©2008 Vision Harvest, Inc.

Crystal Reports 2008 Designer 3 Workshop

Vision Harvest, Inc.

888-236-8087
sales@visionharvest.com

Published: August 4, 2008
Part: VISI100072S

©2008 Vision Harvest, Inc.

ALL RIGHTS RESERVED

This course covers Crystal Reports® 2008

No part of this manual may be copied, photocopied, or reproduced in any form or by any means without permission in writing from the Author - Vision Harvest, Inc. All other trademarks, service marks, products or services are trademarks or registered trademarks of their respective holders.

This course and all materials supplied to the student are designed to familiarize the student with the operation of the software programs. We urge each student to review the manuals provided by the software publisher regarding specific questions as to the operation of the programs. THERE ARE NO WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, MADE WITH RESPECT TO THESE MATERIALS OR ANY OTHER INFORMATION PROVIDED TO THE STUDENT. ANY SIMILARITIES BETWEEN FICTITIOUS COMPANIES, THEIR DOMAIN NAMES, OR PERSONS WITH REAL COMPANIES OR PERSONS IS PURELY COINCIDENTAL AND IS NOT INTENDED TO PROMOTE, ENDORSE OR REFER TO SUCH EXISTING COMPANIES OR PERSONS.

VISION HARVEST, INC. NOR ITS RESPECTIVE PRODUCTS OR SERVICES ARE AFFILIATED WITH, OR ENDORSED, LICENSED, OR SPONSORED BY BUSINESS OBJECTS, THE OWNER OF CRYSTAL REPORTS.

This courseware has been developed by the professional team at Vision Harvest, Inc., www.visionharvest.com. Questions, comments or concerns about courseware content should be sent to sales@visionharvest.com or 888.236.8087.

Table of Contents

©2008 Vision Harvest, Inc.

Introduction

Introduction Objectives

This manual is written to give you a step-by-step guide for your classroom training and a handy reference for your daily work. In this Introduction, you will learn how to use this training guide effectively. This section covers the following topics:

- An introduction to the Crystal Reports application
- Pre-Qualification Exercise
- Class objectives
- Help with using this training guide
- Information on how to start the program

About Crystal Reports

In today's information intensive environment, every business has a database of some sort. After all, business today is all about information and databases give you a handle on the massive amounts of information you must deal with. Therefore, your business has a database and from that database, you need reports. The problem is, most reporting capabilities that come with database programs are limited. They only report on data from that program. Many users need to report on data from multiple sources, even databases such as Oracle, Microsoft SQL Server, DB2 or Sybase.

Crystal Reports is one of the most powerful reporting programs available with the ability to pull data from all types of data sources. You can use Crystal Reports to generate reports from any of the standard PC database programs; Access, Paradox, or FoxPro, as well as from a mainframe or server database. Crystal also has a powerful web-reporting server that allows you to distribute your reports over the web.

Crystal Reports is bundled with more than 160 other programs including Visual Basic, some medical applications, many accounting packages and several ERP solutions. It makes report generation easy without requiring you to be a programmer or a database expert. If you know how to work in a Windows environment and are familiar with the data you want to use, you can create a Crystal Report that looks professional and makes sense.

Training Philosophy

Studies show that people retain 10% of information they see, 20% of information they hear, 50% of what they see and hear, and 80% of what they see, hear and do. In line with this concept, the class utilizes a hands-on method of training. You will see the effects of new procedures on the screen, hear the instructor explain how and why to use features, and perform the actions yourself as you learn.

In addition, this class focuses on your ability to perform tasks using the most productive techniques. The manual may contain several methods of accomplishing a certain task. However, class time does not allow for practice of all methods for each task. Your instructor will guide you in the most effective method of performing a task, but inform you of other methods that are available.

Questions are encouraged. While we give our best effort to explain new concepts in understandable terms, you may need to hear the concept again or have it explained more thoroughly. Please let the instructor know when you need more information!

Class Objectives

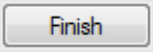
This class is a performance based instructional system. It is geared to provide you with the tools you need to build and distribute reports the quickest, most efficient way. After completing this course, you will be able to perform the following tasks:

- ❖ Create and incorporate custom functions into formulas
- ❖ Build high level advanced formulas with looping statements and case select
- ❖ Use SQL statements to build Crystal Reports SQL Expressions and Commands
- ❖ Apply advanced database concepts with views and stored procedures
- ❖ Performance fine tune reports for shorter run times
- ❖ Build advanced Cross-Tabs using the new Crystal Reports 2008 advanced Cross-Tab features
- ❖ Apply everyday tips and tricks into corporate reports

About This Manual

Each section of this manual contains objectives to provide you with the overall goals for the lesson. Lessons have descriptions of features and concepts followed by systematic directions for completing a specific task. Each section ends with a challenge exercise to help you practice the skills you learned in the lesson. Challenge exercises provide you with tasks to accomplish. Try to complete these exercises on your own.

As you work in this Training Guide, certain conventions are used to identify specific procedures. Use the following table as a guide:

Training Guide Conventions	
Item	**Illustrated As**
Menu Commands	Underlined letters for accessing menu commands are shown: Example: File/Open
Command Buttons	Command Buttons in dialog boxes are shown as buttons: Example: 
Categories, Radio Buttons, Text Boxes, Check Boxes	All options within dialog boxes are listed in italicized text: Example: the *Keep Group Together* check box the *Other* radio button
Keystrokes	Keyboard keys are indicated by uppercase text: Example: press ENTER Keyboard combinations are shown in uppercase text with a plus sign (+) between the keys that need to be pressed simultaneously. Example: press CTRL + S to save
Toolbar Buttons	Toolbar buttons are indicated by the button name and a graphic image of the button: Example: click the Print Preview  button
Typing or File Selections	Text to be typed or file names to be selected are printed in bold letters: Example: type **Henry** select **grouping.rpt**
Exercises	Step-by-Step exercises in the text are indicated by bold text and the ❖ symbol. For example: ❖ **Exercise - Format Objects**

Tips, Notes, and Warnings

Tips, notes and warnings display with the following icons. Text for these additional comments display in bold and italics as shown below:

This icon indicates a tip or shortcut.

This icon points out a note of additional information.

This icon calls attention to a warning or very important note

©2008 Vision Harvest, Inc.

NOTES

©2008 Vision Harvest, Inc.

Lesson 1
Refresher Exercise

Lesson Objectives

After completing this lesson, you will be able to:

- **Plan and Create a report from a set of business requirements**
  Review the techniques needed to plan and create a report
- **Add multiple groups to reports**
  Create groups within groups to subdivide your reports how you want
- **Conditionally format your report**
  Make your report more informative and attractive with formatting
- **Parameterize your report**
  Use parameter fields to provide flexibility to the user as to the data they are viewing
- **Add multiple charts to a report**
  Use more then one chart and conditional formatting to graphically display data for different groups

Review of Planning a Report

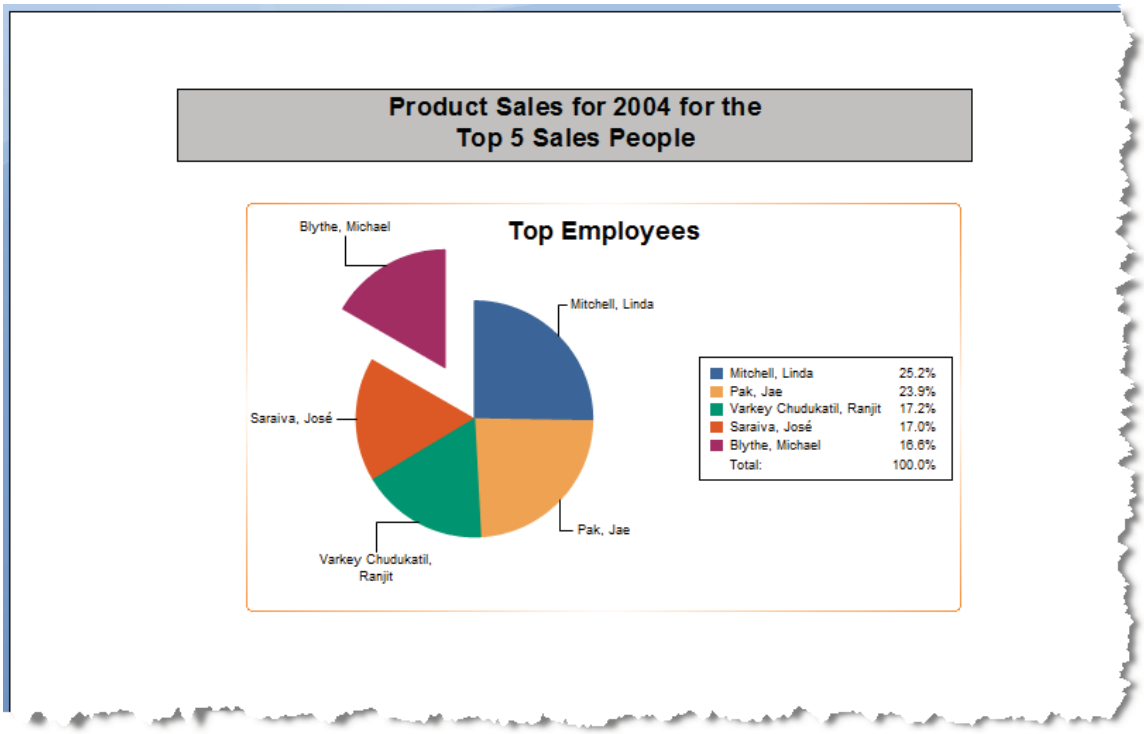
To make the report creation process efficient, you must plan the report. A little work up front before you ever open Crystal Reports can save a lot of time putting the report together. It can also save problems caused by adding more tables to the report that can change the number of records displayed.

You must look at the business requirements for the report and decide what fields you need and the tables in which they are located. It is also helpful to look at things such as groupings, whether you need Cross-Tabs, fields needed for formulas and selection criteria.

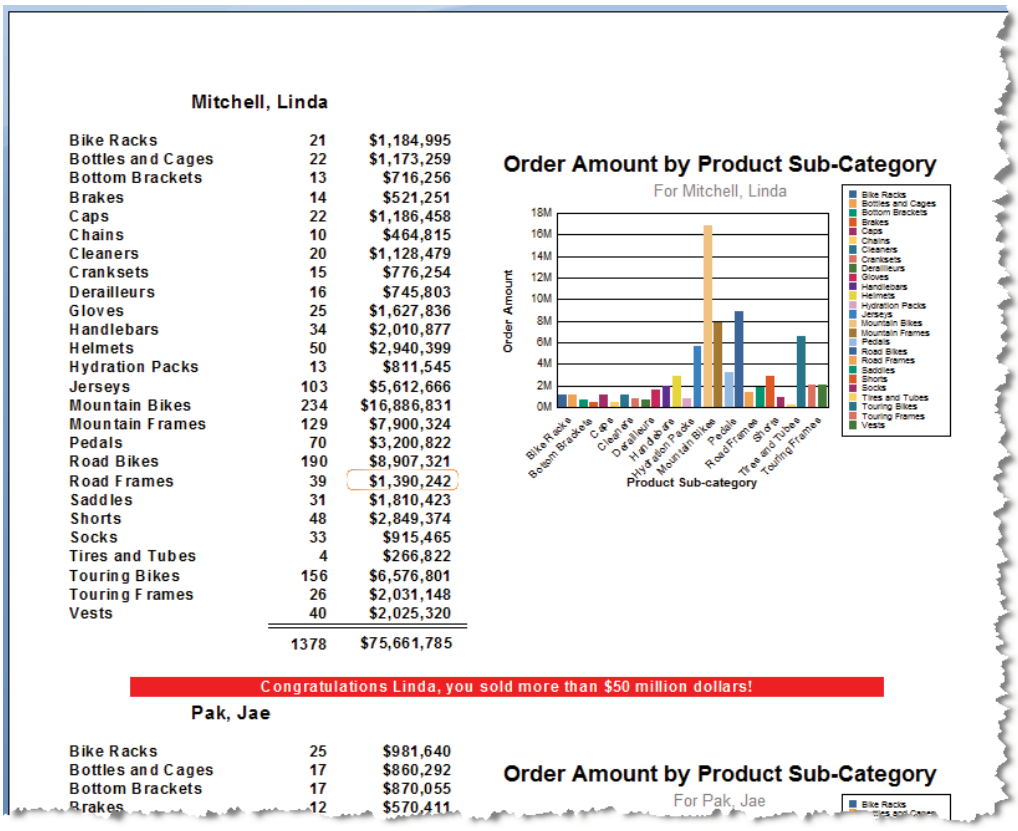
Suppose someone asks you to create a report based on the AdventureWorks database. Below are the business requirements for the report:

- The first page of the report needs to show the top N sales people by product sales; N needs to be a user defined variable.
- The body of the report shows the sales detail for each sales person showing the number of orders and amount by product sub-category. Beside each sales person should be a bar chart showing order amount by product sub-category for that sales person.
- The user needs to be able to choose a year for the report with a default value of 2004.
- Need to be able to see quickly any sales person with over $50,000,000 of sales
- The person requesting the report has supplied you with a printout of what the report should look like, both the first page and detailed pages. See the samples on the next page.

The screen shot below is a view of the first page of the report:



Page 2 of the report should look as follows:



To create this report, first determine which fields you need.

- For the Detail section, you need the SalesOrderID and SubTotal fields
- For grouping, you need the SalesPersonID and the Product Sub-category Name fields
- Finally, you need to select orders based on the OrderDate field

Next, decide which tables you need to add to get the required fields.

- The SalesOrderID, SubTotal and OrderDate fields are in the SalesOrderHeader table
- The SalesPersonID field is in the SalesPerson table; Sales Person Name is in the Contact table via the Employee table
- The Product Sub-category Name field is in the ProductSubcategory table. To get to this table you need the SalesOrderDetail and Product tables

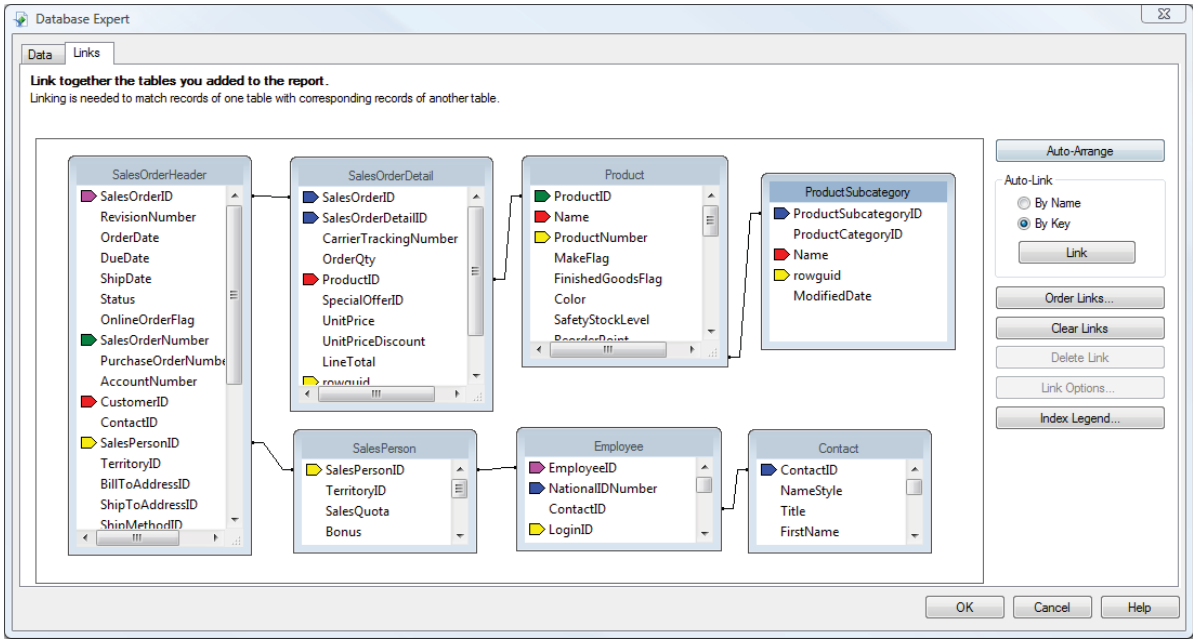
The order in which the tables are linked is usually very important from a performance viewpoint. When using the Smart Linking feature provided by Crystal Reports, the tables will be linked automatically in the most logical performance-driven fashion, but this is only true in Crystal Reports 2008.

Crystal Reports 2008 attempts to link tables together where the "To" table contains a primary key (indexed, no duplicates). The reason is to improve database performance and in most instances is the best choice to reduce database querying time. In some cases this is not the best choice and it is fairly easy to override the automatic links and create your own.

This means you can simply add the tables in any order and let the Smart Linking feature do the rest of the work.

Creating the Report

Since you have given this some thought and determined the required tables, it is time to create the report. Once the tables are added, Smart Linking will create the links (joins) similar to the following example:



NOTE: Remember you can add multiple tables to a report from the Database Expert. You work with links using the Links tab on the Database Expert. The Database Expert is accessed under the Database menu or by selecting the Database Expert button on the Experts toolbar.

Placing Fields on the Report

Remember you have three ways of placing fields on a report. Select the field in the Field Explorer, then:

- Click the Insert button, and then click in the report
- Drag the field to the report
- Right mouse click, choose Insert to Report from the short cut menu, and then click in the report

When you place a field in the Details section, Crystal Reports adds a detail field title in the Page Header section and aligns the field and field title with a guideline. You can drag the guideline marker in the ruler to move the field and its title together.

Now that you have planned the report, you are ready to use Crystal Reports to create the report.

NOTE: Your instructor will provide the database connection entries if they differ from the standard entries in Exercise 1.0.

❖ Exercise 1.0 - Begin the Monthly Sales by Supplier and Category Report and Link the Tables Needed

1. Start a new report as a blank report. Open the **Create New Connection** folder then the **OLE DB (ADO)** data sources; the OLE DB (ADO) dialog will open.

2. In the **OLE DB Provider** section, select **SQL Native Client**, and then click **Next**. The **Connection Information** section appears.

3. Enter **localhost\SQLEXPRESS** in the *Server:* field, **sa** in the *User ID:* field and **cr2008** in the *Password:* field (the password is case sensitive). Select **AdventureWorks** from the *Database:* drop-down field. Be sure that the *Integrated Security:* checkbox is not checked. Click **Finish**.

4. Add the tables indicated by the business requirements. *(SalesOrderHeader, SalesOrderDetail, SalesPerson, Employee, Contact, Product and ProductSubcategory)*

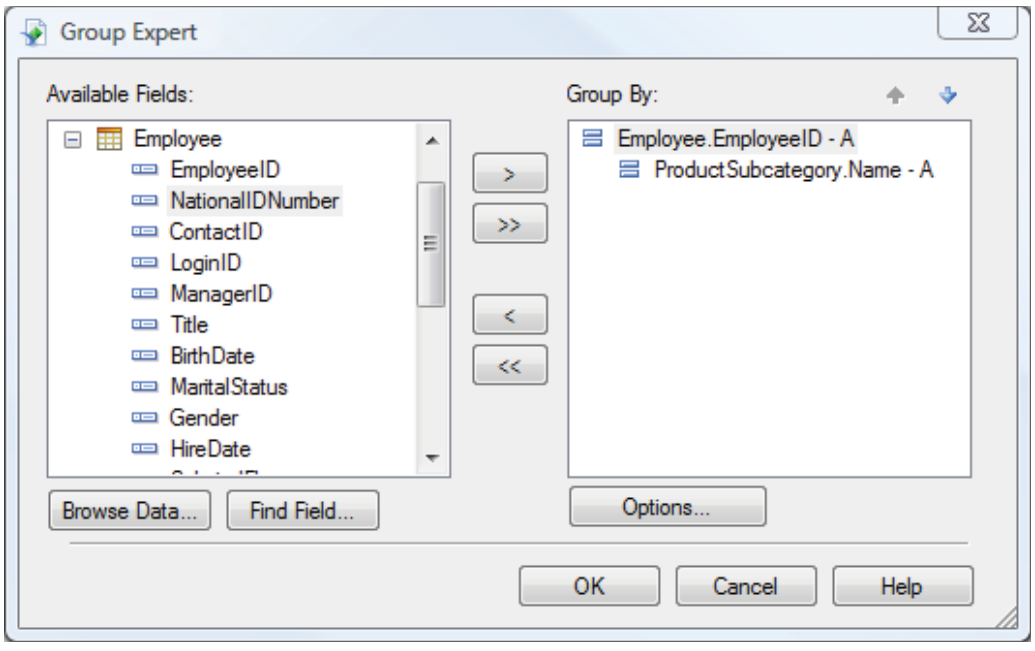
5. The Auto-Link feature should link the tables automatically. Clear the links and do them yourself, but create the links to match the example on the previous page.

6. Add the appropriate fields to the **Details** section *(SalesOrderID, SubTotal)*

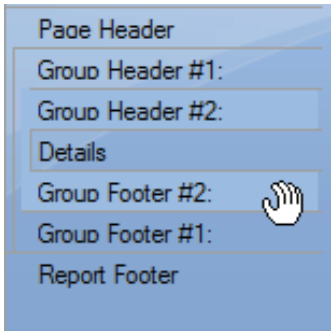
7. Change the Printer Setup to a **Portrait** page layout, if necessary. Change the margins to be **.5"** on all four sides

Advanced Grouping

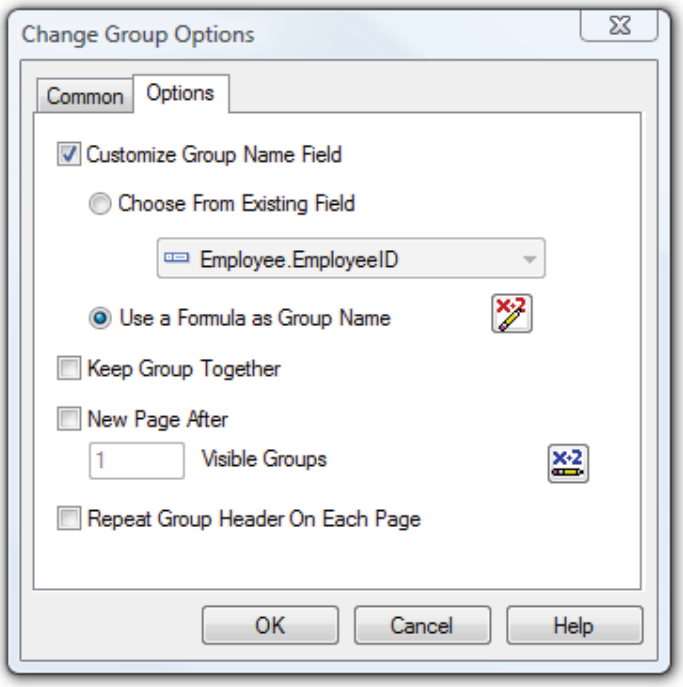
You can group data by using the *Insert/Group…* command or by clicking the Insert Group button on the Insert toolbar or the Group Expert button on the Experts toolbar. This report requires a Product Sub-category Name group nested inside a Contact (Employee) Name group as shown in the Group Expert illustration below:



As you create groups, Crystal Reports nests each group inside the previous one you created. However, you can reorder groups in any order you want. To reorder groups, left-click the Group Header section name at the left of the Design screen for the group you want to move. The group header and footer sections are highlighted. Drag the group to the new location. The mouse pointer changes to a grabbing hand as you drag. In the Group Expert shown above you simply move the group headers using the arrows in the top right of the window:



Remember that you can group on a field but display a formula or another field name as the group name field. When you group on Employee Id using the option shown below (Using a Formula as Group Name) you can display the employee name:



Also remember that you can summarize any field by RIGHT clicking it, then choosing *Insert Summary…* You must specify the type of calculation you want Crystal Reports to perform and the group at which you want the summary to appear.

❖ Exercise 1.1 – Create groups and summarize the SalesOrderID and SubTotal field

1. Open the **Group Expert** window using the Group Expert button. Create a group on the **ProductSubcategory.Name** field.

2. Create a second group on the **Employee.EmployeeID** field. Choose to *Use a formula as Group Name*, displaying the Employee Name as Last Name, First Name (from the **Contact** table).

3. The report requirements call for the report to display information by **EmployeeID** and then **Product Sub-category Name**, so you need to change the order of the groups. Click the Group Expert button. Select the **Product Sub-category Name** field in the *Group By:* list and then click the Down Arrow to reorder the groups
The EmployeeID should be the first group in the Group Expert with the Product Sub-category Name nested underneath.

4. Create a summary for the **SalesOrderID** and **SubTotal** columns. Do this by checking on the *Add to all group levels* option and then choose **OK** from the Insert Summary dialog
All the summary fields should automatically line up on the guideline to which the field is attached.

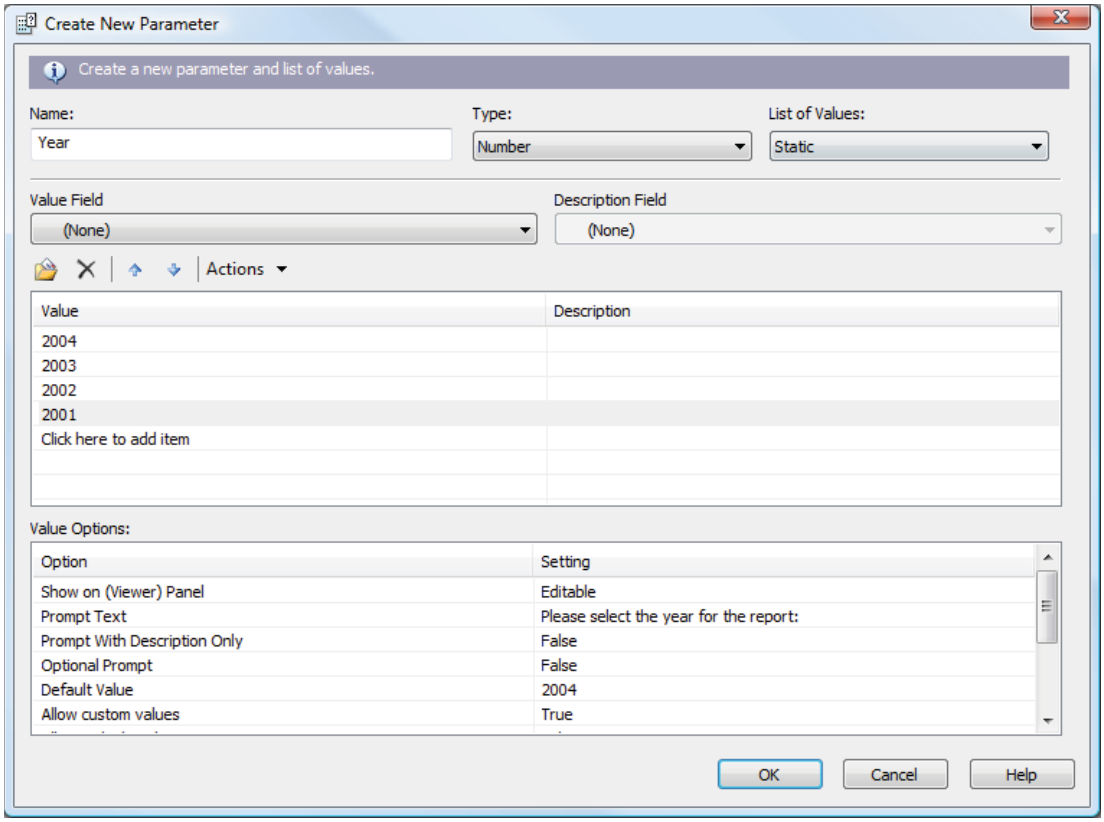
5. Move the summaries in **Group Footer #2** to **Group Header #2**, and then hide the **Details** and **Group Footer #2** sections.

Selecting Certain Records

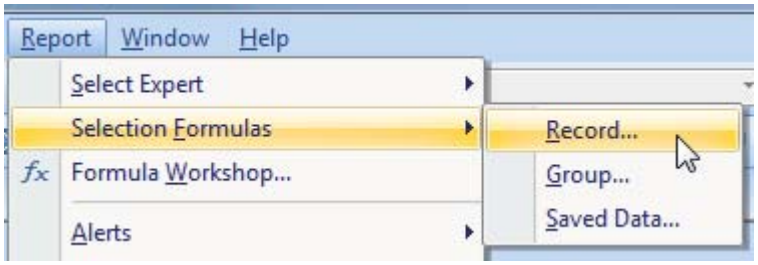
Remember Crystal Reports pulls all records from the database unless you filter the records based on some criteria. The business requirements for this report requests only records from a user selected year. First you need to create a parameter field and then use it in the record selection formula.

❖ Exercise 1.2 – Select Records for a Specific Year

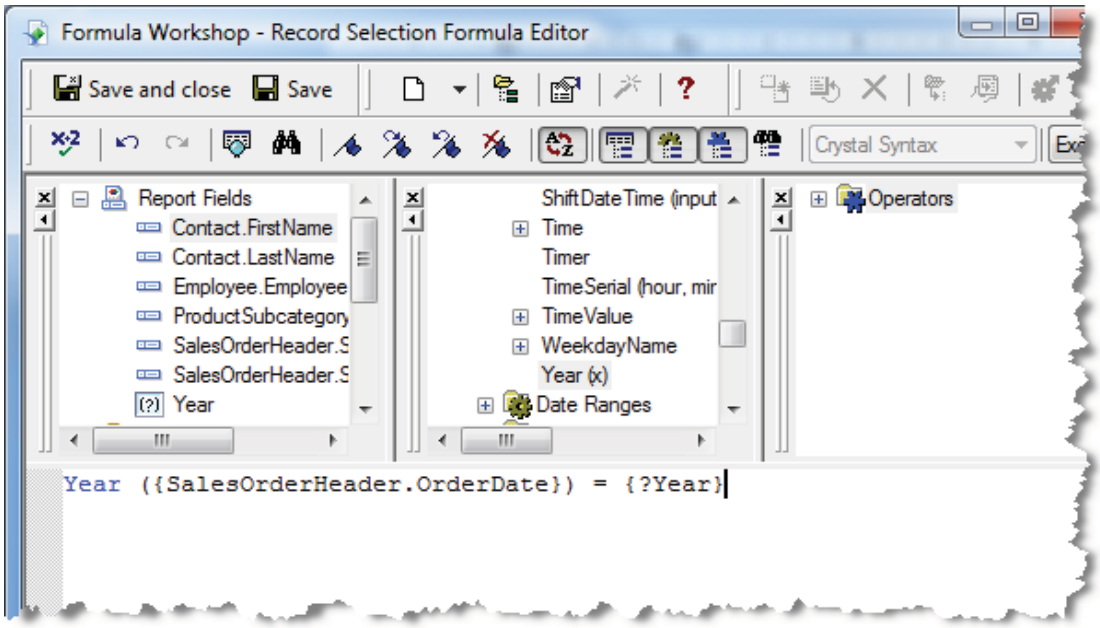
1. Create a parameter field called **Year** with a **Number** data type and a default value of **2004** as shown in the dialog below:



2. Under the **Report Menu** choose to edit the record selection formula as shown in the menu below:



3. In the formula editor you need to enter the following formula:



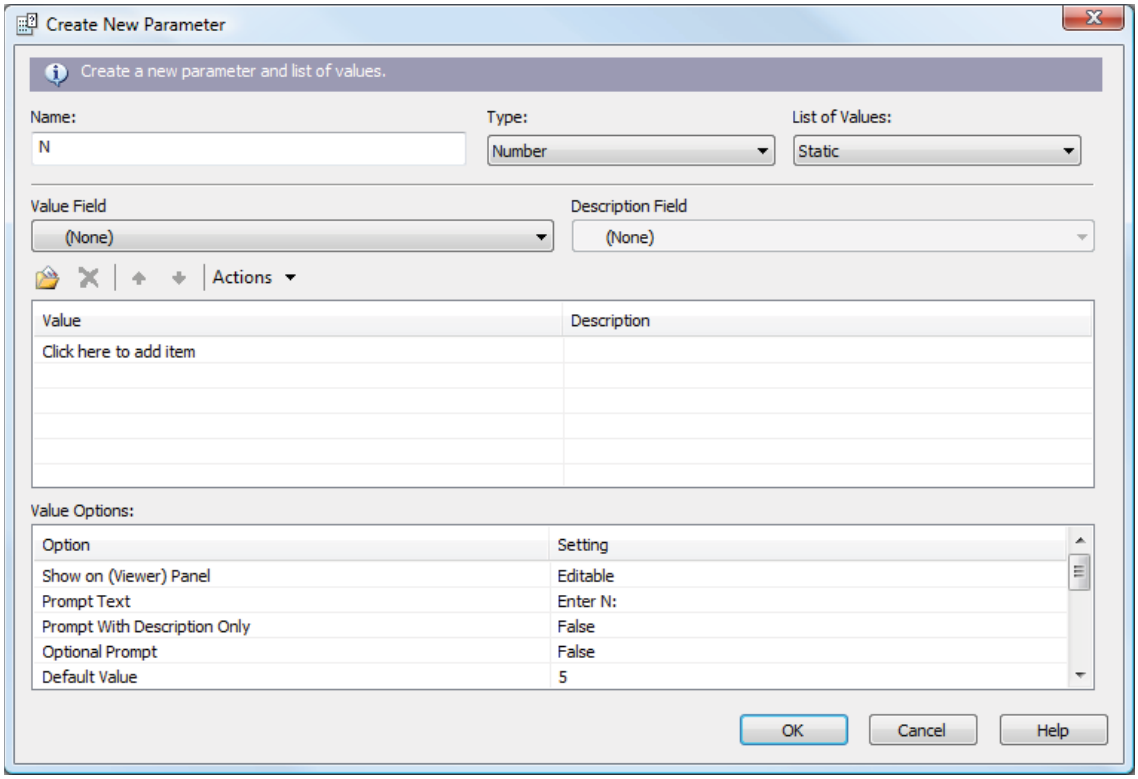
4. At this point save and preview the report. Name the report: **Refresher.rpt**

Group Sorting

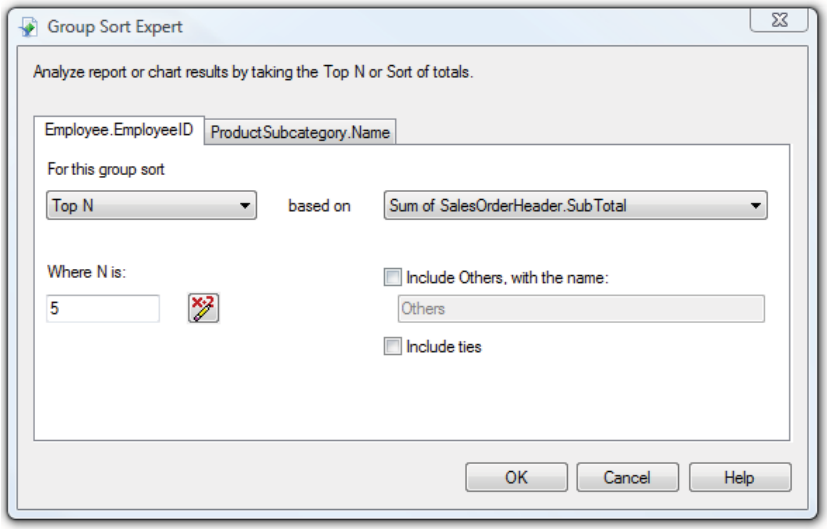
An additional requirement of this report is group sorting. Now you have the group summary fields you can choose how to sort the groups. One of the requirements outlined is the ability to look at the Top N sales people and top is measured by order amount. It is also a requirement to be able to select the value of N at run time.

❖ Exercise 1.3 - Sort Groups

1. Before you start sorting your groups you need to create a parameter field for **N** as the N value needs to be user defined. Create a simple parameter field with a default value of **5** as shown below:



2. Click on the **Group Sort Expert** and you will be able to select the type of sort for your group as shown below. The Group Sort Expert has a tab for each group on the report and within that tab you can choose the summary field on which you want to base your sort. You then decide which type of sort (Top N, Bottom N, All Sort etc) that you would like to do.



3. In this case you want to sort on **EmployeeID**; do a **Top N** sort based on **SubTotal** and exclude *Others*.

4. In order for **N** to be user defined you need to click on the conditional box (next to *Where N is:*). Simply add the parameter field **{?N}** to the formula box.

5. Preview your report and select values for both parameter fields. Save your report.

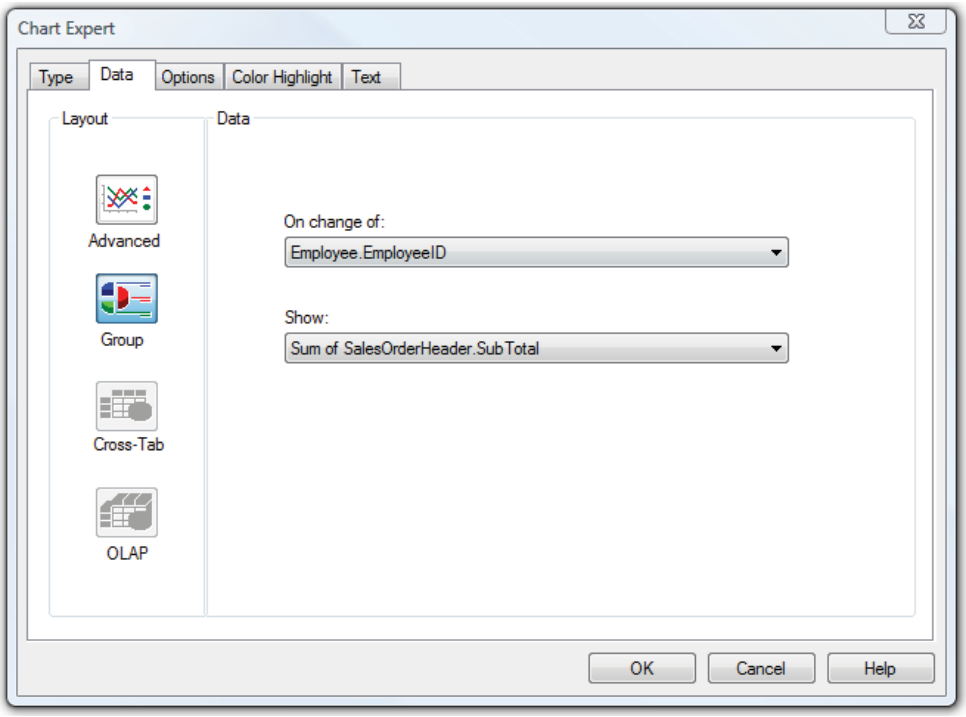
The Report Header

Now the report is providing the correct information before you begin formatting you should think about the business requirements for the report header. The report needs a pie chart on the front page with a text title that contains the parameter values for the report.

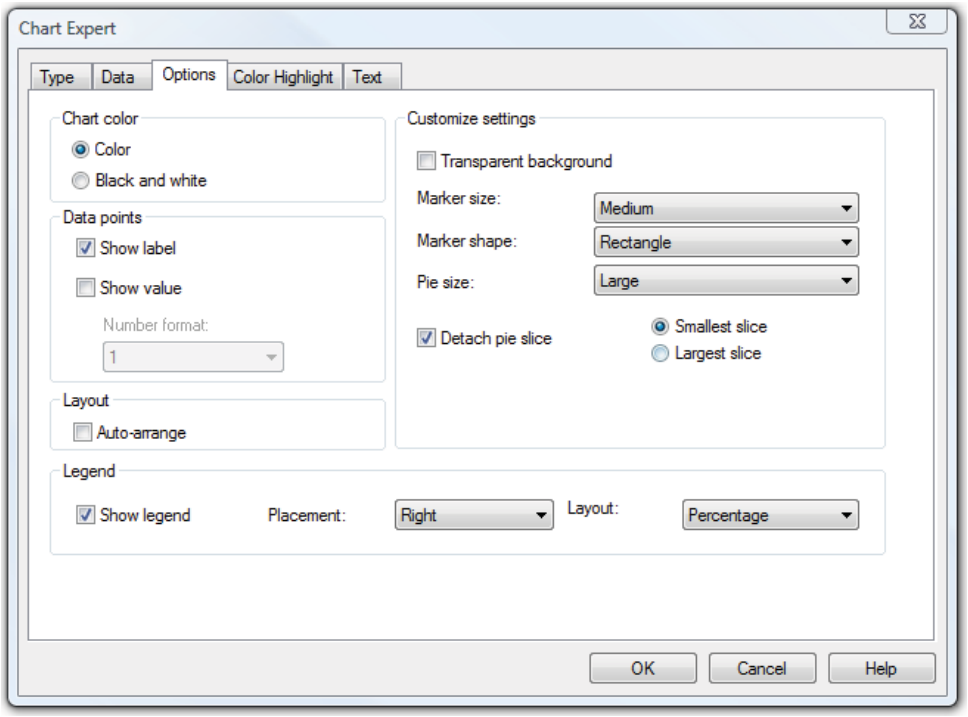
❖ Exercise 1.4 – Report Header

1. Insert a text object into the report header with an appropriate title for the report. Make sure you insert both parameter fields so their values at runtime are displayed. Format the title to stand out.

2. Insert a pie chart into the **Report Header** by clicking on **Insert Chart**; Using the **Chart Expert**, choose a pie chart and then make sure the correct data is selected as shown below:



3. Under **Options** tab you should choose to **show a label** and also **detach the largest piece of the pie** as shown below. Also uncheck the *Auto-arrange* option.



4. Add the title **Top Employees** to the chart, and format as you would like.

5. Finally, in the **Section Expert** choose to format the **Report Header** with a new page after (on the **Paging** tab) to make sure the **Report Header** is the front page of the report.

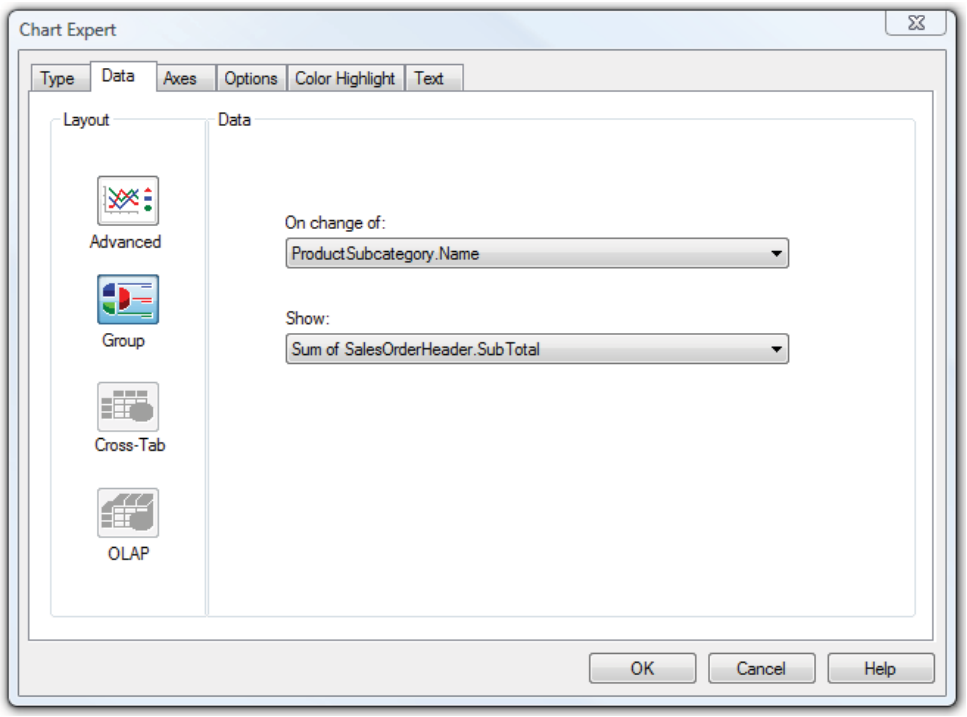
6. Save the report.

The Group Chart

As well as the chart in the report header there is also a chart in the main report for each employee. This chart is showing the product sub-categories sold for an employee.

❖ Exercise 1.5 – Group Chart

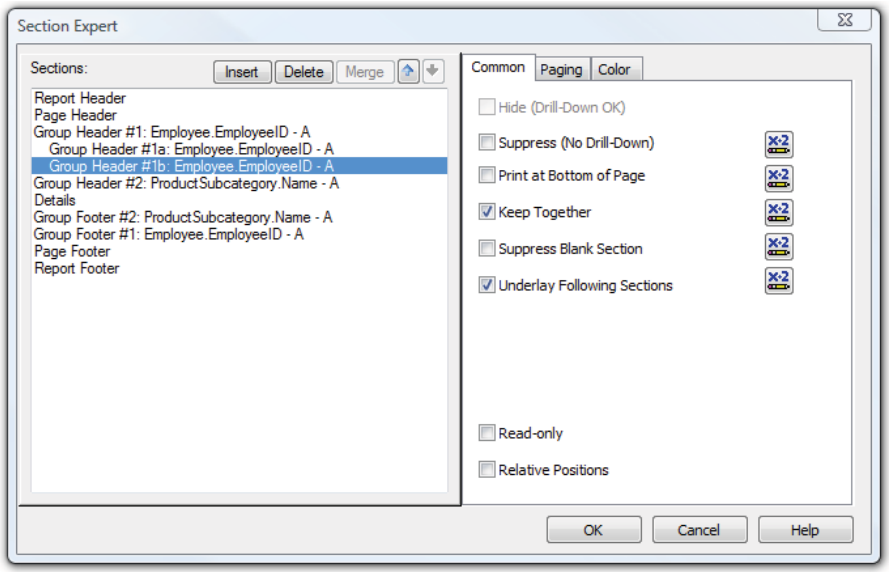
1. Click on the **Insert Chart** button and place the chart into **Group Header 1**. Open the **Chart Expert** and choose a bar chart; click on the **Data** tab to choose *SubTotal by ProductSubcategory.Name* as shown below:



An additional requirement is to have the chart beside the detailed data for the sales person. To do this, use will use the Underlay option in the section expert. However, you need to put the chart in its own section to prevent the group title from being underlayed as well.

2. Insert a new group header section for **Group 1**. Move the chart to **Group Header 1b**. Reformat the **Group Header 1a** to remove the excess white space.
Hint: use Fit Section.

3. In the Section Expert select **Group Header 1b** and choose to underlay the following section:



4. Save the report.

Your report is almost complete but needs further formatting to ensure any sales person who sold over $50,000,000 in a year needs to be recognized. You can achieve this with conditional formatting on the employee group footer adding a message when the sales are over $50,000,000.

❖ Exercise 1.6 - Conditionally Format the Report

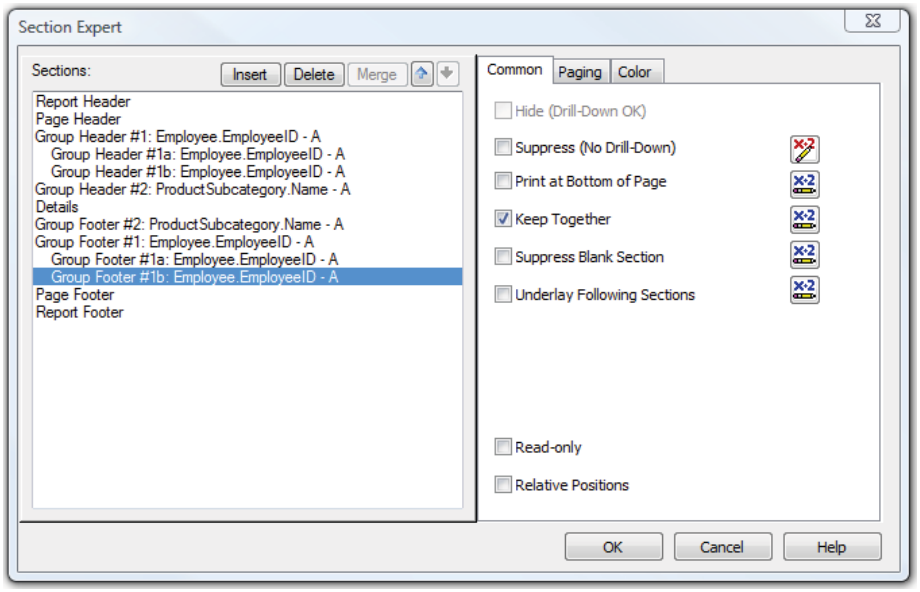
1. Return to the **Design** view. It is easier to add and format objects from Design view.

2. Insert an additional group footer section for **Group Footer 1** and add a text object to it with the sales person's first name embedded The text object should look similar to the following:

Congratulations {FirstName}, you sold more than $50 million dollars!

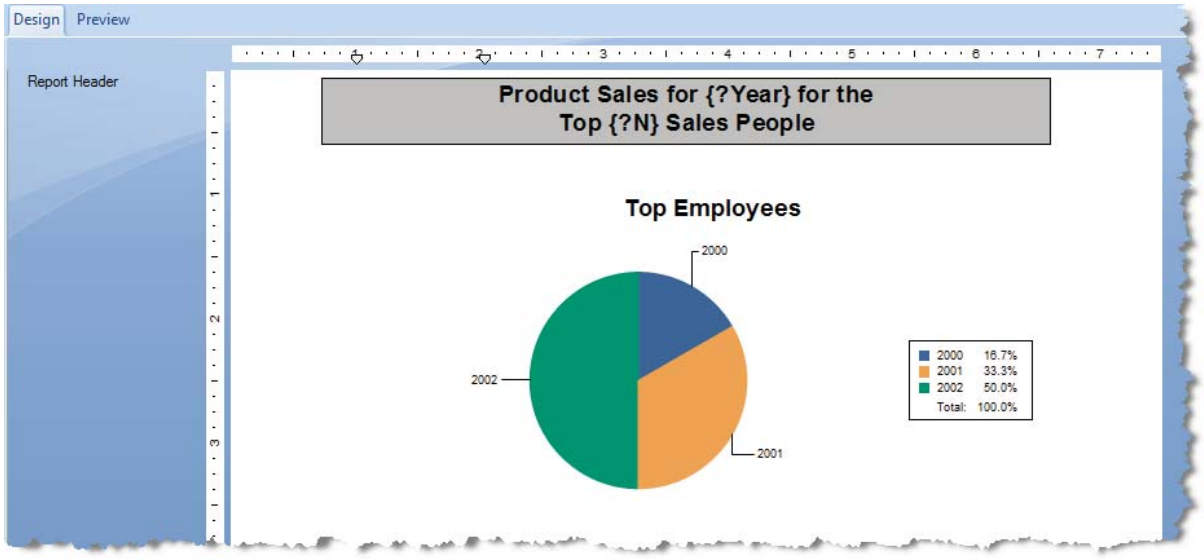
3. If you preview the report now you see that this text object displays for every sales person regardless of what they have sold. You now need to conditionally suppress the section. You need the section to suppress if sum of sales by Employee ID is less than $50,000,000. In the **Section Expert** add the following formula to the conditional box next to the *Suppress* option for **Group Footer #1b**:

   Sum({SalesOrderHeader.SubTotal},{Employee.EmployeeID}) < 50000000



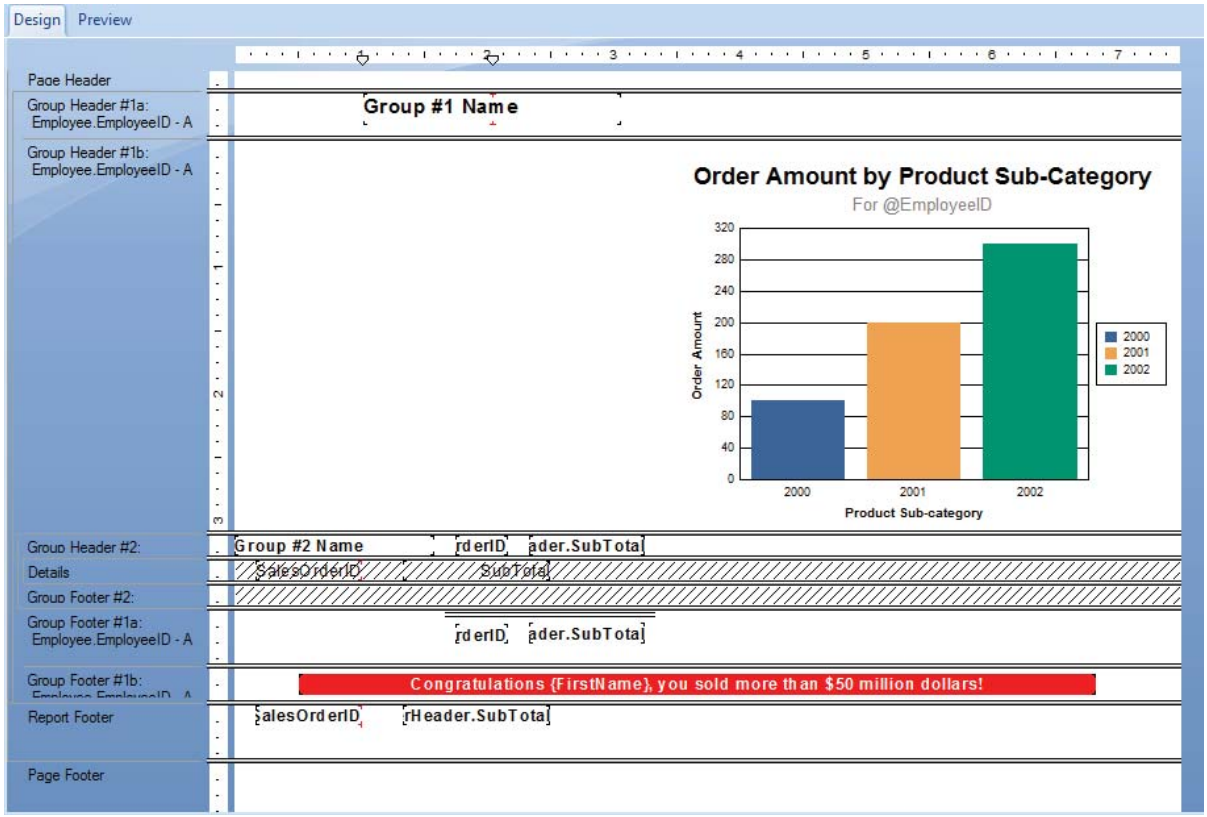
4. **Save** your report and run it with different values for N and the year. Make sure it meets the business specifications.

The Design view of your report should look as follows:



This is the Design View of the Report Header – note the chart is a place holder and does not show the data.

The Design view for the rest of the report should look as follows:



©2008 Vision Harvest, Inc.

NOTES

©2008 Vision Harvest, Inc.

Lesson 3
Using Advanced Formula Features

Lesson Objectives

After completing this lesson, you will be able to:

- **Use the Select Case Statement**
  Learn when and how to use the Select statement
- **Use the For Loop and While Loop constructs**
  Learn when and how to use the For and While loop syntax
- **Understand the Join and Split functions**
  Use the JOIN function to create a string variable from an array of string values. Use the SPLIT to convert a single string into an array object
- **Learn to create your own Array Object**
  Building arrays are a great solution for many difficult type of calculations

Alternatives to the IF THEN ELSE Control Structure

Select Case Statement

The Select expression (sometimes known as a Case expression) is similar to the If…then…else expression. However, if you have several multiple conditions it can be simpler to understand and easier to write a Select expression. Suppose you have a conditional formula that contains several possibilities. A multiple nested if…then…else expression can be very complex and difficult to understand. A select expression, on the other hand, can simplify matters considerably.

A select expression looks like this:

```
Select x
  Case ConditionA : CommandA
  Case ConditionB : CommandB
  Case ConditionC : CommandC
  Default : Command D
```

After the select keyword, you enter the expression you want to evaluate. This is called the select condition. For example, if you were creating regions of the world from countries, you would enter the {Customer.Country} field.

After each case keyword, you add the condition to evaluate. For example let's say we want to modify country so that USA and Canada are grouped as North America and other countries are left without further grouping. The complete Case statement would look as follows:

```
Select {Customer.Country}
Case "USA" : "North America"
Case "Canada" : "North America"
Default: {Customer.Country}
```

After the default keyword, you tell Crystal what to do as the default. It is used only if the select condition does not match any of the specified cases.

TIP: You can add multiple specific conditions by separating them with a comma. You can add comparative conditions by using the keyword "is", for example, is < 1000, is <> "New York", etc.

To use the case expression, you are going to modify the sample **Sales Orders Analysis** report. This report currently shows quarter 1, 2004 sales order information rolled-up as a summary style report. The bottom part of the report displays an order matrix with calculations, which have yet to be created.

The report's order matrix uses the Percentage Of summary function to determine the sales order point for the 40%, 60%, 80% and 100% in terms of orders placed and the individual amounts. With the current data set, this means:

- 40% of orders placed were for $69.99 or less.
- 60% of orders placed were for $617.47
- 80% 0f orders placed were for $2,339.97
- The top sales order amount was for $129,896.10

Using these numbers we can calculate the total revenue and number of orders within each percentage range of 0-40%, 41-60%, 61-80% and 801% or higher.

Before building these calculations we want to create a formula which "marks" each sales order with the name of the basis level it belongs to.

❖ Exercise 3.0 – Use a Select Expression to mark Sales Order Basis Levels

1. Open the **Sales Order Analysis.rpt** report from the **CR2008 D3\L03 - Using Advanced Formula Features** directory

2. Save the report and rename it to **Sales Order Analysis Final.rpt**. Preview the report and take a moment to familiarize yourself with the report layout.

3. Now you are ready to create the case statement to evaluate the sales revenue and count for each level basis. Return to **Design** view, open the **Field Explorer**, and start a new formula named **Order Level**

4. In the **Operators** list on the right, expand the **Control Structures** operator category, then double click the **select x case a: y default : z** operator

5. Using the Select Expression syntax, create the following formula:

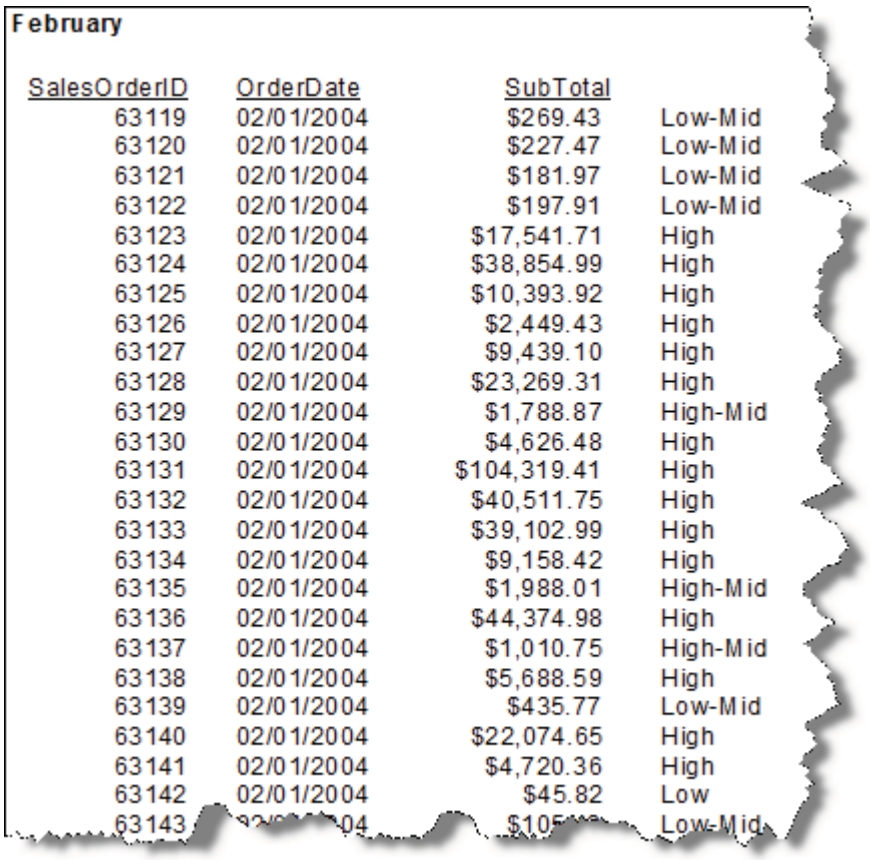
```
Select {SalesOrderHeader.SubTotal}
        Case Is <= PthPercentile (40, {SalesOrderHeader.SubTotal}) : "Low"
        Case Is <= PthPercentile (60, {SalesOrderHeader.SubTotal}) : "Low-Mid"
        Case Is <= PthPercentile (80, {SalesOrderHeader.SubTotal}) : "High-Mid"
        Default : "High"
```

This formula uses the "Is" operator to do the <= comparison. Remember a once a Case comparison returns a true (much like and If-Then_Else) the reaminding Case comparisons are ignored.

6. Check the formula for errors, then save and close it

7. Insert the formula into the **Details** section just to the right of the of the **SalesOrderHeader.SubTotal** field. Delete the field header in the **Page Header** section

8. Save (**Sales Order Analysis Final**) and preview the report. Double-Click the **February** sales total to drill-down to the sales order record level to confirm the **Order Level** formula is performing properly. The drill-down report should look like the following illustration:



February

SalesOrderID	OrderDate	SubTotal	
63119	02/01/2004	$269.43	Low-Mid
63120	02/01/2004	$227.47	Low-Mid
63121	02/01/2004	$181.97	Low-Mid
63122	02/01/2004	$197.91	Low-Mid
63123	02/01/2004	$17,541.71	High
63124	02/01/2004	$38,854.99	High
63125	02/01/2004	$10,393.92	High
63126	02/01/2004	$2,449.43	High
63127	02/01/2004	$9,439.10	High
63128	02/01/2004	$23,269.31	High
63129	02/01/2004	$1,788.87	High-Mid
63130	02/01/2004	$4,626.48	High
63131	02/01/2004	$104,319.41	High
63132	02/01/2004	$40,511.75	High
63133	02/01/2004	$39,102.99	High
63134	02/01/2004	$9,158.42	High
63135	02/01/2004	$1,988.01	High-Mid
63136	02/01/2004	$44,374.98	High
63137	02/01/2004	$1,010.75	High-Mid
63138	02/01/2004	$5,688.59	High
63139	02/01/2004	$435.77	Low-Mid
63140	02/01/2004	$22,074.65	High
63141	02/01/2004	$4,720.36	High
63142	02/01/2004	$45.82	Low
63143	[illegible]	[illegible]	Low-Mid

The next formula we need to create will do the calculations for the Order Matrix. Specifically, we need to total the sales order revenue and count for each basis level. Multiple approaches could be used to accomplish this task. We will use a combination of variables and the Select expression for our approach. Using variables allows us to perform the revenue and order count calculations within one formula.

❖ Exercise 3.1 - Use the Case Expression with Variables

1. Return to the **Design** view and create a new formula called **OrderCalc**

2. Enter eight variable declarations using the following code:

```
CurrencyVar LowRev;
CurrencyVar LowMidRev;
CurrencyVar HighMidRev;
CurrencyVar HighRev;
NumberVar LowCnt;
NumberVar LowMidCnt;
NumberVar HighMidCnt;
NumberVar HighCnt;
```

We have four basis levels to collect information for. Also, we need to calculate for revenue and we need count the number of orders for each basis level. The above variables will be used later in other formulas for displaying their appropriate value.

3. Move your cursor two lines below the last line in the formula for visual effect. You may need to use the **Enter** key to create the new lines. Enter the following code:

```
Select {SalesOrderHeader.SubTotal}
        Case Is <= PthPercentile (40, {SalesOrderHeader.SubTotal}) :
                (LowRev := LowRev + {SalesOrderHeader.SubTotal};
                LowCnt := LowCnt + 1)
```

Notice for the first Case comparison we test for any orders in the 40th percentile and set the appropriate variable values. We are able to perform multiple independent commands using parentheses.

4. Next, add the following lines to complete the formula:

```
Case Is <= PthPercentile (60, {SalesOrderHeader.SubTotal}) :
        (LowMidRev := LowMidRev + {SalesOrderHeader.SubTotal};
        LowMidCnt := LowMidCnt + 1)
Case Is <= PthPercentile (80, {SalesOrderHeader.SubTotal}) :
        (HighMidRev := HighMidRev + {SalesOrderHeader.SubTotal};
        HighMidCnt := HighMidCnt + 1)
Default : (HighRev := Highrev + {SalesOrderHeader.SubTotal};
          HighCnt := HighCnt + 1)
```

If the previous Case comparisons returns false then that means the order fits into the High basis level and we use the Default statement to set the values with a need for testing.

5. Check the formula for errors. Save, but don't close it. The complete formula should match the following:

```
CurrencyVar LowRev;
CurrencyVar LowMidRev;
CurrencyVar HighMidRev;
CurrencyVar HighRev;
NumberVar LowCnt;
NumberVar LowMidCnt;
NumberVar HighMidCnt;
NumberVar HighCnt;

Select {SalesOrderHeader.SubTotal}
        Case Is <= PthPercentile (40, {SalesOrderHeader.SubTotal}) :
                (LowRev := LowRev + {SalesOrderHeader.SubTotal};
                LowCnt := LowCnt + 1)
        Case Is <= PthPercentile (60, {SalesOrderHeader.SubTotal}) :
                (LowMidRev := LowMidRev + {SalesOrderHeader.SubTotal};
                LowMidCnt := LowMidCnt + 1)
        Case Is <= PthPercentile (80, {SalesOrderHeader.SubTotal}) :
                (HighMidRev := HighMidRev + {SalesOrderHeader.SubTotal};
                HighMidCnt := HighMidCnt + 1)
        Default : (HighRev := Highrev + {SalesOrderHeader.SubTotal};
                  HighCnt := HighCnt + 1)
```

Since this formula actually displays eight different values, it does not serve any display purpose for us. We need to create individual formulas for displaying the variables. We could create eight individual variables to display the values or we could do something slightly different.

6. Create a new formula called **DispRev**. From the **Formula Workshop** enter the following code:

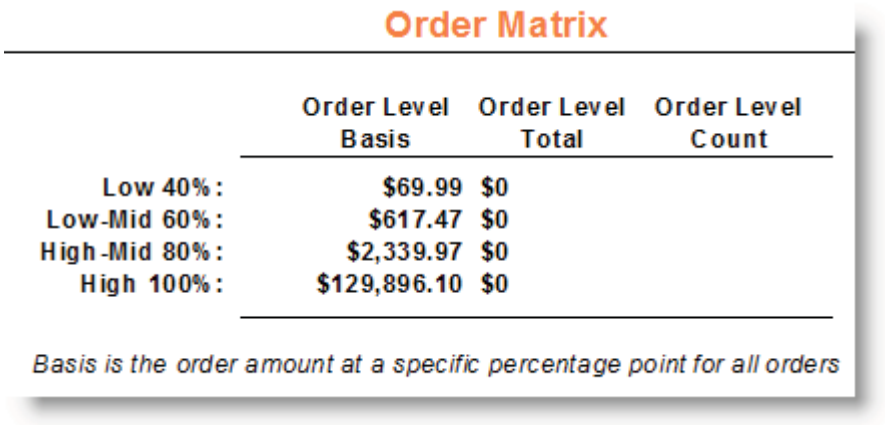
```
WhilePrintingRecords;
CurrencyVar LowRev;
CurrencyVar LowMidRev;
CurrencyVar HighMidRev;
CurrencyVar HighRev;

ToText(LowRev,0,"") + ChrW(13) +
ToText(LowMidRev,0,"") + ChrW(13) +
ToText(HighMidRev,0,"") + ChrW(13) +
ToText(HighRev,0,"")
```

We are taking the revenue variables (always must re-declare in a new formula) and converting them into a multi-line string value. We are also rounding to the nearest whole number and adding thousands separators using the ToText(x,y,z) function arguments. Notice we used the WhilePrintingRecords function in this formula, but not in the previous formula. Why? Because the previous formula used summary subtotals which forces the formula to the "print time" pass. We had to use the WhilePrintingRecords function in this formula to ensure it processed in the same pass to get proper values.

7. Check the formula for errors, then save and close it

8. Place the **DispRev** formula into the **Order Matrix** under the **Order Level Total** column heading. Re-size it to match the text box under the **Order Level Basis** column heading, make the font bold and right justified. Adjust as necessary and then preview the report. Your **Order Matrix** should look similar to the following:



Order Matrix

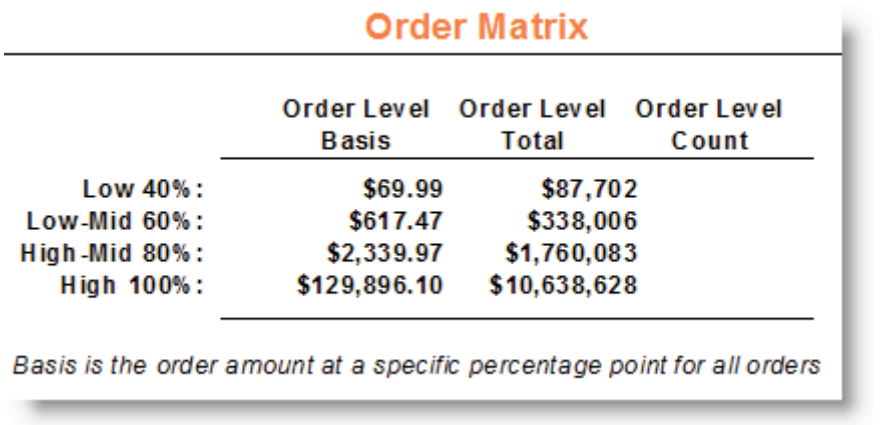
	Order Level Basis	Order Level Total	Order Level Count
Low 40%:	$69.99	$0	
Low-Mid 60%:	$617.47	$0	
High-Mid 80%:	$2,339.97	$0	
High 100%:	$129,896.10	$0	

Basis is the order amount at a specific percentage point for all orders

Notice all the values are zero. This is because Print Time formulas must be physically on the report or they will not calculate. The OrderCalc formula is doing all of the calculations, but has not been placed on the report. Since the calculations are for each record, the formula will need to be placed in the Details section.

9. Return to **Design** view and place the **OrderCalc** formula in the **Details** section all the way to the very right of the section, away from other **Details** objects. Delete the **Field Header** in the **Page Header** section for this formula

10. Before previewing the report, suppress the **OrderCalc** formula in the **Details** section. Preview the report and now you should see proper numbers displayed in the **DispRev** formula to match the following example:



Order Matrix

	Order Level Basis	Order Level Total	Order Level Count
Low 40%:	$69.99	$87,702	
Low-Mid 60%:	$617.47	$338,006	
High-Mid 80%:	$2,339.97	$1,760,083	
High 100%:	$129,896.10	$10,638,628	

Basis is the order amount at a specific percentage point for all orders

11. Create the **DispCnt** formula with the following syntax below:

```
WhilePrintingRecords;
NumberVar LowCnt;
NumberVar LowMidCnt;
NumberVar HighMidCnt;
NumberVar HighCnt;

ToText(LowCnt,0,"") + ChrW(13) +
ToText(LowMidCnt,0,"") + ChrW(13) +
ToText(HighMidCnt,0,"") + ChrW(13) +
ToText(HighCnt,0,"")
```

12. Check the formula for errors, then save and close it.

13. Place the **DispCnt** formula into the **Order Matrix** under the **Order Level Count** column heading. Re-size it to match the text box under the **Order Level Basis** column heading, make the font bold and right justified. Adjust as necessary and then preview the report.

14. Save (**Sales Order Analysis Final**). Your **Order Matrix** should look similar to the following:

Order Matrix

	Order Level Basis	Order Level Total	Order Level Count
Low 40% :	$69.99	$87,702	2,492
Low-Mid 60% :	$617.47	$338,006	1,161
High-Mid 80% :	$2,339.97	$1,760,083	1,219
High 100% :	$129,896.10	$10,638,628	1,215

Basis is the order amount at a specific percentage point for all orders

Using the For Loop Control Structure

With the If and Select Case expressions, Crystal passes through each expression at the most once (and sometimes not at all if the condition is not met) during the formula's evaluation. For loops enable you to evaluate a sequence of expressions multiple numbers of times. For Loops are very good for cycling through arrays and picking out values. For example, a multi value parameter field places values in an array and a For Loop would be useful for displaying these values. The table below identifies and explains the parts of of a For loop.

For Loop Construct	
Component	**Description**
Counter Variable	The counter variable is declared before beginning the For loop. It is used to set the starting point in the array object to begin the looping process. It also is used to identify the array index position to use for each loop through the array. This is why a fixed number does not work since the counter variable needs to increment by 1 or more for each loop through the array
For	This is the operator command which tells Crystal a For loop is to be performed
Step	This operator command is used to determine how to increment (how much) the counter variable after each loop through the array. This is an optional operator and the value of 1 is used to increment the counter variable if not used
Do	This operator command signals to start the loop and is used to define the type of loop - discussed later
Statement	The statement is the action to be performed during each loop. The result can be any data type required
Exit For	This operator provides a mechanism for exiting out of a loop when a certain condition exists. It is common to use when building string variables where the size of the variable may exceed 65535 characters, which is the max limit: **If length(DisplayEmployees) > 64000 then** **(DisplayEmployees := "variable length exceeded "; Exit For));**

Let's consider that we have an array of employee names in a parameter array with the values of "Mark, Spencer, Deb, Karen" called {?Names}. We would like to display these in a report as follows "Employees Selected Are Mark, Spencer, Deb, Karen" Using the For Loop, our formula would look as follows:

//First declare the variables needed:
NumberVar Counter;
StringVar DisplayEmployees : = "Employees Selected Are ";

//Now set up the For Loop
For Counter : = 1 to Ubound (?Name) Step 1 Do

//Now we build our display variable
(DisplayEmployees := DisplayEmployees + {?Name}[Counter] + ",")

//The last line is used to force the value returned by the formula
DisplayEmployees

We have a report called Customer Segmentation which shows the number of customers within each state in the US. The report has a parameter that allows the user to select specific states or all states. We need to edit the report to show if the user selected all states or list which states they did select.

❖ Exercise 3.3 – Using the For Loop

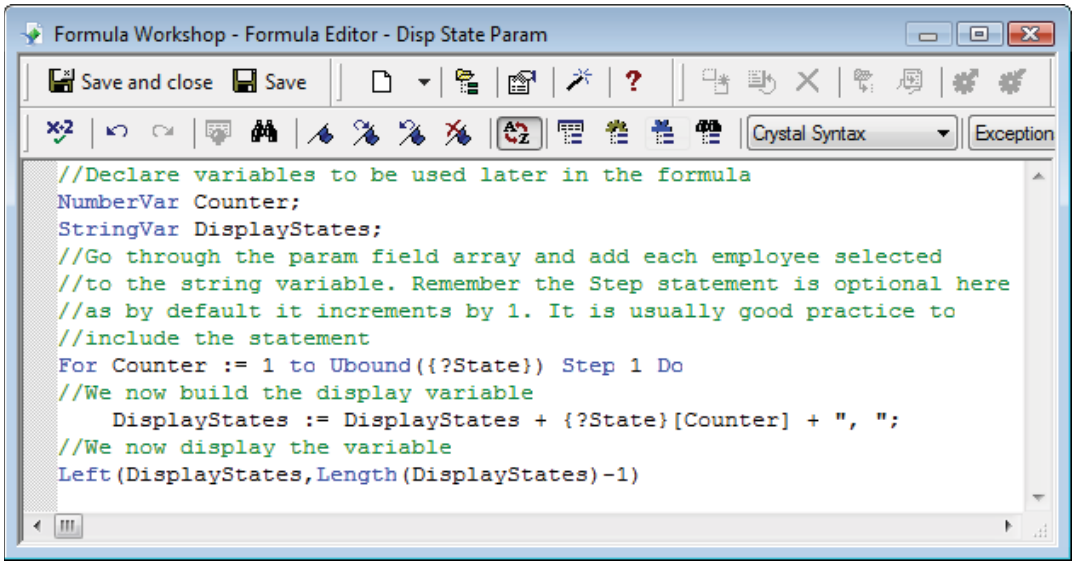
1. Open the **Customer Segmentation** report.

2. Next to our title box we would like to see a list of states selected, so let's build a formula using the For Loop to do just that. Create a formula named **Disp State Param** and place it in the Report Header. Refer to the example in the lesson if you need help or use the formula below:

```
NumberVar Counter;
StringVar DisplayStates;

For Counter := 1 to Ubound({?State}) Step 1 Do
        DisplayStates := DisplayStates + {?State}[Counter] + ", ";
DisplayStates
```

It's a good idea to make comments about the formula. It'll made editing and reviewing the formulas weeks or months later much easier. Refer to the example below:



3. You may notice that there is an extra comma space at the end of your formula. To prevent this from displaying we can display the string with the last two characters stripped off. You can do this using the Left function which displays the left characters of a string up to a certain point which in this case would be length of the string minus 1 (the training space is ignored). Our last line of the formula would read:

Left(DisplayStates,Length(DisplayStates)-1)

The different Types of While Loops

Like the For Loop the While Loop is a looping construct that executes a loop but in this case the loop is not executed a certain number of times but is executed while a condition is true and stops as soon as the condition is not true. There are two types of While Do loops.

While ... Do

There is the While ... Do which evaluates the condition, and if the condition is true, then it evaluates the expression following the Do. When it has finished doing this, it evaluates the condition again and if the condition is true, it evaluates the expression following the Do again. It continues repeating this process until the condition is false. **If the condition is never true, the While ... Do loop will never run.**

Example: We have an array which captures several pieces of information about a product. Only one of those items is numeric and it is the price of the product. We need to find what that price is. The way the information is collected initially can change when the report is run means we don't know where in the array the price is located. We will use the While ... Do loop to determine the price location.

```
StringVar Array ProductInfo := Redim ProductInfo[4];
ProductInfo := ["Bicycle","2530.50","Mozzie","Blue"];
NumberVar k := 1;
NumberVar IndexPos := -1;

While k <= Ubound(ProductInfo) and IndexPos = -1 Do
        (If NumericText (ProductInfo[k]) then IndexPos := k;
         K := K + 1);

ToNumber(ProductInfo[IndexPos])
```

The While ... Do in the formula stops looping after the second index item and the last line returns the value 2530.50.

Do ... While

The Do ... While loop will always evaluate the expression at least once. It then evaluates the condition, and if the condition is true, evaluates the expression again. This process continues until the condition is false. **Even if the condition is never true, the Do ... While will run at least once.**

Example: The same situation as in the While ... Do loop example, but we will construct with the Do ... While construct.

```
StringVar Array ProductInfo := Redim ProductInfo[4];
ProductInfo := ["Bicycle","2530.50","Mozzie","Blue"];
NumberVar k := 1;
NumberVar IndexPos := -1;

Do (If NumericText (ProductInfo[k]) then IndexPos := k; K := K + 1;)
While k <= Ubound(ProductInfo) and IndexPos = -1;

If NumericText(ProductInfo[IndexPos]) then ToNumber(ProductInfo[IndexPos])
Else 0
```

Note: The While loops use an Exit While statement similar to the use of Exit For in For loops.

If we were to look closely at the number of times the formula loops through the array by outputting the value of the variable k, we would see both formulas loop the same number of times. The difference is the Do ... While loop guarantees the array is looped at least once. The While ... Do does not and unless the initial condition is true, will never loop.

Building Arrays

As has been pointed out, an array is a single object which holds multiple values. For example, a string array (Names) could hold the values "Mark", "Spencer", "Deb", "Karen". This means the array holds four values and "Mark" is index 1 and "Karen" is index 4. We can add specific items to a specific index. Suppose we wanted to replace "Spencer" with "Zeke". The formula syntax would be:

```
Names [2] := "Zeke"
```

The Names array would now have the values, "Mark", "Spencer", "Deb", "Karen". Before we can add a value to an index position, the position must already exist. This means we need to re-dimension (define number of indexes) the array. There are two ways of doing this:

- **Redim** - This operator defines the number of indexes in the array. It also will delete any existing values. For example:

  ```
  Redim Names [5}
  ```
  The Names array now has five indexes instead of four and all of the values are blank.

- **Redim Preserve** - This operator performs the same action as Redim, but preserves the current index values. For example:

  ```
  Redim Preserve [5]
  ```
  The Names array now has five indexes and the values are:
  "Mark","Spencer","Deb","Karen,"" "

Let's assume we would like a formula that gives us a list of unique area codes held in an array for future use

❖ Exercise 3.4 - Manually Building an Array

1. Return to the **Customer Segmentation** report, **Design** view

Create a new formula called **AreaCode**. This formula will build an array containing all unique area codes. The formula syntax should look like the following:

```
//Make sure this formula is performed in the second pass
//Must be physically located in the Group Header #1
WhilePrintingRecords;
//Need various variables for use later in the formula
NumberVar k;
StringVar AreaCode := {vIndividualCustomer.Phone}[1 to 3];
StringVar array AreaCodeList;
//We want to an add area codes only
//if it doesn't already exist in the array
If Not (AreaCode in AreaCodeList) then
        (k := k + 1;
        Redim Preserve AreaCodeList[k];
        AreaCodeList[k] := AreaCode);
//Will output all of the array indexes in one single string
Join(AreaCodeList,", ")
```

2. Check the formula for errors, then save and close it. Place the formula in the **Details** section and again in the **Group Footer #1**
We need the formula on the Details section to process for every record and in the Group Footer to display on the Preview page.

3. Save (**Sales Order Analysis Final**) and refresh the report. Select only **Arizona**, **Florida**, **Georgia** and **Illinois** for the State parameter

Notice how the the AreaCode parameter displays a running list of area codes. We would prefer to display only the area codes for the current state.

4. Create a new formula called **Reset AreaCodeList Array** and add the following formula:

```
//Make sure this formula is performed in the second pass
//Must be physically located in the Group Header #1
WhilePrintingRecords;
//Re-declare string array and number counter
StringVar array AreaCodeList;
NumberVar k;
//Reset the variables
Redim AreaCodeList[1];
k := 0;
```

5. Preview the report and notice the area codes display only for the specific state

One issue we haven't addressed with the Customer Segmentation Final report is how do we handle a situation where a user selects "*ALL* from the State parameter as well as selecting a number of states. This means the parameter will contain multiple values with "*ALL*" being one of them. There are a number of ways to deal with this issue and much depends on the business rules for how to handle the situation. For our needs, the report filter will return all states if the value "*ALL*" is selected. So, we need to have a formula which will properly display the words "All States Selected" if "*ALL*" is in the parameter list or list the states selected if it is not.

We also would like to see the states listed in a vertical format.

❖ Exercise 3.5 - Using the While ... Do loop

1. Return to **Design** view and delete the **Disp State Param** formula from the **Report Header** section

2. Create a new formula called Disp State Param Alt and add the following syntax:

```
//We will use several variables throughout the rest of the formula
NumberVar Counter := 1;
//This variable is used to determine if "*ALL*" is in the param list
BooleanVar AllStates := If {?State} = "*All*" then True Else False;
//This is the final output var. It is based on the value "*ALL*" being in
//the param list or not and uses the boolean AllStates var
StringVar DisplayStates := If AllStates = True Then "All States Selected" Else "";
//Using the While ... Do loop means it won't run at all if the variable
//AllStates is set to True
While Counter <= Ubound({?State}) and AllStates = False Do
        (DisplayStates := DisplayStates + {?State}[Counter] + ChrW(13);
        Counter := Counter + 1);
//Need this next line to force the final output
DisplayStates
```

3. Place the parameter in the **Report Header** section and re-size to several lines tall

4. Refresh the report using different parameter combinations of the "***ALL***" selection to see if the formula performs correctly

5. Save the report as **Customer Segmentation.rpt**

The Split Function

The Split Function takes a string variable with elements separated by a delimiter and puts them in an array. So if you had a string containing the value "NY/NJ/MA" you can split the variable so that the elements separated by the / are placed into an array. In this case the array would have 3 elements (NY,NJ,MA). As with JOIN the SPLIT function only works with strings.

This function is particularly useful when dealing with database string fields containing multiple values as one string value. This happens sometimes when legacy data (mainframe) is ported to a new database (SQL) and the database developers decided to combine many detail items into one string value to simplify the process.

NOTE: If you do not specify a delimiter, you cannot specify count nor compare.

Examples:

The following examples demonstrate the various SPLIT function configurations and the results that are returned when these setups are utilized. These examples are applicable to both Basic and Crystal syntax:

Split ("NY NJ MA")
- Returns an array that contains 3 elements, (NY, NJ, MA)

Split ("NY//NJ//MA", "//")
- Returns an array that contains 3 elements, (NY, NJ, MA)

Split ("NY//NJ//MA", "//", 2)
- Returns an array that contains two elements, (NY, NJ//MA) The last element in the array is a concatenation of the 2nd substring and the remaining substring

Split ("NY and NJ and MA", " And ", -1, 0)
- Returns an array that contains 1 element, (NY and NJ and MA). The final argument of 0 indicates that the delimiter is case sensitive and in this case does not find the delimiter And but only and since it is case sensitive these do not match. Note: The '–1' as the second to last argument means return all substrings

Split ("NY and NJ and MA", " And ", -1, 1)
- Returns an array that contains 3 elements, (NY, NJ, MA) as the final argument of 1 means match the delimiter regardless of case.

❖ Exercise 3.6 - Using the SPLIT function

Our Customer Segmentation Final report contains an unused string formula called sports. It list several sports with each sport being separated by "++". Using the Split function, convert this formula and create an array of values. Print out the first value as the result of the formula since the value cannot be an array.

1. Return to the **Customer Segmentation Final** report go to **Design** view

2. Create a formula named Sport Array and add the following syntax:

   Split ({@Sports},"++")[1]

   This will return the first index in the array, "Baseball"

3. Place the **Sports** formula into the **Report Footer**, re-size it display the complete string. Place the **Sports Array** formula under the **Sports** formula in the **Report Footer** and then preview. . Notice the result of the formula. You should see the 1st sport in the array

Edit the previous formula to create an array of the second item. Your formula should look like this:

Split ({@Sports},"++")[2]

This will return the first index in the array, "Baseball"

Crystal Formula Sizing Limitations

It is always good to know the limitations of a language so here are the sizing limitations of the Crystal Formula Language:

- The maximum length of a String constant, a String value held by a String variable, a String value returned by a function or a String element of a String array is 65,534 characters
- The maximum size of an array is 1000 elements (indexes)
- The maximum number of arguments to a function is 1000. (This applies to functions that can have an indefinite number of arguments such as Choose)
- The maximum number of loop condition evaluations per evaluation of a formula is 100,000. Note that you can use the Option Loop statement to change this limit
- Date-time functions modeled on Visual Basic accept dates from year 100 to year 9999. Traditional Crystal Reports functions accept dates from year 1 to year 9999
- There is no limit on the size of a function.

©2008 Vision Harvest, Inc.

NOTES

©2008 Vision Harvest, Inc.

NOTES

©2008 Vision Harvest, Inc.

Lesson 7
Tips and Tricks

Lesson Objectives

This lesson is slightly different from other lessons as we are going to begin with a report and add onto it learning some tips and tricks along the way. The tips and tricks you will learn are as follows:

- **Create the Manual Cross-Tabs**
  Create a cross tab without the expert

- **Format a subreport to show a message if no records**
  When a subreport is blank show a message instead of a blank subreport

- **Allow an ALL selection on a multi select dynamic parameter field**
  Dynamic parameters do not allow for "wild-card" selections unless, of course, you know how to get around this limitation

- **Display a range of values from a parameter field**
  Know how to effectively show the values in a dat ranged parameter

- **Use Conditional Formatting to alternate background color**
  On reports that have many rows and columns of data, often it is easier to read if every other (or every third or fourth) row has the background shade

- **Use additional sections and conditional formatting to provide a different page header for the last page**
  You can display a different page header for the last page containing Grand Totals or other info

- **Use Character mapping to provide flair for your reports**
  By using different fonts, you can display check marks to select record

©2008 Vision Harvest, Inc.

Creating a Cross-Tab without the Expert

The number of reasons for building manual Cross-Tabs have dropped dramatically with the addition of Embedded Summaries, Calculated Members and Grid Value function. However there will still be opportunities to build this type of report and probably for many versions to come. Manual Cross-Tabs offer the ability to get exactly what you want in report design with little or no restrictions. The reason for this flexibility is because you are doing all the work in terms of design and formula calculations.

The first example of a manual Cross-Tab uses the Manual Cross-Tab 1.rpt file found in the Tips and Tricks directory of your class resource files. This report is pre-formatted to look like a regular Cross-Tab, but really is a standard grouped report. The formatting has been set to read only and all objects are locked into place. You can only affect formatting and positioning with new objects (formulas) placed on the report. It is possible to override these settings, but not advisable. The initial view of the report looks similar to the following:

Territory Sales
Last Three Years

				Territory	
Territory	2002	2003	2004	Total	Average
Australia			$1,758,386	$1,758,386	$1,758,386
Canada	$3,668,140	$2,968,929	$4,954,295	$11,591,365	$3,863,788
France		$1,677,652	$3,827,950	$5,505,603	$2,752,801
Germany			$2,241,204	$2,241,204	$2,241,204
United Kingdom		$5,287,044	$5,015,682	$10,302,727	$5,151,363
United States	$16,059,485	$23,075,616	$24,963,421	$64,098,522	$21,366,174
	$19,727,625	$33,009,242	$42,760,939	$95,497,806	

Our task is to create formulas to calculate three years of sales for each territory, a three year grand total and a territory yearly average. The territory yearly average will need to reflect averages based on the number of years in operation and not necessarily for the past three years. The Cross-Tab Expert could possibly be used to generate the required information, but would require more effort in terms of formulaic expressions. When complete the report will look similar to the following:

Territory Sales
Last Three Years

				Territory	
Territory	2002	2003	2004	Total	Average
Australia			$1,758,386		
Canada	$3,668,140	$2,968,929	$4,954,295		
France		$1,677,652	$3,827,950		
Germany			$2,241,204		
United Kingdom		$5,287,044	$5,015,682		
United States	$16,059,485	$23,075,616	$24,963,421		
	$19,727,625	$33,009,242	$42,760,939		

❖ Exercise 7.0 - Create Manual Cross-Tab Calculation Formulas

1. Open the **Manual Cross-Tab 1.rpt** file found in the **CR2008 D3 Class Resources/ L07 - Tips and Tricks** directory. Return to **Design** view if necessary

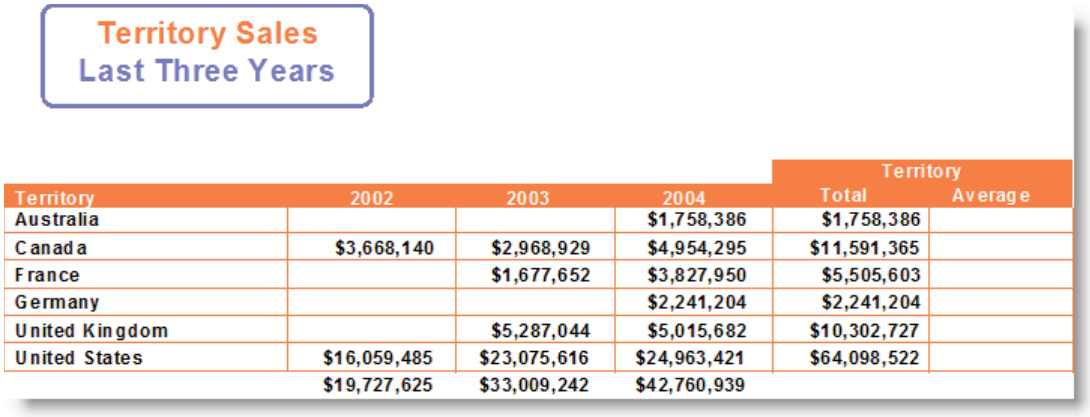
2. Create a new formula called **Territory Total (3yrs)** and add the following formula. You may exclude comments if desired:

```
//This formula automatically prints in the second pass
//because of the subtotals used later. No need to include
//the WhilePrintingRecords function.
//We need these intial variables to determine if yearly data
//exists, and if so, how much is it. We set everything to zero
//and only change later if data does exist
CurrencyVar Yr2002 := 0;
NumberVar Yr2002Exist := 0;
CurrencyVar Yr2003 := 0;
NumberVar Yr2003Exist := 0;
CurrencyVar Yr2004 := 0;
NumberVar Yr2004Exist := 0;
//The series of If-Then_else statements determine if data exists
//for the year and only changes the variable values if it does
If Not(IsNull(Sum ({vSalesPersonSalesByFiscalYears.2002}, {@Territory}))) then
        (Yr2002 := Sum ({vSalesPersonSalesByFiscalYears.2002}, {@Territory});
        Yr2002Exist := 1);

If Not(IsNull(Sum ({vSalesPersonSalesByFiscalYears.2003}, {@Territory}))) then
        (Yr2003 := Sum ({vSalesPersonSalesByFiscalYears.2003}, {@Territory});
        Yr2003Exist := 1);

If Not(IsNull(Sum ({vSalesPersonSalesByFiscalYears.2004}, {@Territory}))) then
        (Yr2004 := Sum ({vSalesPersonSalesByFiscalYears.2004}, {@Territory});
        Yr2004Exist := 1);
//This formula will process once for each territory. We need to know
//how many years of data that territory has. We will use this variable value
//in the {@Territory Total(3yrs) Average} formula
NumberVar YrCount := (Yr2002Exist + Yr2003Exist + Yr2004Exist);
//RowTotal adds all of the years together and will be used as the final output
//We set it to zero to clear out the value from the previous Territory grouping
CurrencyVar RowTotal := 0;
//The ColumnTotal is a manual running total and is not cleared out from one
//group to the next. We will use it in the
//{Territory Total (3yrs) ColTot} formula
CurrencyVar ColumnTotal;
//Now doing all of the final calculations
RowTotal := Yr2002 + Yr2003 + Yr2004;
ColumnTotal := ColumnTotal + RowTotal;
//Forcing which variable is output for this formula's display
RowTotal
```

3. Check the formula for errors, save and close it. Place the formula onto the **Group Footer #1** in the column slot for **Territory Total**. Size it to match the **Total** label width, make it bold and decrease the decimals to none. Preview the report and it should look similar to the following:



Territory	2002	2003	2004	Territory Total	Territory Average
Australia			$1,758,386	$1,758,386	
Canada	$3,668,140	$2,968,929	$4,954,295	$11,591,365	
France		$1,677,652	$3,827,950	$5,505,603	
Germany			$2,241,204	$2,241,204	
United Kingdom		$5,287,044	$5,015,682	$10,302,727	
United States	$16,059,485	$23,075,616	$24,963,421	$64,098,522	
	$19,727,625	$33,009,242	$42,760,939		

4. Return to **Design** view and create a new formula called **Territory Total (3yrs) Average**. This formula will take variable calculated in the previous formula and calculate the yearly average. The formula syntax is:

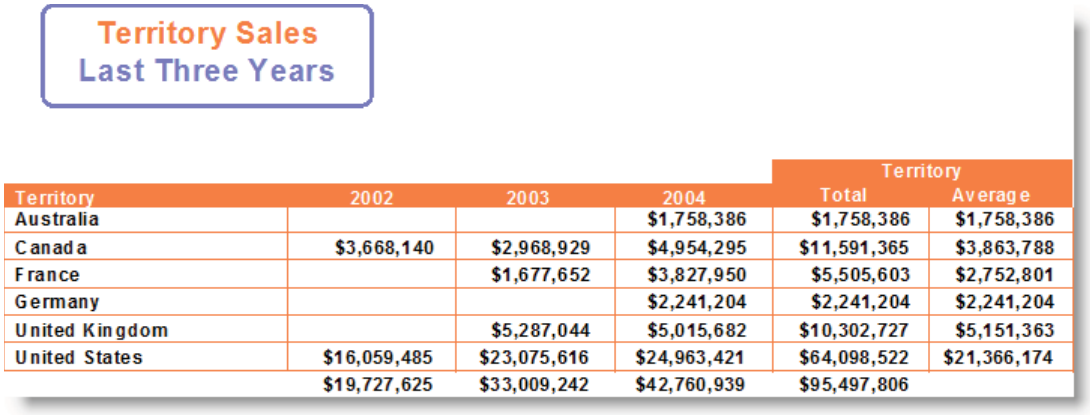
```
//We must force the formula into the second pass
WhilePrintingRecords;
//Re-declare variables from the {@Territory Total(3yrs)} formula
CurrencyVar RowTotal;
NumberVar YrCount;
//Do the average calculation
RowTotal / YrCount
```

5. Check the formula for errors, save and close it. Place the formula onto the **Group Footer #1** in the column slot for **Territory Average**. Size it to match the **Average** label width, make it bold and decrease the decimals to none. Preview the report

6. Return to **Design** view and create a new formula called **Territory Total (3yrs) ColTot**. This formula simply displays the running total variable called **ColumnTotal** we created in the initial formula. The syntax is:

```
WhilePrintingRecords;
CurrencyVar ColumnTotal;
```

7. Check the formula for errors, save and close it. Place the formula onto the **Report Footer** in the column slot for **Territory Total**. Size it to match the **Total** label width, make it bold and decrease the decimals to none. Preview the report and it should resemble the following illustration:



Territory	2002	2003	2004	Territory Total	Territory Average
Australia			$1,758,386	$1,758,386	$1,758,386
Canada	$3,668,140	$2,968,929	$4,954,295	$11,591,365	$3,863,788
France		$1,677,652	$3,827,950	$5,505,603	$2,752,801
Germany			$2,241,204	$2,241,204	$2,241,204
United Kingdom		$5,287,044	$5,015,682	$10,302,727	$5,151,363
United States	$16,059,485	$23,075,616	$24,963,421	$64,098,522	$21,366,174
	$19,727,625	$33,009,242	$42,760,939	$95,497,806	

8. Save the report as **Manual Cross-Tab 1 Final.rpt** and close it

Formatting Subreports

Consider the following report which is a list of countries with their Last Year's Sales totals and a list of Suppliers for that country. The report is limited to USA, Canada and Mexico.

First we need to open the container report (Last Year's Sales.rpt) and add the linked subreport (Supplier Listing), then we'll learn some formatting tips and tricks to make this report more flexible and user friendly

❖ Exercise 7.1 - Adding and Linking a Subreport

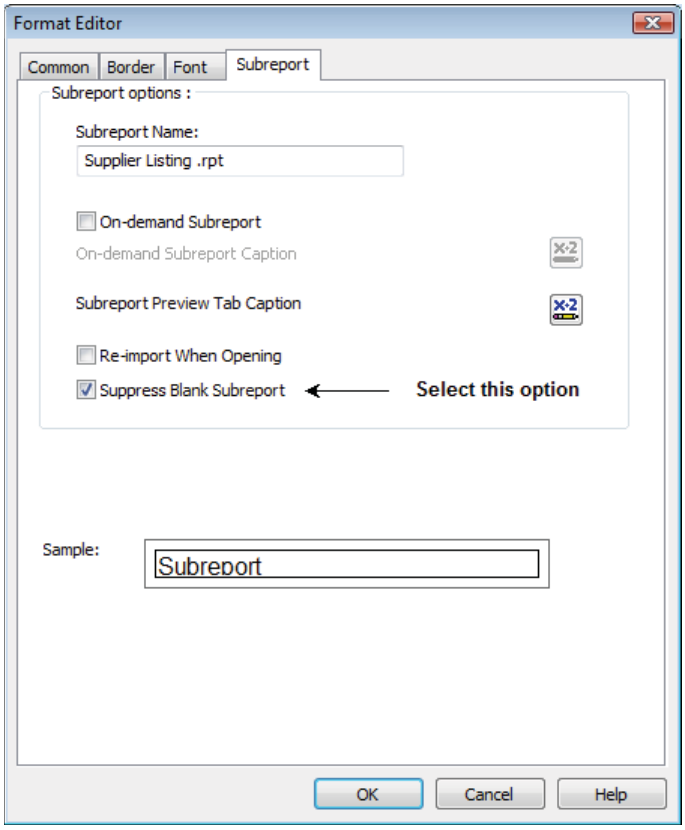
1. Open the **Last Year's Sales.rpt** report from the **CR2008 D3 Class Resources\L07 - Tips and Tricks** directory
   *This report and the subsequent subreport use the Northwind 2008 Microsoft Access database located in the **CR2008 D3 Class Resources\Databases** directory.*

2. Insert a subreport into **Group Footer #1b** using the **Supplier Listing.rpt** from the **CR2008 D3 Class Resources\L07 - Tips and Tricks** directory. Link the two reports on the **Country** field

3. Re-size the subreport object from **Design** to be approximately 5" long. Suppress the subreport's **Report Footers** and then Preview

4.

5. Now let's assume we don't want to see the subreport when there are no suppliers. We can simply format the subreport to *Suppress Blank Supreport* in the following dialog:



6. Click **OK**. Preview the report

7. This achieves one result but a message saying there were no suppliers would be a more user friendly report. First un-check the *Suppress Blank Subreport* option in the **Format Editor** for the subreport

8. Edit the subreport and insert a new **Report Header** section. Add a text box with the following text: "**There are no suppliers in** " and insert the parameter field **{?Pm-Customers.Country}**. If you insert the **Supplier.sCountry** field it will be blank as no records are being read. Make the text object Bol with a 12pt font. Re-size to about 4 inches and align to match the "**Suppliers for ...**" text box

You now need to suppress each report heading as follows:

Report Header a (which lists suppliers) should be suppressed when:
IsNull({Suppliers.SupplierName})

Report Header b (There are no suppliers) should be suppressed when:
Not IsNull({Suppliers.SupplierName})

Report Header c and **Details** sections with the formula:
IsNull({Suppliers.SupplierName})

9. Save the report as **LYS and Supplier.rpt** and preview. Your report should look as follows:

Parameter Tips and Tricks

Using dynamic parameters offers a great solution for keeping pick lists fresh and up-to-date. One problem with dynamic parameters is it is impossible to add "bogus" values to the pick list using the Create New Parameter or Edit Parameter dialog windows. As a matter of fact Crystal Reports doesn't offer this functionality at all. However, using a little SQL we can perform an easy work around solution.

Another problem is we cannot add filters to exclude data in the pick list. Again, this can only be accomplished through writing our own SQL command.

The exercise report Customer Segmentation Param.rpt has a dynamic parameter called States. It is getting its pick list values from the StateProvinceName field contained in the view the report is using. The report itself is filtering on United States customers and the StateProvinceNames equal to the select values in the States parameter.

The problem is this, the States parameter is displaying StateProvinceNames values for all countries and not just the United States StateProvinceNames values.

To address this issue, we are going to create an SQL command which gathers on United States State names and then re-point the the States parameter to use a field from the SQL command.

Lastly, we want the end-user to have the ability to pick all states for the report.

❖ Exercise 7.2 - Creating an SQL Command for a Dynamic Parameter Pick List

1. Open the **Customer Segmentation Param.rpt** report from the **CR2008 D3 Class Resources\L07 - Tips and Tricks** directory

2. Preview the report and refresh the parameter values. Notice the non-United States values in the pick list

3. Return to **Design** view and then select the **Database** menu option and choose **Database Expert**

4. The Database Expert dialog window opens. Open the **AdventureWorks** data connection and then Double-Click on the **Add Command** item

5. The Add Command to Report dialog open. In the **SQL Query** text box, enter the following query:

```
SELECT Person.StateProvince.Name as State
FROM   Person.StateProvince INNER JOIN Person.CountryRegion ON
       Person.StateProvince.CountryRegionCode =
       Person.CountryRegion.CountryRegionCode
WHERE (Person.CountryRegion.Name = N'United States')
```

This query pulls the StateProvinceName field and renames it State. It also filters to only include States from the United States. We don't have to worry about duplicate values or a sort order because the States parameter will take care of that as part of its normal functionality.

6. After entering the code, click **OK** to close the **Add Command to Report** dialog window.
   Crystal Reports will validate the query by submitting it to the database. At this point, if there are errors a database error window will open with a message

7. Rename the command to **StateLookup**
   HINT: Double-Click on the Command slowly twice or highlight the Command and press the F2 keyboard key

8. Select the **Links** tab and make sure there are no joins between the view and the SQL command and then click **OK**
   This will generate warning messages, but ignore them

9. Edit the **States** parameter by opening the **Edit a parameter and list of values** dialog window. Under *Choose a Data Source*, select the **New** option

10. In the **Value** grid set the **Value** item to the **State** field under the **StateLookup** table (command), change *Allow multiple values* to **True** and then click **OK**

11. Refresh the report and prompt for new values. Notice only United States regions are being displayed with no duplicates and in alphabetical order. Leave or confirm only **Arizona**, **Florida**, **Georgia** and **Oregon** are in the *Selected Values:* list box and then click **OK**

❖ Exercise 7.3 - Adding a Non-Database Value to a Dynamic Parameter Pick List

1. Return to **Design** view and then select the **Database** menu option and choose **Database Expert**

2. The Database Expert dialog window opens. From the *Selected Table:* list box, Right-Click the **StateLookup** command and choose **Edit Command**

3. Add two new lines at the top of your SQL Command query:

```
SELECT '*ALL*' as State
UNION
```

Be sure to use only single quotes around the *ALL* value.

A Union Join performs an append query and essentially adds two lists together. Different databases have different rules and some database don't support this type of join

4. Refresh the report and prompt for new values.
   Notice the "bogus" value is listed at the top. This is why we added the asterisk prefix. Without it "All" would be placed after Alaska.

5. Remove all *Selected Values:* and add only the ***ALL*** value then click **OK**
   Notice no records were returned. We need to edit the record selection formula next.

6. Edit the Record Selection Formula to match the following filter:

   {vIndividualCustomer.CountryRegionName} = "United States" and
   If {?States} = "*ALL*" then True
   Else {vIndividualCustomer.StateProvinceName} = {?States}

7. Check for errors and then save and close the formula. Preview the report and choose the **Refresh Data** button
   You should get 7843 records returned

Conditional Formatting

In this section you will look at different formatting options to make your report easier to read. You will learn how to alternate background color on an entire section and a technique to only color the data part of the section when that is needed.

❖ Exercise 7.4 - Conditionally Formatting a Section Background Color

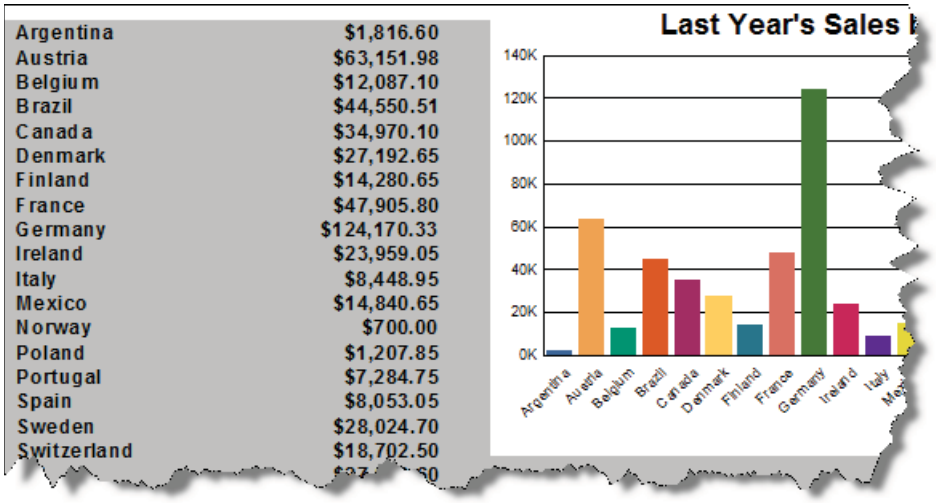
1. Open the **Last Year's Sales by Country.rpt** report from the **CR2008 D3 Class Resources\L07 - Tips and Tricks** directory
   *This report and the subsequent subreport use the Northwind 2008 Microsoft Access database located in the **CR2008 D3 Class Resources\Databases** directory.*

2. You are now ready to conditionally format the **Group Footer #1** to alternate background color. In the **Section Expert**, for **Group Footer #1**, choose **Color**; **Background Color** and click on the conditional formatting **X+2** button. Enter the following formula:

   If GroupNumber Mod 2 = 0 then crSilver Else crNoColor

   NOTE – If you do not add the else statement alternate lines will have a background of black which is the default.

3. Save your report as **Last Year's Sales by Country Final.rpt**. Your report should now look as follows:



❖ Exercise 7.5 - Conditionally Alternating Background Color for Data Only

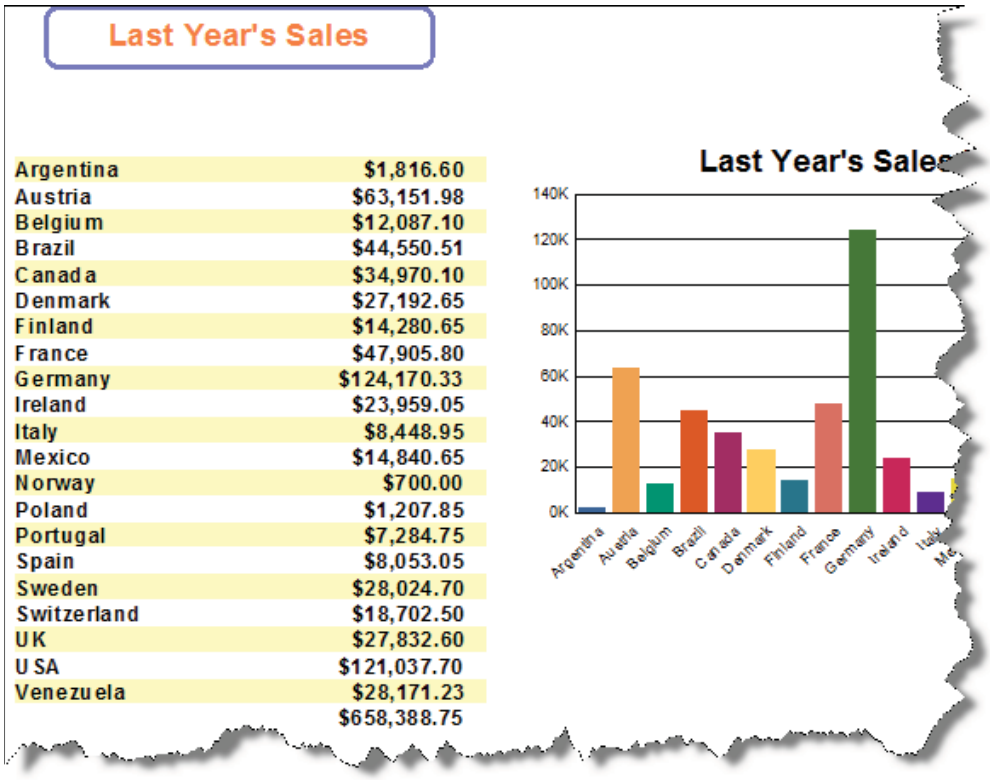
1. Notice the background shading looks odd with the chart. You are now going to see how to make the background color the width of the data. In the **Section Expert** delete the background color formula.

2. Insert a **Group Footer** below the **Group Footer 1**. Move the name and summary field to the second **Group Footer 1b** (or swap section order using your mouse). In **Group Footer 1a** draw a box object the width of the data and the height of the data in **Group Footer 1b**. Format the box object with a background fill and border (same color)

3. Open the **Section Expert**, make sure **Group Footer 1a** is selected and then create a conditional suppression formula with the following formula:

   GroupNumber Mod 2 = 0

4. Still working in the Section Expert, check on the *Underlay Following Sections* property for **Group Footer 1a**. Click **OK** when finished and preview. The report should look similar to the following:



Argentina	$1,816.60
Austria	$63,151.98
Belgium	$12,087.10
Brazil	$44,550.51
Canada	$34,970.10
Denmark	$27,192.65
Finland	$14,280.65
France	$47,905.80
Germany	$124,170.33
Ireland	$23,959.05
Italy	$8,448.95
Mexico	$14,840.65
Norway	$700.00
Poland	$1,207.85
Portugal	$7,284.75
Spain	$8,053.05
Sweden	$28,024.70
Switzerland	$18,702.50
UK	$27,832.60
USA	$121,037.70
Venezuela	$28,171.23
	$658,388.75

5. Save (**Last Year's Sales by Country Final**) and close the report

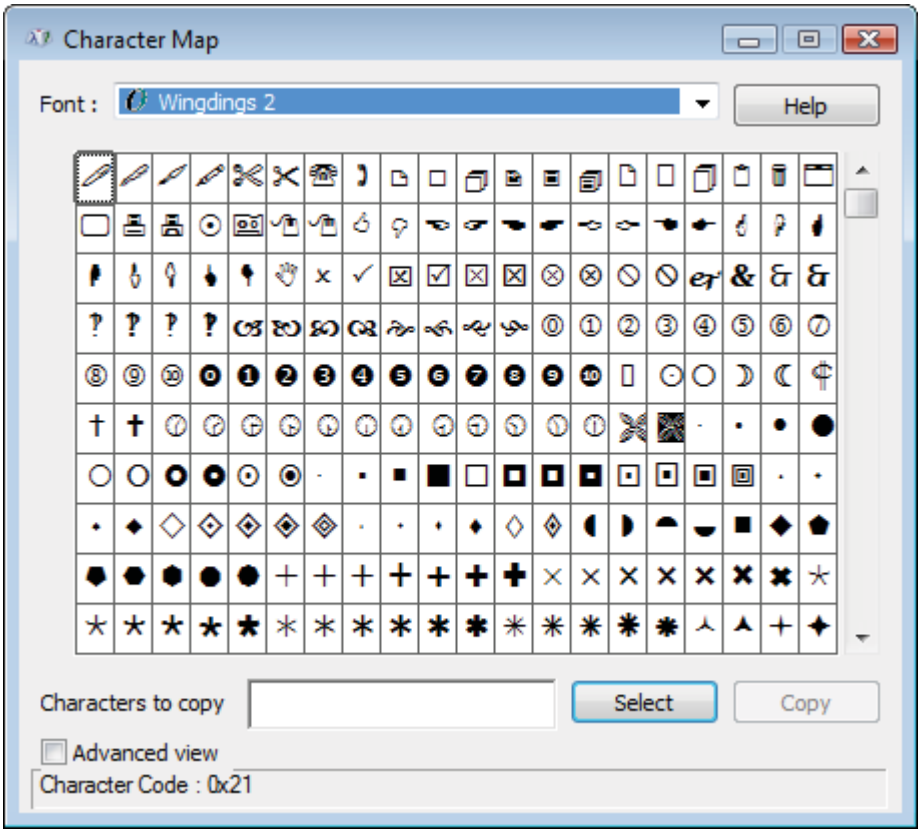
©2008 Vision Harvest, Inc.

Using Fonts to Enhance Readability of your report

You have used conditional formatting to highlight amounts that are over a certain criteria but imagine you would like to simply check mark some records and mark with an X others. To do this you will use formulas and the character map.

Windows Character Map

To work out the character value for a symbol you would like to use in your report you can use the Character Map from Windows. This is available from Accessories|System Tools and looks as shown below:



You choose the symbol font and character you would like to use. You copy and then paste the symbol into Crystal Reports. You then need to format the object with the correct font such as Wingdings.

❖ Exercise 7.6 - Add Special Characters using the Character Map Dialog

1. Open the **Country LYS.rpt** report from the **CR2008 D3 Class Resources\L07 - Tips and Tricks** directory
   *This report and the subsequent subreport use the Northwind 2008 Microsoft Access database located in the **CR2008 D3 Class Resources\Databases** directory.*

2. You are now going to create a formula that places an ☒ if the country's last year's sales are below $75,000 and a check mark ☑ if it is above. Open your **Character Map** (Accessories|System Tools); select **Wingdings 2** as your font and locate the check mark and cross. Now go to your report and create a formula called **Top Sales** with the following:

   If Sum ({Customers.LastYearsSales}, {Customers.Country}) < 75000 Then

3. Now you need to select the ☒ from the character map and paste it into your formula – you will see a character in this case Q – you need to put this in double quotes. Now do the same thing for the ☑. Your formula should read as follows:

   If Sum ({Customers.LastYearsSales}, {Customers.Country}) < 75000 Then "Q" Else "R"

4. Place the formula field on the report and your report should look as follows:

	Total	Quarter 1				
		Total		Jan	Feb	M
Total	14,962,555 0.00% 0%	6,603,706 0.00% 0%	Monthly Pct: Yearly Pct:	1,534,433 23.24% 10.26%	2,586,832 39.17% 17.29%	2,482 37.59% 16.59%
Europe	4,031,531 0.00% 0%	1,723,911 0.00% 0%	Monthly Pct: Yearly Pct:	378,978 21.98% 9.40%	821,994 47.68% 20.39%	522 30.33% 12.97%
North America	10,931,024 0.00% 0%	4,879,795 0.00% 0%	Monthly Pct: Yearly Pct:	1,155,455 23.68% 10.57%	1,764,838 36.17% 16.15%	1,959 40.16% 17.93%

5. Select the field and format the field to have Wingdings 2 Font, 12pt.

6. Save the report as **Country LYS Final.rpt** and close it

©2008 Vision Harvest, Inc.

NOTES

©2008 Vision Harvest, Inc.

NOTES