**HOW TO export SSRS report to separate EXCEL worksheets**

* Add a separate Table/Tablix to the report and call a separate SP for the separate logic.
* Set the page break to force a separate excel worksheet
* Name the worksheets in the PageName of the properties section.

="For the period " + FormatDateTime(Parameters!StartingDate.Value, DateFormat.ShortDate) + " - " + FormatDateTime(Parameters!EndingDate.Value, DateFormat.ShortDate)

=First(Fields!AgencyName.Value, "Query") & " (" & First(Fields!AgencyCode.Value, "Query") & ") as of: " & FormatDateTime(Parameters!AsOfDate.Value, DateFormat.ShortDate)

=Join(Parameters!State.Label,", ")

For Drop Downs

select DISTINCT ' All' as VendorType, 0 as TransTypeID

union all

SELECT DISTINCT ct.SettlementDesc as VendorType, ct.ClaimTransactionTypeID as TransTypeID

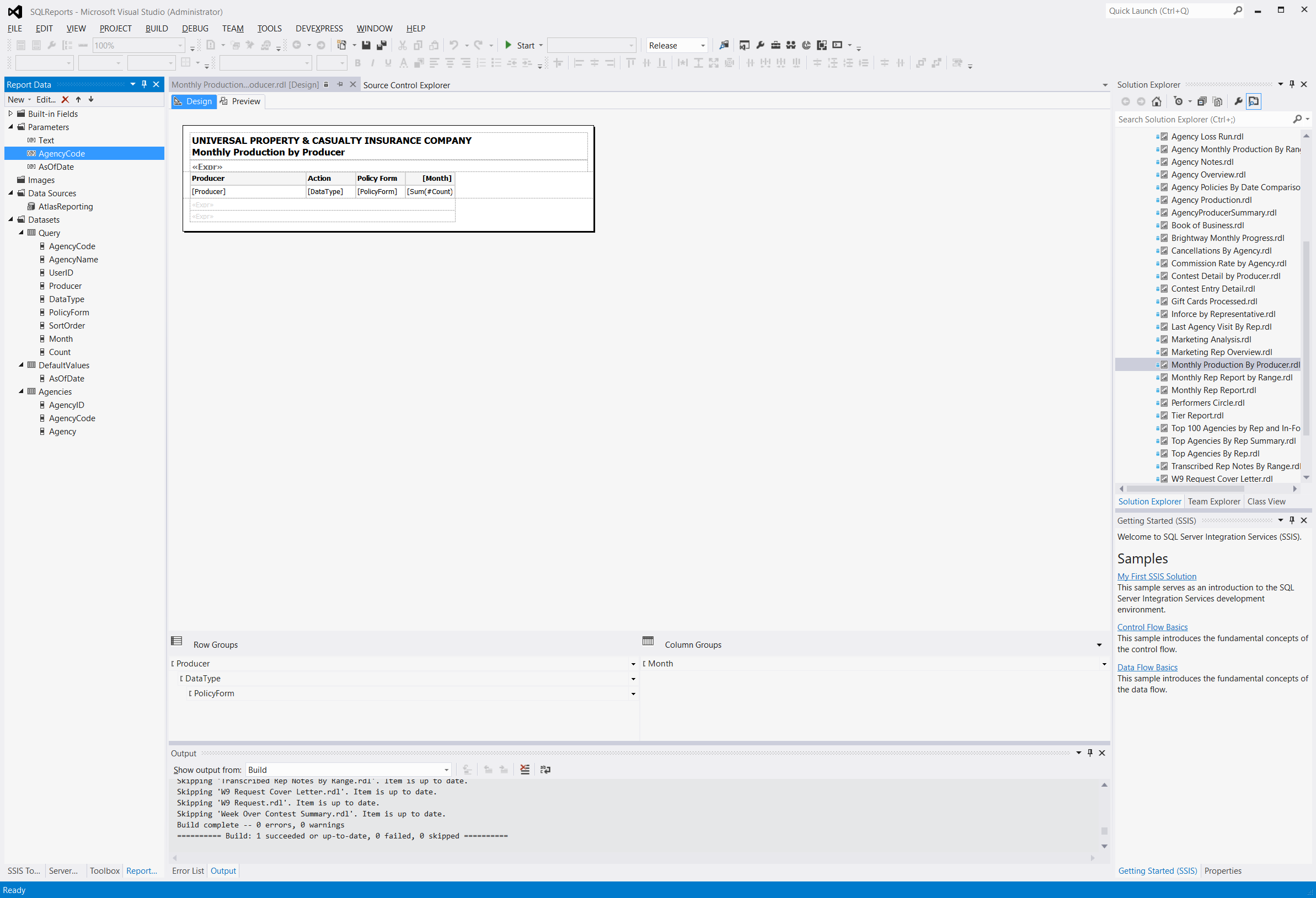
from dbo.ClaimTransactions as ct

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

ORDER BY 1

**HOW TO Create a Search Text Box Cascading Parameter for report**

1. Add a Parameter to the SSRS report itself(Not the SP) called “Text” or “Search Criteria” Prompt for called “Search Text:”
2. No Available or Default values
   1. Or set up a Default values Dataset for NULL default value
      1. select null as SearchCriteria
3. Have the SP Parameter get it’s values from a separate Dataset called “Agencies”
4. Have the Agencies dataset call the SP below called [dbo].[reportGetAgencyList] sending it the @Text parameter we just created in step 1



USE [Atlas]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[reportGetAgencyList] Script Date: 3/3/2016 11:46:14 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER procedure [dbo].[reportGetAgencyList]

@Text as varchar(512) = null

as

begin

set nocount on

declare @SearchText as varchar(100) = '%' + ltrim(rtrim(replace(@Text, ' ', '%'))) + '%'

create table #Results (

AgencyID integer,

AgencyCode varchar(8),

Agency varchar(100))

insert into #Results

select top 500

ag.AgencyID,

ag.AgencyCode,

ag.AgencyName as Agency

from dbo.Agency as ag

where ag.AgencyCode like @SearchText

or ag.AgencyName like @SearchText

if @@rowcount = 0

begin

if object\_id('tempdb.dbo.#Words') is not null drop table #Words

create table #Words (Word varchar(100))

create clustered index idx\_Word on #Words (Word)

insert into #Words

select '%' + sp.StringValue + '%' as Word

from dbo.Split(@Text, ' ') as sp

where sp.StringValue != ''

insert into #Results

select top 500

ag.AgencyID,

ag.AgencyCode,

ag.AgencyName as Agency

from dbo.Agency as ag

inner join #Words as wo

on ag.AgencyCode like wo.Word

or ag.AgencyName like wo.Word

end

select distinct

r.AgencyID,

r.AgencyCode,

concat(r.Agency, ' (', r.AgencyCode, ')') as Agency

from #Results as r

order by AgencyCode

if object\_id('tempdb.dbo.#Results') is not null drop table #Results

if object\_id('tempdb.dbo.#Words') is not null drop table #Words

end

**OR**

1. In SSRS Add a parameter called “@SearchCriteria”
2. Available values = NONE
3. Default values = Get values from a query
4. Create a separate dataset called DefaultValues
5. Set the default value for the @SearchCriteria param
   1. Select max([Date]) as AsOfDate,

1 as CompanyID,

15 as StateID,

'Colin' as SearchCriteria

from SevenDays

where [DayOfWeek] = 'Friday'

**OR**

SELECT null as SearchCriteria

Then create another dataset that is dependent upon the @SearchCriteria parameter

select ca.CatastropheID,

ca.[Description]

from dbo.Catastrophe as ca

where ca.[Description] **like** **concat('%',** **@SearchCriteria, '%')**

order by ca.[Description]

**OR**

**Call separate SP**

ALTER procedure [dbo].[reportGetMarketingGroup]

@SearchCriteria varchar(100) = null,

@ActiveOnly bit = 1

as

begin

set nocount on

if @SearchCriteria is not null

begin

set @SearchCriteria = ltrim(rtrim(@SearchCriteria))

while charindex(' ', @SearchCriteria) > 0 set @SearchCriteria = replace(@SearchCriteria, ' ', ' ')

if object\_id('tempdb.dbo.#Results') is not null drop table #Results

create table #Results (

SortOrder integer,

Value integer,

Label varchar(100))

insert into #Results

select 0 as SortOrder,

-re.RepID as Value,

re.FirstName + ' ' + re.LastName as Label

from Marketing.dbo.Representatives as re

where re.Active = isnull(@ActiveOnly, 1)

and ( re.FirstName + ' ' + re.LastName like '%' + @SearchCriteria + '%'

or re.LastName like '%' + @SearchCriteria + '%'

or re.FirstName like '%' + @SearchCriteria + '%')

union

select 1 as SortOrder,

-(at.TierID + 500) as Value,

at.Tier + ' Tier' as Label

from Atlas.dbo.AgencyTier as at

where at.Tier + ' Tier' like '%' + @SearchCriteria + '%'

or at.Tier like '%' + @SearchCriteria + '%'

union

select top 500

2 as SortOrder,

ag.AgencyID as Value,

'[' + ag.AgencyCode + '] - ' + ag.AgencyName as Label

from Atlas.dbo.Agency as ag

where ag.Active = isnull(@ActiveOnly, 1)

and ( ag.AgencyCode like '%' + @SearchCriteria + '%'

or ag.AgencyName like '%' + @SearchCriteria + '%')

if @@rowcount = 0

begin

if object\_id('tempdb.dbo.#Words') is not null drop table #Words

create table #Words (Word varchar(100))

create clustered index idx\_Word on #Words (Word)

insert into #Words

select '%' + sp.StringValue + '%'

from Common.dbo.Split(@SearchCriteria, ' ') as sp

where sp.StringValue != ''

insert into #Results

select 0 as SortOrder,

-re.RepID as Value,

re.FirstName + ' ' + re.LastName as Label

from Marketing.dbo.Representatives as re

inner join #Words as wo

on re.FirstName like wo.Word

or re.LastName like wo.Word

where re.Active = isnull(@ActiveOnly, 1)

union

select top 500

1 as SortOrder,

ag.AgencyID as Value,

'[' + ag.AgencyCode + '] - ' + ag.AgencyName as Label

from Atlas.dbo.Agency as ag

inner join #Words as wo

on ag.AgencyCode like wo.Word

or ag.AgencyName like wo.Word

where ag.Active = isnull(@ActiveOnly, 1)

end

select r.Value,

r.Label

from #Results as r

order by r.SortOrder,

r.Label

if object\_id('tempdb.dbo.#Results') is not null drop table #Results

if object\_id('tempdb.dbo.#Words') is not null drop table #Words

end

else

begin

begin with

WithSortOrder as (

select 0 as SortOrder,

-re.RepID as Value,

re.FirstName + ' ' + re.LastName as Label

from Marketing.dbo.Representatives as re

where re.Active = 1

union

select 1 as SortOrder,

-(at.TierID + 500) as Value,

at.Tier + ' Tier' as Label

from Atlas.dbo.AgencyTier as at

union

select 1 as SortOrder,

-500 as Value,

'No Tier' as Label

-- union

--select 2 as SortOrder,

-- ag.AgencyID as Value,

-- '[' + ag.AgencyCode + '] - ' + ag.AgencyName as Label

--from Atlas.dbo.Agency as ag

--where ag.Active = 1

)

select wso.Value,

wso.Label

from WithSortOrder as wso

order by wso.SortOrder,

wso.Label

end

end

end

**NEW ABOVE**

**UNIVERSAL SSRS EXPRESSIONS**

**FORMULA for AVERAGES**

=Sum(Fields!FastTrackAverageYTD.Value \* (Fields!FastTrackYTD.Value)) / SUM(Fields!FastTrackYTD.Value)

--This bottom/Denominator section gives us the count = dollars/count

--This codes also accommodates for divide by zero errors by calling the Quotient function

=Code.Quotient(Sum(Fields!AverageYTD.Value \* (Fields!TotalYTD.Value - Fields!FastTrackYTD.Value)),Sum(Fields!TotalYTD.Value - Fields!FastTrackYTD.Value),0)

* ="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)

**YTD**

=DateAdd("d",-DatePart(DateInterval.DayOfYear,Today,0,0)+1,Today)

**MTH**

=DateSerial( year(today()), month(today()), 1)

**WTD**

=DateAdd("d",-DatePart(DateInterval.WeekDay,Today,0,0)+1,Today)

**DONs version**

select @FirstOfYear = dateadd(day, -datepart(dayofyear, @AsOfDate) + 1, @AsOfDate),

@FirstOfMonth = dateadd(day, -datepart(day, @AsOfDate) + 1, @AsOfDate),

@FirstOfWeek = dateadd(day, -datepart(dw, @AsOfDate) + 1, @AsOfDate),

@AsOfDate = dateadd(millisecond, -3, dateadd(day, 1, @AsOfDate))

where ca.CreatedDate

between @FirstOfYear

and @AsOfDate

=IIF(Fields!InsuredName.Value="Balance Forward" OR Fields!InsuredName.Value="Debit Memo" OR Fields!InsuredName.Value="Credit Memo","",FormatCurrency(Fields!Commission.Value/Fields!Rate.Value,2))

=IIF(Fields!Earned.Value OR Fields!InsuredName.Value="Balance Forward" OR Fields!InsuredName.Value="Debit Memo" OR Fields!InsuredName.Value="Credit Memo","",FormatCurrency(Fields!Cash.Value,2))

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

=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)

= Parameters!CompanyID.Label + vbcrlf

+ "Commission Statement" + vbcrlf

+ "For The Month of " + FormatDateTime(Parameters!MonthEnd.Value,2)

=IIF(CBOOL(Fields!Earned.Value),"E","")

=Join(Parameters!RepType.Label,", ")

=IIF(Parameters!County.Count=CountRows ("Counties"), "All", Join(Parameters!County.Label,", "))

Date Formulas: Find First Day of Previous Month:

DateAdd("m", -1, DateSerial(Year(Today()), Month(Today()), 1))

Find Last Day of Previous Month:

DateSerial(Year(Today()), Month(Today()), 0)

Find First Day of Current Month:

DateSerial(Year(Today()),Month(Today()),1)

Find Last Day of Current Month:

DateSerial(Year(Today()),Month(DateAdd("m", 1, Today())),0)

=IIF(Parameters!MonthEnd.Value=CDATE("6/30/2010"),Code.SpecialMessage(Parameters!AgencyCode.Value),IIF(Parameters!MonthEnd.Value=CDATE("11/30/2010"),Code.SpecialMessage2(Parameters!AgencyCode.Value),""))

**HIDE a ROW based on a CONDITION**

Go to row visability choose Show or Hide based on an expression.

=iif(len(Fields!Jan.Value) = 0 AND len(Fields!Feb.Value) = 0 AND len(Fields!Mar.Value) = 0,True,False)

=iif(len(Fields!Mar.Value) = 0,True,False)

**=Iif(IsNothing(Fields!TASouthYTD.Value),True,False)**

=Iif(IsNothing(ReportItems!Textbox1.Value),True,False)

=iif((Sum(IIf(Fields!Brand.Value = "Outfitters", Fields!Quantity.Value, Nothing))) is nothing ,True,False)

**THIS HIDES DUPLICATE ROWS IN ROW VISABILITY**

iif(Fields!Storm.Value = Previous(Fields!Storm.Value), True, False)

**One Parameter is dependent upon another Parameter (Cascading Parameters)**

Count is dependent upon State being populated

SELECT DISTINCT CountyID,[StateID], CountyDescription

FROM [dbo].[County]

WHERE [StateID] IN (@State)

**SSRS DATASET QUERIES for DROP DOWNS**

ALTER procedure [dbo].[reportGetAdjustingCompanies]

as

begin

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) --UAC, UPCIC, vern's conversion crap

end

or just choose “ALLOW MULTIPLE VALUES”

WHERE p.StateID IN (SELECT StringValue FROM dbo.Split(@StateID,','))

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

**WRITING CUSTOM CODE**

function ConvertLossType(IsLAE As Boolean, IsAO as Boolean, Payee1 as string) AS string

dim LossType as string

if not IsLAE then

LossType="L"

else

if not IsAO then

LossType="A"

else

if StrComp(Payee1, "Universal Adjusting Corporation") = 0 then

LossType = "U"

else

LossType = "O"

end if

end if

end if

return LossType

End Function 'Convert LossType

**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**

**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

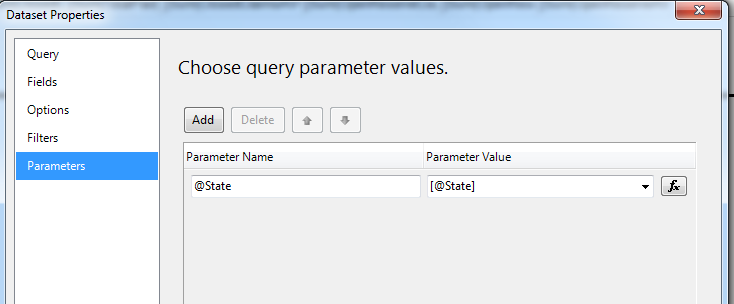
WHERE S.[StateID] IN (15,25,30,13,32,42,67,36,35,17,21)

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

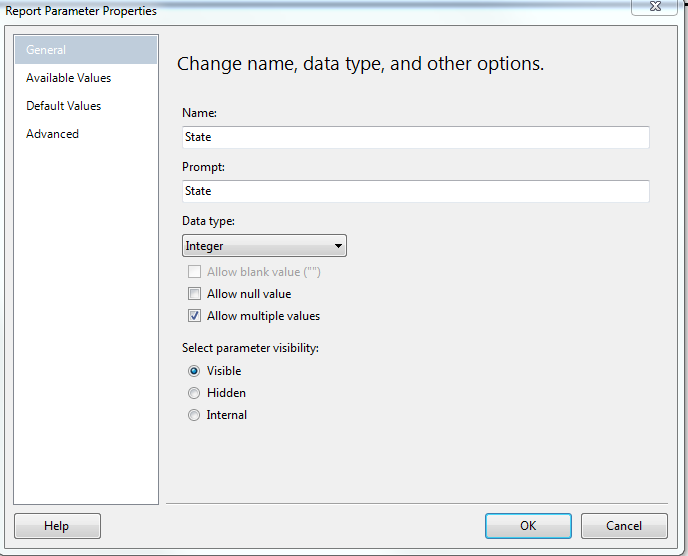
SELECT DISTINCT CountyID,[StateID], CountyDescription

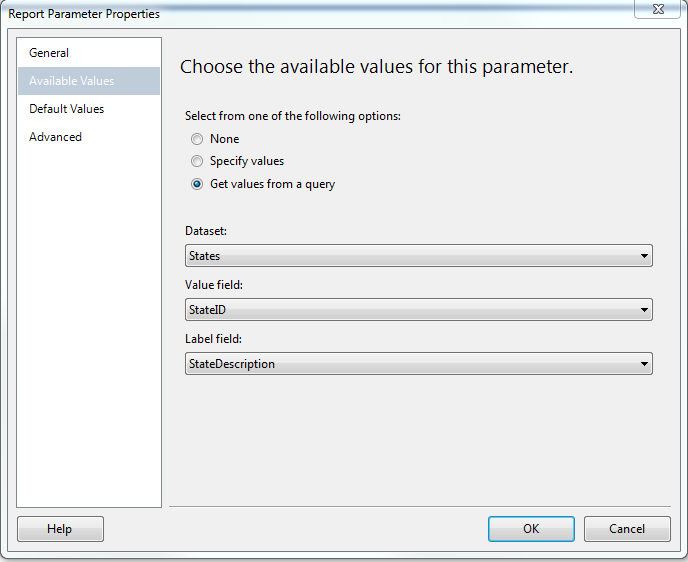
FROM [dbo].[County]

WHERE [StateID] IN (@State)



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

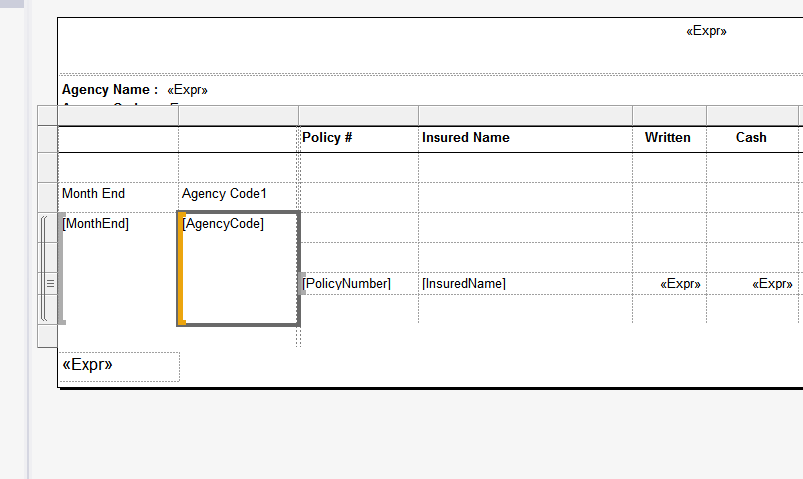




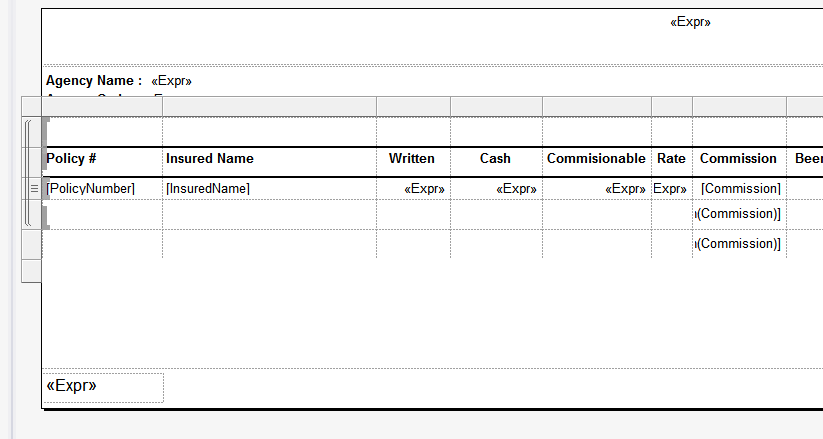
**\*\*\*HOW TO SHOW THE COLUMNS WE GROUP BY TO SHOW ABOVE ALL THE OTHER HEADER NAMES IN ITS OWN SECTION BUT STILL BE PART OF THE GROUP\*\*\***

**This will save space on the length of the report and allow us to present more columns**

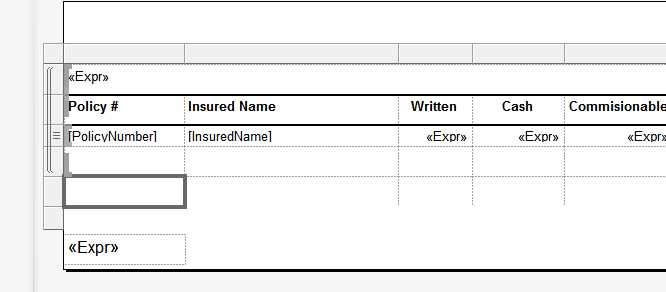
* Right click on far left field – choose “Add Group” – “Row Group” – “Parent Group”
* Repeat steps above for required groups
* Right click on details in bottom Row Groups section and click “Add Total” after
* Right click on **first group field on right** in Tablix choose “Inset Row” “Inside Group Above” – Repeat again.
* Now you should have added 2 blank rows above the details row and grouping should look like picture below.

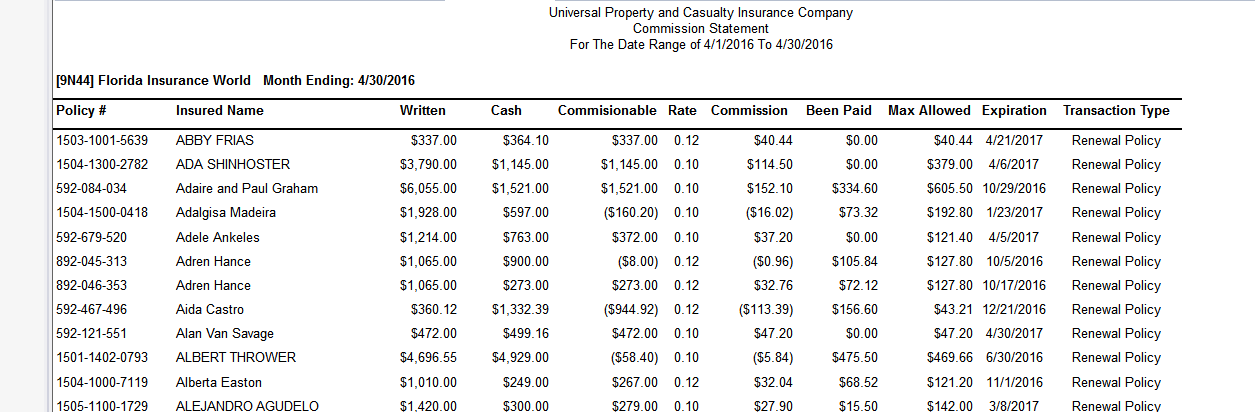


* Now delete the two grouped columns – Delete columns only not associated groupings
* Copy header from above and paste above the details row.
* Delete not needed rows and old header row.
* Now you should have one blank row above the header but still inside the group. See picture below



* Merge and center top blank row and insert an expression with the two fields you deleted below. Now the report should look like this below





**HOW TO GET THE HEADER TO REPEAT ON EACH PAGE**

* Click on the drop down arrow to the right of “Colum Groups” and choose advanced Mode
* Under “Row Groups” select the static row of the Header row and in the properties pane set
  + “Repeat on new page” = TRUE
  + “Keep with Group” = AFTER
* Right click to the left of the HEADER row and choose “Tablix Properties”
  + **Under the column Headers section put a check in the** 
    - Repeat header columns on each page
    - Keep header visible while scrolling

**HOW TO DEPLOY TO PDF AND NOT GET A BLANK PAGE EVERY OTHER PAGE**

* In report properties change the report to **landscape**

**HOW TO SORT ON A DYNAMICALLY CALCULATED TOTAL COUNT FILE FROM TOP TO BOTTOM**

**HOW TO SORT A FULL NAME COLUMN BASED ON THE LAST NAME.**

**HOW TO SORT ON THE ACTUAL MONTHS IN ORDER VERSES ALPHABETICAL ORDER.**

**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

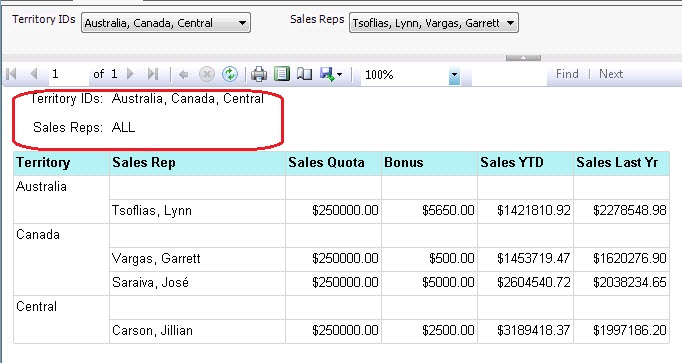
JOIN Person.Person AS P ON SP.BusinessEntityID = P.BusinessEntityID

SET NOCOUNT OFF

GO

### Step 4

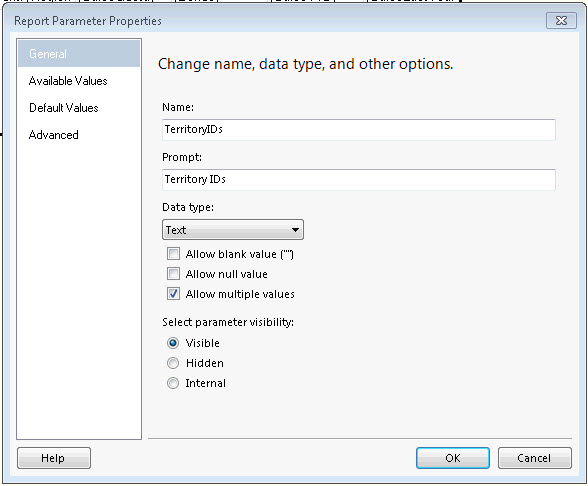
Add the three stored procedures that were created in steps 1 through 3 to a new report. Then create the layout for the report. My report is laid out by grouping the Sales Reps by Territory.

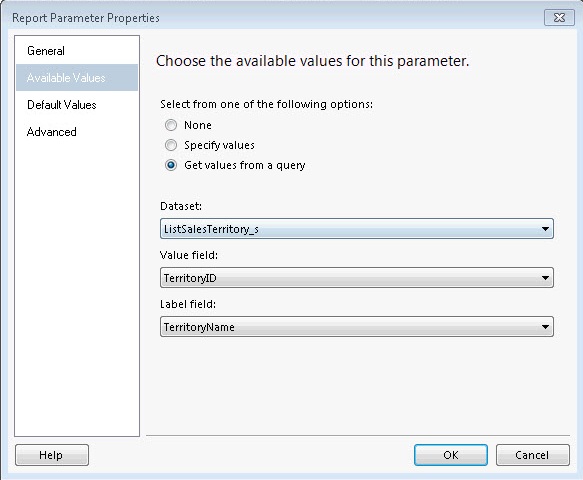
   
*Finished Report*

### 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.

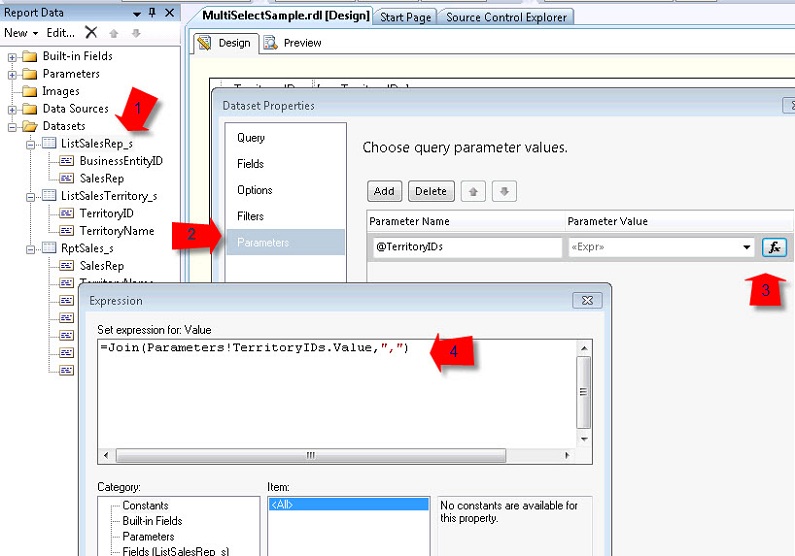




### Step 6

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.

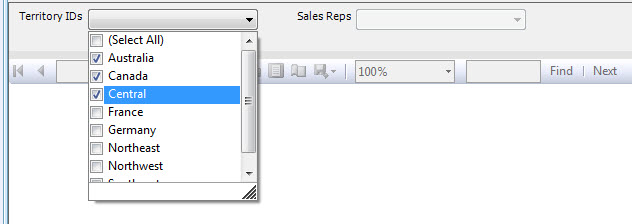
1. Right click on the stored procedure that needs the list of values and select *Dataset Properties* from the drop down list.
2. Select *Parameters* in the left hand list of the *Dataset Properties* dialog box.
3. In this example, the delimited list of Territories need to be assigned to the @TerritoryIDs 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.

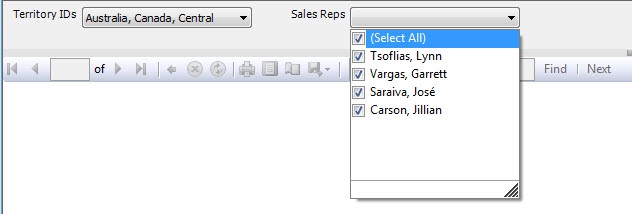


=Join(Parameters!TerritoryIDs.Value,",")

### 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.

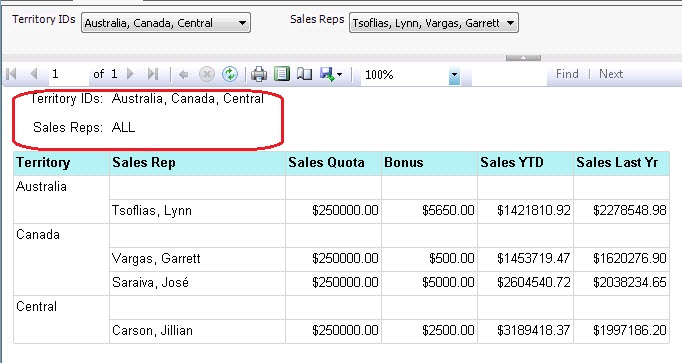




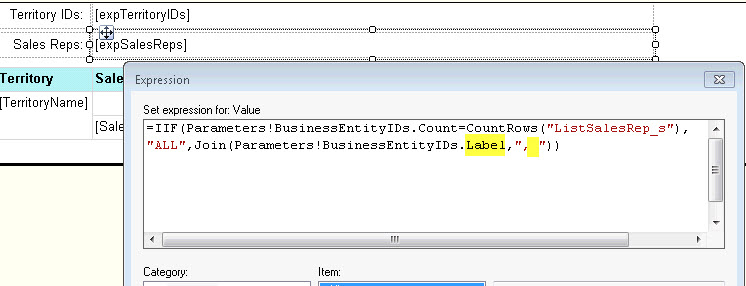
### BUT THAT'S NOT ALL...

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.



**DATA DRIVEN SUBSCRIPTIONS**

select cast(dateadd(day, -1, getdate()) as date) as AsOfDate,

cast(dateadd(day, -2, getdate()) as date) as CompareDate,

75000 as XWindLimit,

250000 as WindLimit,

cast('04/01/1998' as date) as LossStartDate,

cast(dateadd(day, -1, getdate()) as date) as LossEndDate,

cast(0 as bit) as PendingReserves

**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 +
* **Expression used if you do not want to display over 2,000 counties is (Counties is the name of the dataset)**=IIF(Parameters!County.Count=CountRows ("Counties"), "All", Join(Parameters!County.Label,", "))

**PAGE BREAKS**

To display row headers on multiple pages

1. Right-click the row, column, or corner handle of a tablix data region, and then click **Tablix Properties**.
2. In **Row Headers**, select **Repeat header rows on each page**.
3. Click OK.

To display column headers on multiple pages

1. Right-click the row, column, or corner handle of a tablix data region, and then click **Tablix Properties**.
2. In **Column Headers**, select **Repeat header columns on each page**.
3. Click OK.

To display a static tablix member (row or column) on multiple pages

1. On the design surface, click the row or column handle of the tablix data region to select it. The Grouping pane displays the row and column groups.
2. On the right side of the Grouping pane, click the down arrow, and then click **Advanced Mode**. The Row Groups pane displays the hierarchical static and dynamic members for the row groups hierarchy and the Column groups pane shows a similar display for the column groups hierarchy.
3. Click the static member that corresponds to the static member (row or column) that you want to remain visible while scrolling. The Properties pane displays the **Tablix Member** properties.

If you don’t see the Properties pane, click the **View** tab at the top of the Report Builder window and then click**Properties**.

1. In the Properties pane, set **RepeatOnNewPage** to True.
2. Set **KeepWithGroup** to After.
3. Repeat this for as many adjacent members as you want to repeat.
4. Preview the report.

**SSRS Problems / Solutions**

**Problem:**

The Total Claims Count was correct coming from the SP but not the SSRS report.

**Solution**:

Took all the code out of the SP and ran in a separate window after assigning the correct values to the parameters. The size of the SP input parameters State and County, were way too small and truncating the input values which resulted in lower numbers.

**SSRS DIVIDE BY ZERO ERROR**

Go to REPORT – REPORT PROPERTIES – CODE

Enter the following code.

Public Function Quotient(ByVal Numerator As Decimal, Denominator As Decimal, DefaultValue as Decimal) As Decimal

If Denominator = 0 Then

If Numerator = 0

Return 0

Else

Return DefaultValue

End If

Else

Return Numerator / Denominator

End If

End Function

An easy clean way to prevent a divide by zero error is using the report code area.

In the Menu; got to Report > Report Properties > Code and paste the code bellow

Public Function Quotient(ByVal numerator As Decimal, denominator As Decimal) As Decimal

If denominator = 0 Then

Return 0

Else

Return numerator / denominator

End If

End Function

To call the function go to the the Textbox expresion and type:

=Code.Quotient(SUM(fields!FieldName.Value),SUM(Fields!FieldName2.Value))

 in this case I am putting the formula at the Group level so I am using sum. Otherwise it would be:

=Code.Quotient(fields!FieldName.Value,Fields!FieldName2.Value)