBreaks) in this topic.

The scope you specify for **RowNumber** controls when renumbering begins. The **Nothing** keyword indicates that the function will start counting at the first row in the outermost data region. To start counting within nested data regions, use the name of the data region. To start counting within a group, use the name of the group.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_8a2e2ef6-07a7-4e5f-832a-5b7fca27d35b');)

=RowNumber(Nothing)

## [Appearance of Report Data](javascript:void(0))

You can use expressions to manipulate how data appears on a report. For example, you can display the values of two fields in a single text box, display information about the report, or affect how page breaks are inserted in the report.

### [Page Headers and Footers](javascript:void(0))

When designing a report, you may want to display the name of the report and page number in the report footer. To do this, you can use the following expressions:

* The following expression provides the name of the report and the time it was run. It can be placed in a text box in the report footer or in the body of the report. The time is formatted with the .NET Framework formatting string for short date:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_7f05f4a4-ac55-4fc6-9928-5c807b27b957');)

=Globals.ReportName & ", dated " & Format(Globals.ExecutionTime, "d")

* The following expression, placed in a text box in the footer of a report, provides page number and total pages in the report:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_8cbef510-16c9-4a05-97e4-353192c4fbf8');)

=Globals.PageNumber & " of " & Globals.TotalPages

The following examples describe how to display the first and last values from a page in the page header, similar to what you might find in a directory listing. The example assumes a data region that contains a text box named LastName.

* The following expression, placed in a text box on the left side of the page header, provides the first value of the LastName text box on the page:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_5550c65b-ee34-4d23-8364-762ba864ee8f');)

=First(ReportItems("LastName").Value)

* The following expression, placed in a text box on the right side of the page header, provides the last value of the LastName text box on the page:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_3e9c863d-507b-417d-ae91-a0b493fc849d');)

=Last(ReportItems("LastName").Value)

The following example describes how to display a page total. The example assumes a data region that contains a text box named Cost.

* The following expression, placed in the page header or footer, provides the sum of the values in the Cost text box for the page:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_4c7eafae-bb3b-4714-a4c7-e107b4ceb175');)

=Sum(ReportItems("Cost").Value)

|  |
| --- |
| **System_CAPS_noteNote** |
| You can refer to only one report item per expression in a page header or footer. Also, you can refer to the text box name, but not the actual data expression within the text box, in page header and footer expressions. |

### [Page Breaks](javascript:void(0))

In some reports, you may want to place a page break at the end of a specified number of rows instead of, or in addition to, on groups or report items. To do this, create a group that contains the groups or detail records you want, add a page break to the group, and then add a group expression to group by a specified number of rows.

* The following expression, when placed in the group expression, assigns a number to each set of 25 rows. When a page break is defined for the group, this expression results in a page break every 25 rows.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_b4ed5ffb-6eec-4860-b857-e1e0507652d2');)

=Ceiling(RowNumber(Nothing)/25)

To allow the user to set a value for the number of rows per page, create a parameter named RowsPerPage and base the group expression on the parameter, as shown in the following expression:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_ce86f7a8-3782-492b-935a-c3aceeeeb077');)

=Ceiling(RowNumber(Nothing)/Parameters!RowsPerPage.Value)

For more information about setting page breaks for a group, see [Add a Page Break (Report Builder and SSRS)](https://msdn.microsoft.com/en-us/library/dd207058.aspx).

## [Properties](javascript:void(0))

Expressions are not only used to display data in text boxes. They can also be used to change how properties are applied to report items. You can change style information for a report item, or change its visibility.

### [Formatting](javascript:void(0))

* The following expression, when used in the **Color** property of a text box, changes the color of the text depending on the value of the Profit field:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_4a7c35dc-f112-4d61-9dc2-7f5b6aefe485');)

=Iif(Fields!Profit.Value < 0, "Red", "Black")

You can also use the Visual Basic object variable Me. This variable is another way of referring to the value of a text box.

=Iif(Me.Value < 0, "Red", "Black")

* The following expression, when used in the **BackgroundColor** property of a report item in a data region, alternates the background color of each row between pale green and white:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_1bf76691-f9e4-4b4d-b7ca-c6d11a496aa1');)

=Iif(RowNumber(Nothing) Mod 2, "PaleGreen", "White")

If you are using an expression for a specified scope, you may have to indicate the dataset for the aggregate function:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_470ad202-ec3d-4586-acf3-0a333bc99e40');)

=Iif(RowNumber("Employees") Mod 2, "PaleGreen", "White")

|  |
| --- |
| **System_CAPS_noteNote** |
| Available colors come from the .NET Framework **KnownColor** enumeration. |

### [Chart Colors](javascript:void(0))

To specify colors for a Shape chart, you can use custom code to control the order that colors are mapped to data point values. This helps you use consistent colors for multiple charts that have the same category groups. For more information, see [Specify Consistent Colors across Multiple Shape Charts (Report Builder and SSRS)](https://msdn.microsoft.com/en-us/library/dd239350.aspx).

### [Visibility](javascript:void(0))

You can show and hide items in a report using the visibility properties for the report item. In a data region such as a table, you can initially hide detail rows based on the value in an expression.

* The following expression, when used for initial visibility of detail rows in a group, shows the detail rows for all sales exceeding 90 percent in the PctQuota field:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_b35ae82f-4649-4e96-be3f-cc085d6b70b3');)

=Iif(Fields!PctQuota.Value>.9, False, True)

* The following expression, when set in the **Hidden** property of a table, shows the table only if it has more than 12 rows:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_49be04e3-fb83-43d5-bb18-0415c3b54f43');)

=IIF(CountRows()>12,false,true)

* The following expression, when set in the Hidden property of a column, shows the column only if the field exists in the report dataset after the data is retrieved from the data source:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_44b8e688-165b-42d1-9c99-08a4f7f17a53');)

=IIF(Fields!Column\_1.IsMissing, true, false)

### [URLs](javascript:void(0))

You can customize URLs by using report data and also conditionally control whether URLs are added as an action for a text box.

* The following expression, when used as an action on a text box, generates a customized URL that specifies the dataset field EmployeeID as a URL parameter.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_ab0500f8-4461-4a99-aeda-6edcb58bb065');)

="http://adventure-works/MyInfo?ID=" & Fields!EmployeeID.Value

For more information, see [Add a Hyperlink to a URL (Report Builder and SSRS)](https://msdn.microsoft.com/en-us/library/dd239346.aspx).

* The following expression conditionally controls whether to add a URL in a text box. This expression depends on a parameter named IncludeURLs that allows a user to decide whether to include active URLs in a report. This expression is set as an action on a text box. By setting the parameter to False and then viewing the report, you can export the report Microsoft Excel without hyperlinks.

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_d32a6ac3-fadf-4d97-ade5-f0f4a724b3f5');)

=IIF(Parameters!IncludeURLs.Value,"http://adventure-works.com/productcatalog",Nothing)

## [Report Data](javascript:void(0))

Expressions can be used to manipulate the data that is used in the report. You can refer to parameters and other report information. You can even change the query that is used to retrieve data for the report.

### [Parameters](javascript:void(0))

You can use expressions in a parameter to vary the default value for the parameter. For example, you can use a parameter to filter data to a particular user based on the user ID that is used to run the report.

* The following expression, when used as the default value for a parameter, collects the user ID of the person running the report:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_ce0f3bad-ce27-4fc3-aa1b-6177a0b702c9');)

=User!UserID

* To refer to a parameter in a query parameter, filter expression, text box, or other area of the report, use the **Parameters** global collection. This example assumes that the parameter is named Department:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_539eb497-1b99-4089-801e-3a1553748838');)

=Parameters!Department.Value

* Parameters can be created in a report but set to hidden. When the report runs on the report server, the parameter does not appear in the toolbar and the report reader cannot change the default value. You can use a hidden parameter set to a default value as custom constant. You can use this value in any expression, including a field expression. The following expression identifies the field specified by the default parameter value for the parameter named ParameterField:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_ab894c8a-6e1c-4ff2-a931-d1705918cb52');)

=Fields(Parameters!ParameterField.Value).Value

**FROM HP PARTNERS**

**SSRS REPORT EXPRESSIONS USED:**

* =IIf(Sum(Fields!TotalVisit.Value)=0," -- ",FormatNumber(Sum(Fields!TotalVisit.Value),0))
* =IIf(Sum(Fields!TotalVisit.Value)=0," -- ",FormatNumber(Sum(Fields!TotalRVU.Value)/Sum(Fields!TotalVisit.Value),2))
* =Right(Fields!Yr\_Mth.Value,2) + "-" + LTrim(Left(Fields!Yr\_Mth.Value,4))
* =FormatCurrency(sum(Fields!ChgAmt.Value),0)
* =FormatCurrency(sum(Fields!ChgAmt.Value)/Fields!TOT\_Visits.Value,0)
* =iif(sum(Fields!ChgAmt.Value) = 0,"N/A",FormatPercent((sum(Fields!PmtAmt.Value)\*-1)/iif(sum(Fields!ChgAmt.Value) = 0,1,sum(Fields!ChgAmt.Value)),1))
* =FormatCurrency(code.divideby(sum(Fields!PmtAmt.Value),Fields!TOT\_Visits.Value,"N/A"),2)
* =FormatNumber(CODE.DivZero(Sum(Fields!PmtAmt.Value),Sum(Fields!Visits.Value)\* -1),2)
* =FormatPercent(IIf(sum(Fields!NetAR.Value)<=0,1,1-code.divideby(sum(Fields!NetAR.Value),sum(Fields!ChgAmt.Value),0)),1)
* =FormatNumber(code.divideby(sum(Fields!TotRvu.Value),Fields!TOT\_Visits.Value,"N?A"),2)
* =FormatCurrency(code.divideby(sum(Fields!PmtAmt.Value),Sum(Fields!TotRvu.Value),0),2)
* ="Procedure Code Analysis - "&Parameters!PhyType.Value
* ="Breakdown by " & Parameters!BreakDown.Value
* =IIf((Sum(Fields!Lv1.Value)+Sum(Fields!Lv2.Value)+Sum(Fields!Lv3.Value)+Sum(Fields!Lv4.Value)+Sum(Fields!Lv5.Value)+Sum(Fields!CritCare.Value))=0,"",FormatPercent(Sum(Fields!Lv1.Value)/(Sum(Fields!Lv1.Value)+Sum(Fields!Lv2.Value)+Sum(Fields!Lv3.Value)+Sum(Fields!Lv4.Value)+Sum(Fields!Lv5.Value)+Sum(Fields!CritCare.Value))))
* =IIf(Left(Parameters!TrendBy.Value,1)="M",Right(Fields!Group\_Filed.Value,2)&"/"&Left(Fields!Group\_Filed.Value,4),"Q"& right(Fields!Group\_Filed.Value ,1)&" - " &(Left(Fields!Group\_Filed.Value,4)))
* ="(" & Fields!DTACRO.Value & "-" & Fields!DTLOC.Value & ") - " & Fields!DLONAME.Value
* =Right(Fields!Group\_Filed.Value,2)&"/"&Left(Fields!Group\_Filed.Value,4)
* =(Fields!DDRNAME.Value) &" "& Fields!DDRTITL.Value
* ="\*\*\*\* CPT Code - " & Fields!DTPROC.Value
* =Code.GetBoldJoin(Trim(Fields!ProvFullNameDegree.Value), Trim(Fields!PhysicianStatus.Value))
* =Count(IIF(DateDiff("d", Today, IIF(IsNothing(Fields!DpsLicenseExpiration.Value), Today, Fields!DpsLicenseExpiration.Value)) <= 30 And DateDiff("d", Today, IIF(IsNothing(Fields!DpsLicenseExpiration.Value), Today, Fields!DpsLicenseExpiration.Value)) > 0, 1, Nothing))
* =First(Fields!EmployeeName.Value, "EvalPerformance")
* =ReportUtil.Localization.LocalizedString("DeliveryCountColHead",User!Language)
* =Count(Fields!DeliveryNumber.Value)/Count(Fields!DeliveryNumber.Value,"Hub")
* =Year(Fields!PickupDateTime.Value)
* =Sum(Fields!LineAmount.Value) - Code.TaxDeductions(Sum(Fields!LineAmount.Value))
* =WeatherInfo.PlanetaryWeather.GetWeather(Fields!PlanetAbbrv.Value)

**FROM BHG**

* ="Funded by closer for: " + MonthName(Parameters!Month.Value) + " " + cstr(Parameters!Year.Value)
* This is an expression used for the background color fill
  + =IIF(RunningValue(Fields!CloserName.Value,COUNTDISTINCT,NOTHING) MOD 2,"White","LightGrey")
  + =IIf(RowNumber(Nothing) Mod 2 = 0,"WhiteSmoke","White")
* =DateAdd("d", -6, TODAY())
* =IIF(Parameters!ReportType.Value = 1,"Core Withdrawn Before Submission","Initial Decline")
* = "Days Remaining : " & IIF(DateDiff ("d",Today,"2015-03-10 00:00:00.000")<0,0,DateDiff ("d",Today,"2015-03-10 00:00:00.000"))
* THIS FORMATS AS A PERCENTAGE AND CONCATENATES TWO FIELDS INTO ONE EXPRESSION
* =IIf(Sum(Fields!ResponseCount.Value) = 0, 0, Format(Sum(Fields!ScoredCount.Value) / IIf(Sum(Fields!ResponseCount.Value) = 0, 1, Sum(Fields!ResponseCount.Value)),"P")) & " / " & Sum(Fields!ScoredCount.Value)
* =Month(DateAdd(DateInterval.Minute,-3,Now()))
* =year(today())
* THIS IS USED AS CONDITIONAL FORMATTING FOR A fill EXPRESSION IN A TEXT BOX PROPERTY
  + =Switch(Fields!FundedAmount.Value <= 200000, "Red",Fields!FundedAmount.Value <= 400000, "Orange",Fields!FundedAmount.Value <= 600000, "Yellow",Fields!FundedAmount.Value > 600000, "Green")
* This is used to capture the month to date for the Start Date Parameter
  + =dateadd("d",-datepart("d",today())+1,today())

**FROM UNIVERSAL**

* ="Unearned Premium Detail through: " + Format(Parameters!EndDate.Value, "MM/dd/yyyy") + vbCRLF +
* "for policy effective dates ending " + Format(iif(Parameters!EffectiveEndDate.Value is nothing, Parameters!EndDate.Value, Parameters!EffectiveEndDate.Value), "MM/dd/yyyy") + vbCRLF +
* "for transactions evaluated as of " + Format(iif(Parameters!BookedEndDate.Value is nothing, Parameters!EndDate.Value, Parameters!BookedEndDate.Value), "MM/dd/yyyy")+ vbCRLF +
* "for policy types: " + Join(Parameters!PolicyForms.Label, ", ") + vbCRLF +
* "for states: " + Join(Parameters!States.Label, ", ") + vbCRLF +
* "for LOB's: " + Join(Parameters!LoBIDs.Label, ", ") + vbCRLF +
* "for companies: " + Join(Parameters!CompanyIDs.Label, ", " + vbCRLF) + vbCRLF +
* "Run Date: " + FormatDateTime(ToDay(), 1)

=Sum(Fields!TotalPaid.Value) / Sum(Fields!ClosedClaimCount.Value)

=First(Fields!AddressLine1.Value, "ClaimInfo") + " " + First(Fields!AddressLine2.Value, "ClaimInfo")

=First(Fields!City.Value, "ClaimInfo") + " " + First(Fields!StateCode.Value, "ClaimInfo") + " " + First(Fields!ZipCode.Value, "ClaimInfo")

=First(Fields!Insureds.Value, "ClaimInfo")

=CDate(Fields!createddate.Value).ToString("MM/dd/yyyy") + " By " + Fields!CreatedBy.Value

="Date Reported : " + CDate(Fields!DateReported.Value).ToString("MM/dd/yyyy")

="Total : " + CountRows().ToString()

=IIF(Parameters!TypeID.Value = 1, "Requested By","Verified By")

="USER NAME : " + IIF(TRIM(Fields!FriendlyUserName.Value) = "",Fields!username.Value, Fields!FriendlyUserName.Value )

=FormatNumber((Fields!NumberOfPhotos.Value \* Fields!PhotoRate.Value), 2)

=Fields!AssignedTo.Value.ToString().ToUpper()

="Claim Document Lookup - [" + FormatDateTime(Parameters!StartDate.Value, DateFormat.ShortDate) + " - " + FormatDateTime(Today(), DateFormat.ShortDate) + " ]"

=RunningValue(Fields!Losses.Value, sum, Nothing)

="Claim Activity Reminder"

="\*\*\* Multiple Assignments. # of Assignments: " + CountRows().ToString()

="Claims Reported Under Catastrophe [ " + First(Fields!Catastrophe.Value, "DataSet1") + " ]"

=Sum(Fields!AverageDaysToClose.Value) / Count(Fields!AverageDaysToClose)

=IIF(Sum(Fields!AverageIncurred.Value) = 0,0, Sum(Fields!AverageIncurred.Value) / IIF(Count(Fields!AverageIncurred.Value) = 0,1, Count(Fields!AverageIncurred.Value)))

=Code.Quotient(Sum(Fields!TotalIncurred.Value), Sum(Fields!Reported.Value), 0)

=IIF (Fields!InforceCount.Value = 0, "", Fields!InforceCount.Value)

=IIF (Sum(Fields!InforceCount.Value) = 0, "", Sum(Fields!InforceCount.Value))

="Claim Drafts Log from: " + formatdatetime(Parameters!StartDate.Value,2) + " Thru: " + formatdatetime(Parameters!EndDate.Value,2)

=Code.ConvertLossType(Fields!IsLAE.Value, Fields!IsAO.Value, Fields!Payee1.Value)

=Parameters!ClaimTransactionTypeID.Label

=" payments for the period: " + CDate(Parameters!StartingDate.Value).ToString("MM/dd/yyyy") + " - " +CDate(Parameters!EndingDate.Value).ToString("MM/dd/yyyy")

="In-force Policies (as of " & Parameters!AsOfDate.Value & ")"

=iif(Fields!Year.Value < 9999, Fields!Year.Value, "")

="Catastrophe - " + First(Fields!Storm.Value, "Bordereaux") + " as of " + CDate(Parameters!EndingDate.Value).ToString("MM/dd/yyyy")

=SUM(Fields!OpenTotalPaid.Value + Fields!OpenReserve.Value)

=Fields!Payee1.Value +

IIF(Fields!Payee2.Value = "","", ", " + VBCRLF + Fields!Payee2.Value) +

IIF(Fields!Payee3.Value = "","", ", " + VBCRLF + Fields!Payee3.Value) +

IIF(Fields!Payee4.Value = "","", ", " + VBCRLF + Fields!Payee4.Value)

**HOW TOs**

**How to set up GROUPING in SSRS**

* **USE the CCBalanceTransfers & CCBalanceActivation reports as a template**
* Right click on any field in the detail row.
* Go to “Add Group”, “Row Group”, “Parent Group” under Tablix.
* Group By the first column to be grouped on and choose Add header and Footer.
* Remove the extra old column.
* Delete the bottom record.
* Insert a row above the detail row.
* Fill in the middle row with a darker row.

**How to set up Multi Value Paramerters in SSRS**

select ' All' as BusinessName, 0 as AdjustingCoID

union all

select BusinessName, c.ContactId as AdjustingCoID

from Contact c

where c.ContactTypeID = 11 and c.ParentContactID is null

and c.contactid not in (1091, 22708, 23434, 2260102)

select DISTINCT ' All' as VendorType, 0 as TransTypeID

union all

SELECT DISTINCT ct.SettlementDesc, ct.ClaimTransactionTypeID

from dbo.ClaimTransactions as ct

where ct.ClaimTransactionTypeID IN (17, 19, 23, 34, 37, 39)

ORDER BY 1

**USING SPLIT FUNCTION**

**This example uses Cascading parameters called State and County. County depends on a parameter from State**

* **Declare @State & @County as varchar in main SP**

--[dbo].[reportYearOverReserveAnalysis] '20150101', '20150902', '20150902', 2, 15, 6, 'With Rep.'

ALTER procedure [dbo].[reportYearOverReserveAnalysis]

@StartingDate as datetime = null,

@EndingDate as datetime = null,

@AsOfDate as datetime = null,

@LookBackPeriod as integer = null,

@State as varchar(50) = null,

@County as varchar(50) = null,

@RepType as varchar(50) = null

* Use the split function in main SP

where coalesce(ibs.AccidentYear, cbs.AccidentYear) is not null

AND ps.[StateID] IN (SELECT StringValue FROM dbo.Split(@State,','))

AND ps.CountyID IN (SELECT StringValue FROM dbo.Split(@County,','))

* Set up a separate dataset for States

SELECT DISTINCT S.[StateID],S.Statecode, S.[StateDescription]

FROM States S

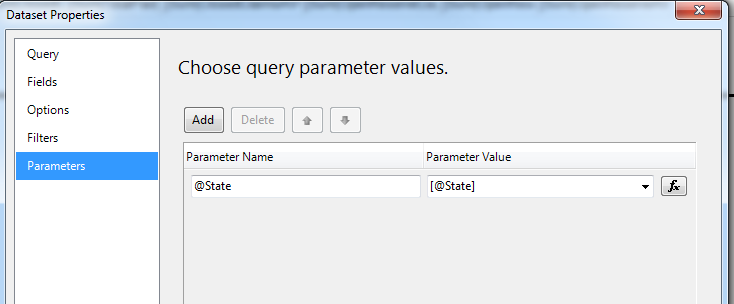
WHERE S.[StateID] IN (15,25,30,13,32,42,67,36,35,17,21)

* Set up a separate dataset for County. The County dataset requires a parameter from State

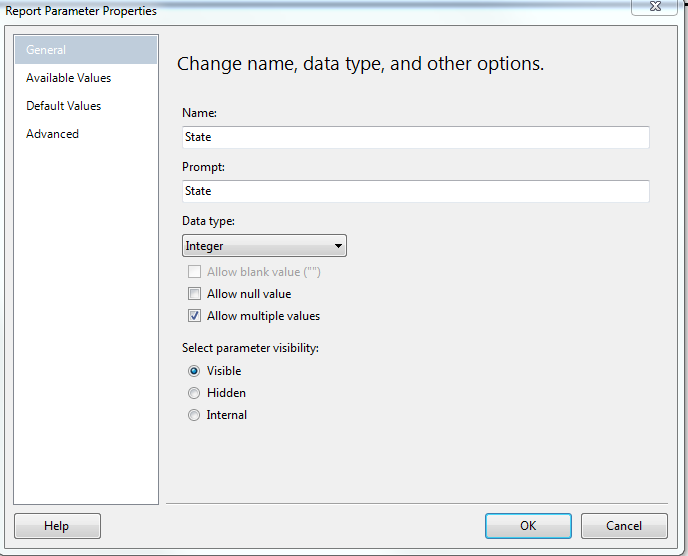
SELECT DISTINCT CountyID,[StateID], CountyDescription

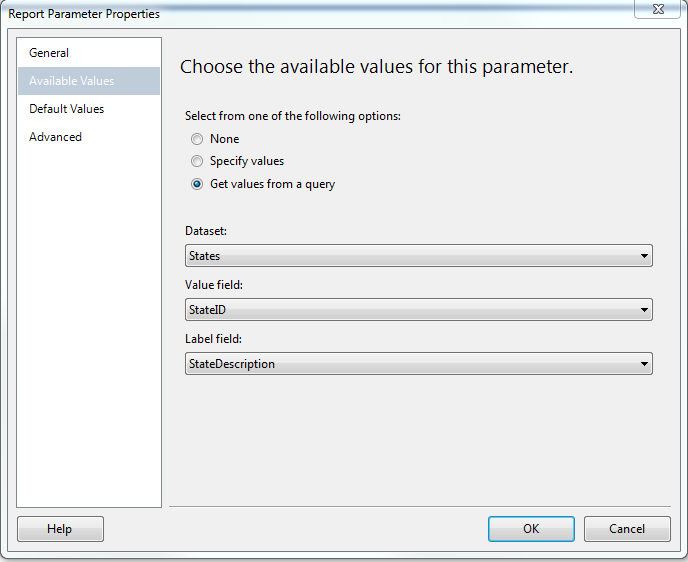
FROM [dbo].[County]

WHERE [StateID] IN (@State)



* Set the values for the parameters. Set the datatype to Integer





**USING CTEs**

##### Problem

Multi-select parameters give your users control over their reports while reducing the number of reports they have to work with. In this example, I will demonstrate how to create a multi-select parameter list and pass the values to a stored procedure that will then populate the report. I will be working with the AdventureWorks2008R2 database to create a report which will list sales quotas and amounts for selected Sales Reps.

##### Solution

The key to this solution is a delimited list of values that can be passed from the report to a stored procedure. I will be using the comma as a delimiter, but any delimiter will work.

For this example I will be using two parameters. The first parameter will provide a list of Sales Territories. The second parameter will provide a list of Sales Reps based on the selected Sales Territories from the first list. The report will show sales quotas and amounts for the selected Sales Reps.

### Step 1

Create a stored procedure that will return a list of Sales Territories. This stored procedure will be used by the first parameter of the report.

CREATE PROCEDURE dbo.ListSalesTerritory\_s

AS

SET NOCOUNT ON

SELECT

TerritoryID

,[Name] AS TerritoryName

FROM

Sales.SalesTerritory

ORDER BY

[Name]

SET NOCOUNT OFF

GO

### Step 2

Create a second stored procedure that will return a list of Sales Reps for 1 to *N* Sales Territories. This will be used by the second parameter of the report. The list of selected Sales Territories will be passed to the stored procedure as a comma delimited list of TerritoryIDs in a parameter called @TerritoryID. (If your list of values have commas, then you'll have to use a different delimiter.) The size of the parameter should be determined by the maximum list of values that might be sent to the stored procedure.

There are many different ways to break up a delimited list of values. They are usually referred to as *Split Functions*. I like to use a recursive CTE (Common Table Express) to split the values up. After the list of values are in a table structure, the table structure can be joined to the rest of the tables needed to return the list of Sales Reps.

CREATE PROCEDURE ListSalesRep\_s (@TerritoryIDs AS varchar(100))

AS

SET NOCOUNT ON;

WITH CTE\_Pieces

AS

(

SELECT

1 AS ID

,1 AS StartString

,CHARINDEX(',', @TerritoryIDs) AS StopString

UNION ALL

SELECT

ID + 1

,StopString + 1

,CHARINDEX(',', @TerritoryIDs, StopString + 1)

FROM

CTE\_Pieces

WHERE

StopString > 0

)

,CTE\_Split

AS

(

SELECT

CONVERT(int,SUBSTRING(@TerritoryIDs, StartString,

CASE

WHEN StopString > 0 THEN StopString - StartString

ELSE LEN(@TerritoryIDs)

END)) AS TerritoryID

FROM

CTE\_Pieces

)

SELECT

P.BusinessEntityID

,P.LastName + ', ' + P.FirstName AS SalesRep

FROM

CTE\_Split AS S

JOIN Sales.SalesPerson AS SP ON sp.TerritoryID = s.TerritoryID

JOIN Person.Person AS P ON SP.BusinessEntityID = P.BusinessEntityID

SET NOCOUNT OFF

### Step 3

Create the stored procedure for the body of the report. In this example, it will also have a parameter, @BusinessEntityIDs that will contain a comma delimited list of selected Sales Reps.

CREATE PROCEDURE dbo.RptSales\_s(@BusinessEntityIDs AS varchar(100))

AS

SET NOCOUNT ON;

WITH CTE\_Pieces

AS

(

SELECT

1 AS ID

,1 AS StartString

,CHARINDEX(',', @BusinessEntityIDs) AS StopString

UNION ALL

SELECT

ID + 1

,StopString + 1

,CHARINDEX(',', @BusinessEntityIDs, StopString + 1)

FROM

CTE\_Pieces

WHERE

StopString > 0

)

,CTE\_Split

AS

(

SELECT

CONVERT(int,SUBSTRING(@BusinessEntityIDs, StartString,

CASE

WHEN StopString > 0 THEN StopString - StartString

ELSE LEN(@BusinessEntityIDs)

END

)

) AS BusinessEntityID

FROM

CTE\_Pieces

)

SELECT

P.LastName + ', ' + P.FirstName AS SalesRep

,ST.Name AS TerritoryName

,ST.CountryRegionCode

,SP.SalesQuota

,SP.Bonus

,SP.SalesYTD

,SP.SalesLastYear

FROM

CTE\_Split AS s

JOIN Sales.SalesPerson AS SP ON s.BusinessEntityID = sp.BusinessEntityID

JOIN Sales.SalesTerritory AS ST ON SP.TerritoryID = ST.TerritoryID

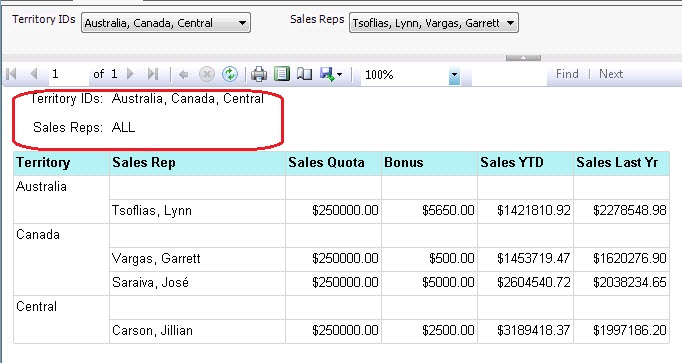
JOIN Person.Person AS P ON SP.BusinessEntityID = P.BusinessEntityID

SET NOCOUNT OFF

GO

### Step 4

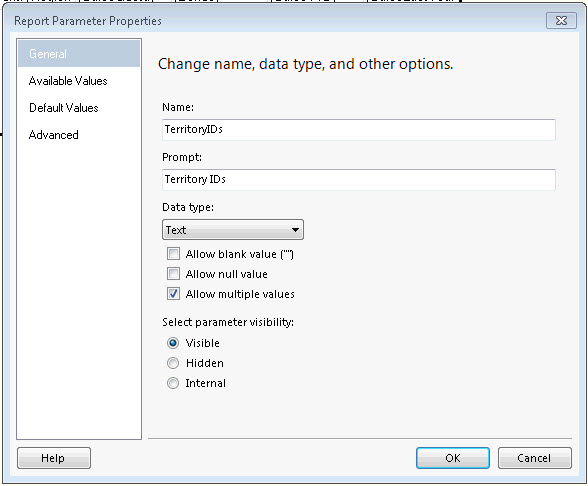
Add the three stored procedures that were created in steps 1 through 3 to a new report. Then create the layout for the report. My report is laid out by grouping the Sales Reps by Territory.

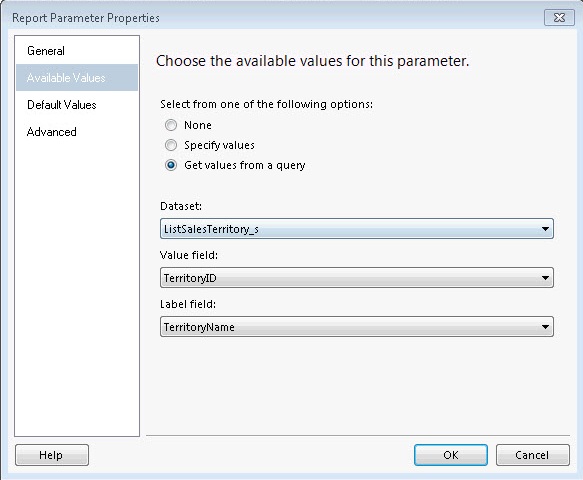
   
*Finished Report*

### Step 5

Set the Properties of both parameters by right clicking on the parameter and selecting Parameter Properties from the drop down list.

* Check the *Allow multiple values* checkbox.
* Select *Available Values* from the left hand list.
  + Select the *Get values from a query* option button.
  + Set the *Dataset* drop down list to the proper dataset.
  + Set the *Value field* drop down list to the proper field. This is the value that will be returned to the dataset that needs it.
  + Set the *Label field* drop down list to the proper field. This is the value that will be displayed to the user.

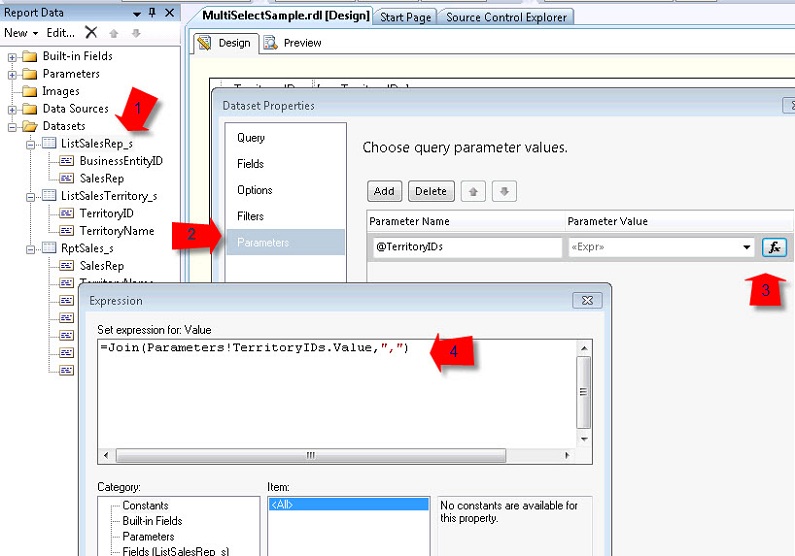




### Step 6

The list of selected values need to be returned to the ListSalesRep\_s and the RptSales\_s stored procedures. This will be done using the JOIN expression.

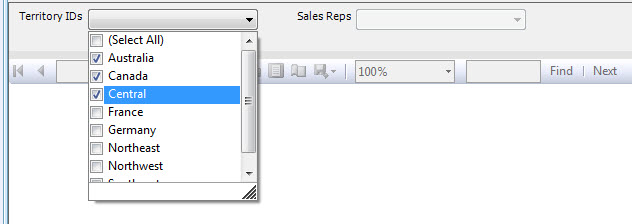
1. Right click on the stored procedure that needs the list of values and select *Dataset Properties* from the drop down list.
2. Select *Parameters* in the left hand list of the *Dataset Properties* dialog box.
3. In this example, the delimited list of Territories need to be assigned to the @TerritoryIDs parameter. Click the function button to enter a function for the parameter value.
4. Add the following code for the expression. Make sure to use the Value property, not the Label property or the wrong list of values will be sent to the stored procedure.

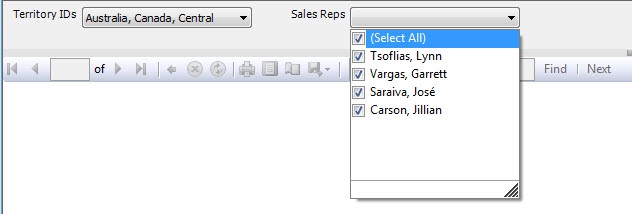


=Join(Parameters!TerritoryIDs.Value,",")

### Step 7

It's time to run your report. When the drop down lists are pulled down, there should be check boxes to select only the rows you want. Each time the selected items are changed in the Territory IDs drop down, the Sales Rep list will automatically be regenerated when the Sales Rep list is pulled down.

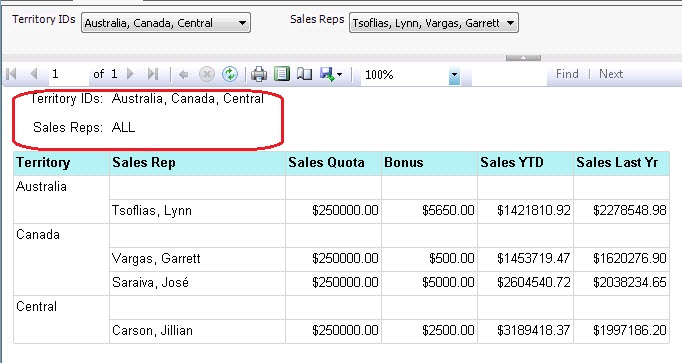




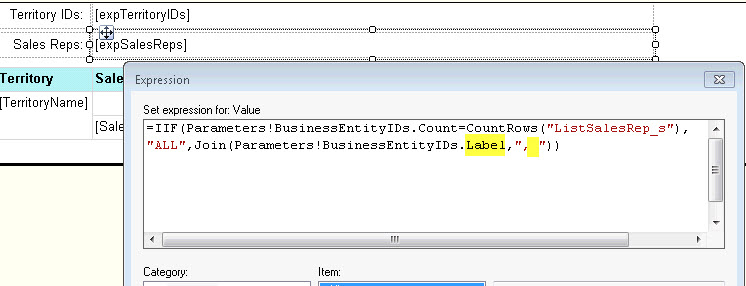
### BUT THAT'S NOT ALL...

I think all reports should display the values of the parameters used in the report. This way when someone brings up a concern about a report, you know exactly which parameter values were used.

If you'll notice in the sample below, I listed only the three Territories that were selected, but since all the Sales Reps for those territories are used in the report, the word *ALL* is used instead of listing each Sales Rep out individually. This can be done with an *IIF* expression in a Label control.



The first parameter of the *IIF* expression compares the count of how many items are in the parameter list, with how many that were selected. If the two values match, then the label *ALL* is used. If they don't match, then the *JOIN*expression is used again, but this time the Labels are joined together. Make sure to include a space after the delimiter so that the values don't run together.



**HOW TO DISPLAY THE VALUES OF A PARAMETER USED AT THE TOP OF THE REPORT.**

* Place desired text into a text box
  + **Year over Year Analysis Trend**
  + **States included are: [@State.Label]**
* **Expression used if it is not multi valued is** =Parameters!State.Label
* **Expression used if it is multi valued is** =Join(Parameters!State.Label, “,”) + vbCRLF +

**HOW TO remove a blank page 2 from printing from PDF file**

* Keep adjusting the right and left margins in report properties and shorten the heading widths until the blank page 2 disappears.