What's Inside this Guide:

Discern Explorer®

Layout Builder Tutorial Part 2 of 2

CERNER

Table of Contents

Discern Explorer Layout Builder Tutorial

Code Level

This tutorial is geared for clients on the *Cerner Millennium*® Cumulative Support Package 2007.18 release of Discern Visual Developer (DiscernVisualDeveloper.exe), also referred to as DVDev. Much of the functionality discussed in this document is also available for earlier releases, but there are significant differences between 2007.18 and some of the earlier versions. All references to help files and functionality also pertain to the 2007.18 production release.

A basic understanding of using DVDev and the *Discern Explorer*® (CCL) programming language to create programs and reports is assumed.

General Information

Layout Builder is a component that allows visual development of layout sections used to create *Discern Explorer* reports using the Report API. You can link the layout sections to a *Discern Explorer* select command, a program that prompts users for input, a driver program, or a record structure that contains information that needs to be displayed or printed. For each layout section, two *Discern Explorer* subroutines are created by Layout Builder. A third subroutine is created in certain scenarios. More information on the subroutines is provided in Appendix A.

This tutorial contains instructions that will guide you through using the common functionality of Layout Builder. Validate the steps listed throughout this document in a non-production environment to help you gain an understanding of Layout Builder.

Using the Graph Tool

Most of the Layout Tools were discussed in the Creating a New Free Form Label section in *Discern Explorer* Layout Builder Tutorial Part 1. However, the Graph Tool was not covered in that section because its functionality is normally not used in the context of a Free Form Label.

The Graph Tool enables you to display the output of a *Discern Explorer* query in graphical format in a layout section. The following common graph formats are supported:

- XY (Scatter)
- Line
- Column
- Bar
- Pie
- Line – Column
- Line – Column on 2 Axes

Graphing Data From a Single Query

Often, it is more useful to display data in a graphical format rather than in a textual format. For example, suppose we wanted to see if there was a pattern to the number of encounters registered on each day of the week. The number of encounters registered on each day could be reported as text but a pattern might be easier to detect if the data is displayed as a graph.

The following example creates a query that returns the number of encounters that were registered on each day of the week over a date range entered by the user at run time.

1. Using DVDev, from the File menu, select New.
2. From the File Type list, select Layout Program.
3. In the Program Name box, enter **1_*your_initials*_Example_Graph** and click OK. The New Layout Program dialog box opens.
4. Verify that the Standard Layout option for the Report Layout and PostScript option for the Output Type are selected and click Next. The Paper Size dialog box is displayed.
5. Keep the defaulted values for the Paper Size and click Finish. The previous two dialog boxes, New Layout Program and Paper Size are used to populate the basic properties of the layout. These properties can be reviewed at any time by selecting

Doc ID: A1001001A08G07B34433A94851

Report Properties from the Edit menu. A layout with a single section named DetailSection is created.

6. From the Tools menu, select Prompt Builder. The Prompt Builder dialog box opens with a single control for an output device.

7. Click Add to add a second prompt to ask for a start date from the user.

8. On the General Tab, set the options as follows:

Prompt Display: Select the Beginning Date

Prompt Name: Sdate

Control Type: Date Time

Prompt Type: String

9. On the Date / Time tab, verify that the Date Only option is selected and the Command Line Format is DD-MMM-YYYY.

10. In the Anchor date & time area on the Calculate Default tab, uncheck the Current date & time option. Enter a negative value in the Day or Month control to make the default start date far enough in the past to allow a query to qualify encounters that were registered on several days of the week.

The negative value you enter should depend on the amount of data that exists in your environment. If you have a lot of data, set the Day field to -7 to make the default start date one week in the past. If you do not have a lot of data, or if the encounters you have were not registered very recently, you may need to set the Month field to -3 to make the default start date three months in the past.

11. Click Add to add a third prompt to get an end date from the user.

12. On the General Tab, set the following:

Prompt Display: Select the Ending Date

Prompt Name: Edate

Control Type: Date Time

Prompt Type: String

13. On the Date / Time tab verify the Date Only option is selected and the Command Line Format is DD-MMM-YYYY.

14. Click Save to save the prompt form and close the Prompt Builder dialog box.

15. From the Tools menu, select Query Builder.

16. Click Add. The Add Query dialog box opens.

17. Name the query **Get_Encounters_By_Day**. Verify that the Associate Layout option is selected, and click OK to add the query. The Discern Query Builder dialog box opens.

18. On the Tables tab, select the ENCOUNTER table.

19. On the Fields tab:

    - Select the REG_DT_TM field (registration date and time)
    - Add an expression to get the name of the day when the encounter was registered using the following text:

```
Reg_day=format(e.reg_dt_tm, "WWWWWWWWW;;d")
```

    - Add an expression to display the number of the day when the encounter was registered using the following text:

```
Reg_day_num=weekday(e.reg_dt_tm)
```

Using the expressions created above, Reg_Day is set to the text string that represents the name of the day. For example, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, or Saturday. The text string flexes based on the language and location. Reg_Day_Num is set to 0, 1, 2, 3, 4, 5, or 6. Using these two expressions enables you to sort the output in the order the days occur in the week using Reg_Day_Num and to display the name of the day using Reg_Day.

On the Qualifications tab, add the following code to select encounters that were registered between the dates the user selected at the prompts:

```
Where e.reg_dt_tm between cnvtdatetime($Sdate) and
cnvtdatetime(concat($Edate, " " , "23:59:59"))
```

Notice in the concat statement above there is one and only one space between the quotation marks that follow the $Edate symbol. Since the Command Line Format for the Edate prompt is DD-MMM-YYYY, and the Prompt Type is string, the date which the user selects at the prompt is passed to the layout program as a string in DD-MMM-YYYY format. Passing that string to the cnvtdatetime() function returns a date time value that represents the very earliest time on that date. Using the concat() function to append a single space and 23:59:59 to the date the user selected at the prompt causes the cnvtdatetime() function to return a date time value that represents the very last possible time on the date the user selected.

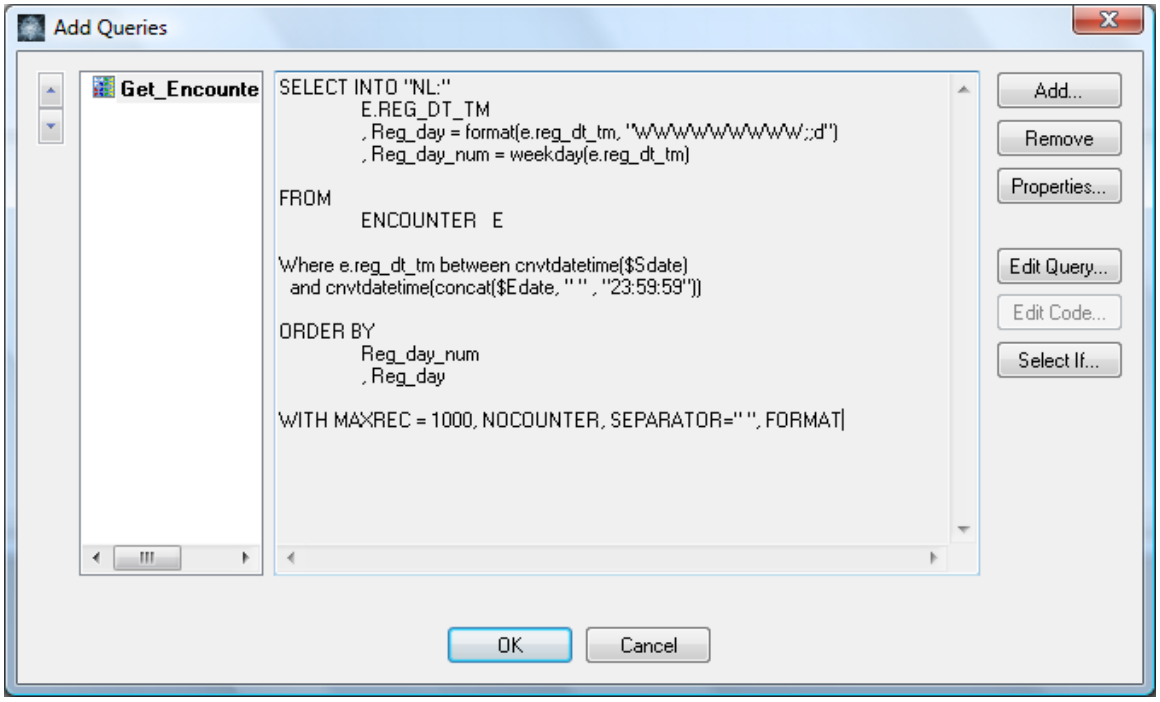
20. On the Sort Tab, sort on the REG_DAY_NUM expression and then on the REG_DAY expression.

21. On the Control Options Tab:

    - Select the Into "NL:" option.
    - For testing, set the Max Records: to 1000 to limit the query to only reading 1000 records.

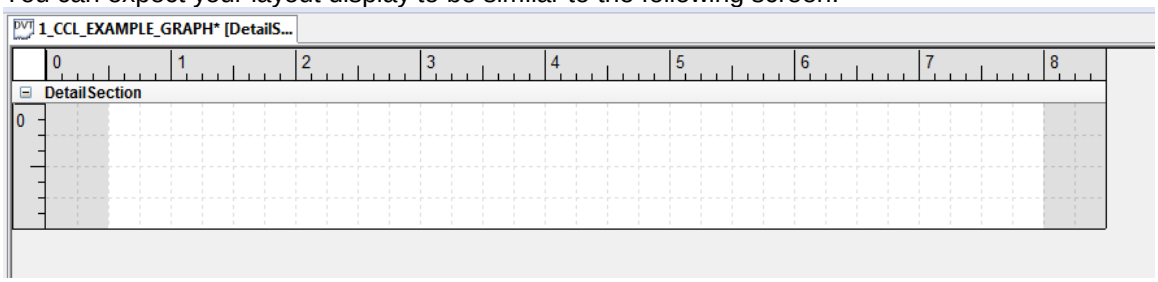
When setting the Max Records, select a number large enough to allow you to return encounters that were registered on several days of the week, but small enough to prevent an extremely long query from running if you make a mistake while creating this example. If this was a program, you would eventually move to production and would want to remove the Max Records value after you had verified that the query was efficient.

22. Click Close to close Query Builder. Your Add Queries dialog box should look similar to the following example:



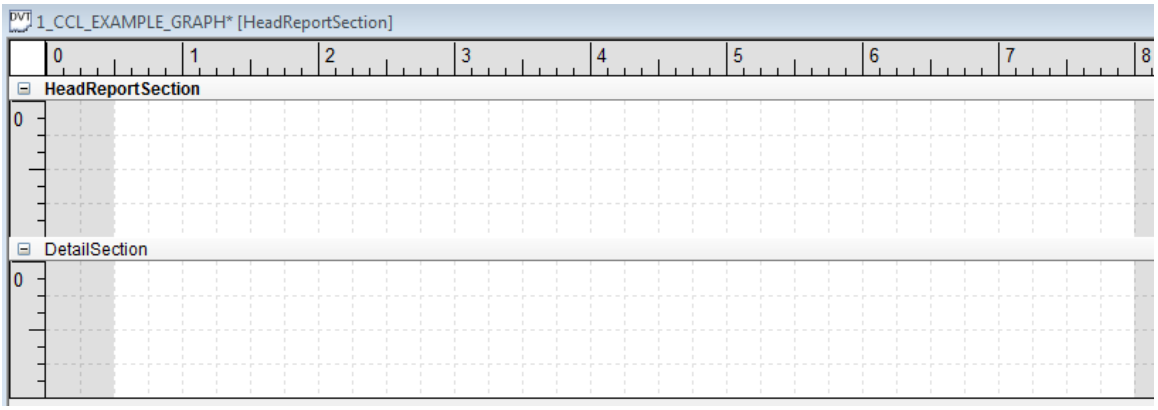
23. Click OK to close the Add Queries dialog box. The Layout Work space dialog box opens. The Select/Modify Sections dialog box opens.

You can expect your layout display to be similar to the following screen:



We want to create a section that renders only once at the beginning of the report to display the report title,

24. From the Select/Modify Section of the Workspace dialog box, select the Head Report option. Click Yes to create a new layout section. A new section called HeadReportSection is displayed.



25. Double-click on the HeadReportSection title bar. The Properties dialog box is displayed.

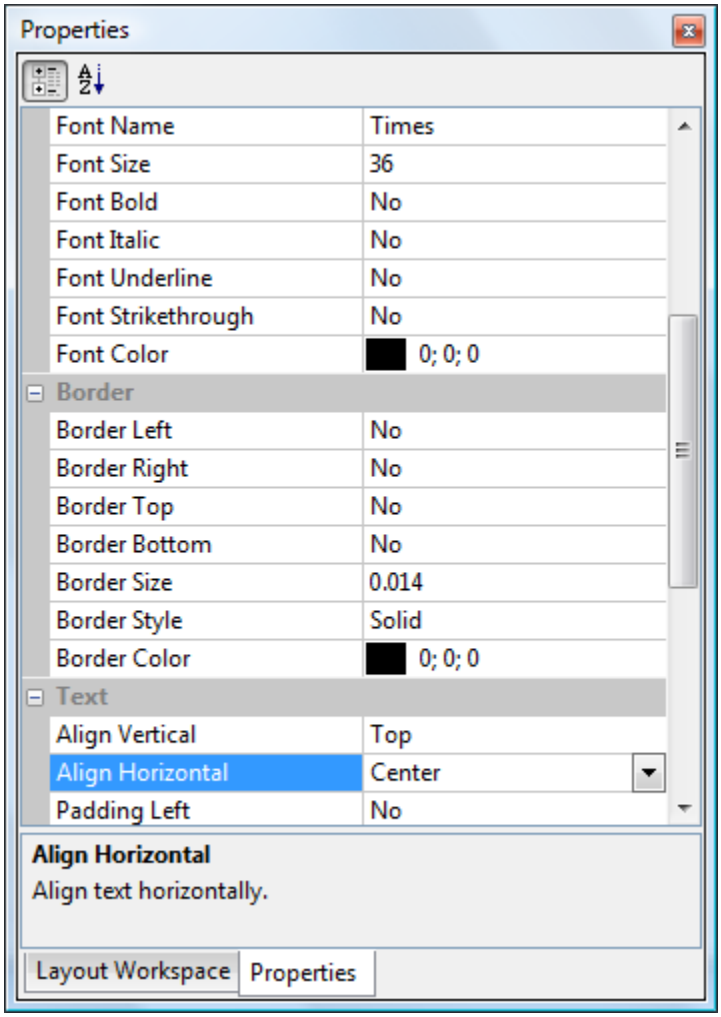
26. In the Properties dialog box, modify the name to **TitleLayoutSection**.

27. Use the Label Tool  to add the report title **Encounters Registered by Day** to the TitleLayoutSection.

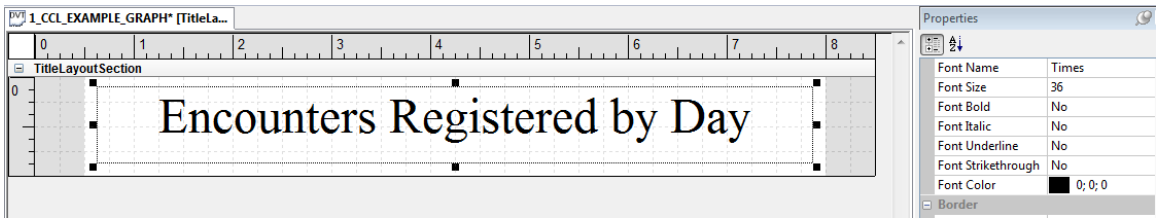
28. Use the Formatting toolbar or the Properties dialog box to set the following items:

    - The font to Times
    - The font size to 36
    - Center the text

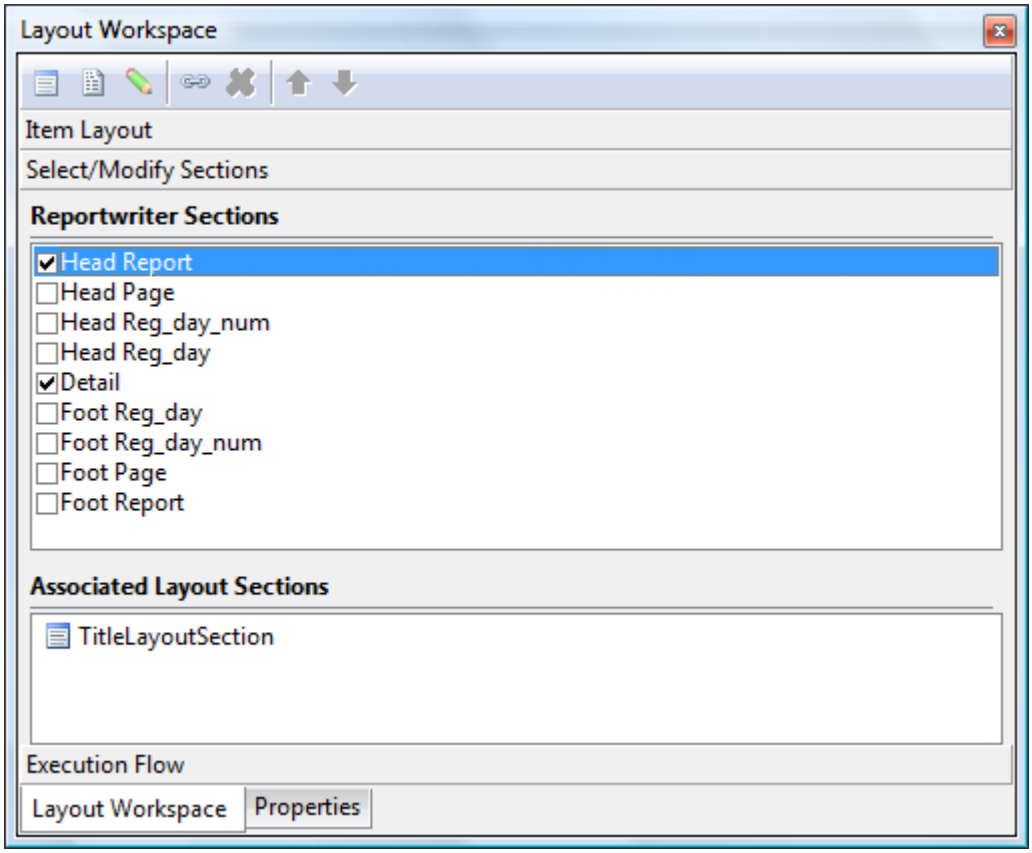
If the Formatting toolbar is not displayed, from the View menu, select Toolbars > Formatting.



Verify the Label Item is tall enough and wide enough to display the entire text.



29. Select the Layout Workspace dialog box and click on the word Head Report, not the checkbox option. Verify that the TitleLayoutSection is listed in the Associated Layout Sections.



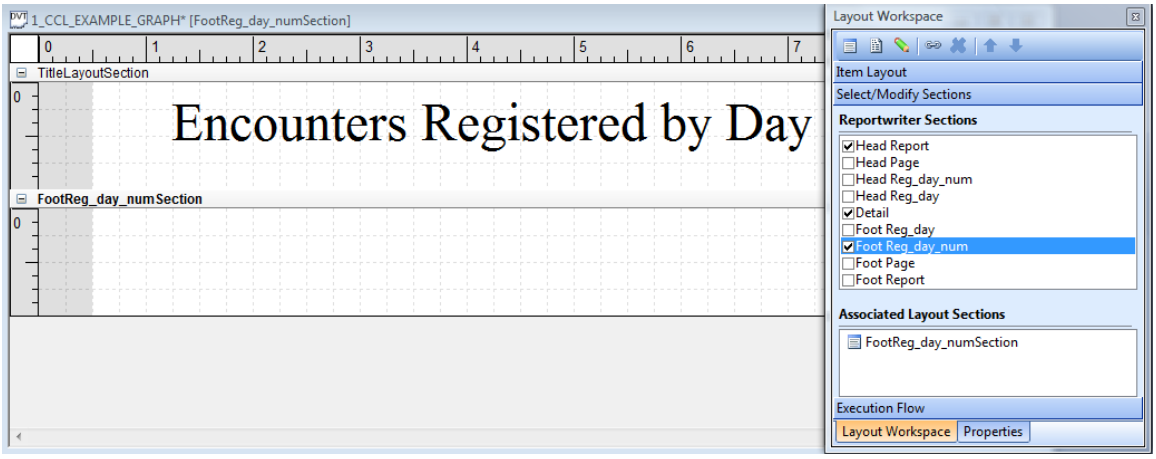
This report calculates the number of encounters registered in a day, but does not display any detailed information from the specific day. As we do not need to render any information for a specific day, we can delete the DetailSection.

30. Click on the DetailSection title bar.

31. Select Remove > Section from the Edit menu. Click Yes to verify that you want to remove the section.

    The Foot Reg_day Reportwriter section is at the end of the logical grouping for the Reg_day and is where the number of encounters can be calculated. We need to create a section associated to the Foot Reg_day Reportwriter section to report on the calculated number of encounters registered in the day.

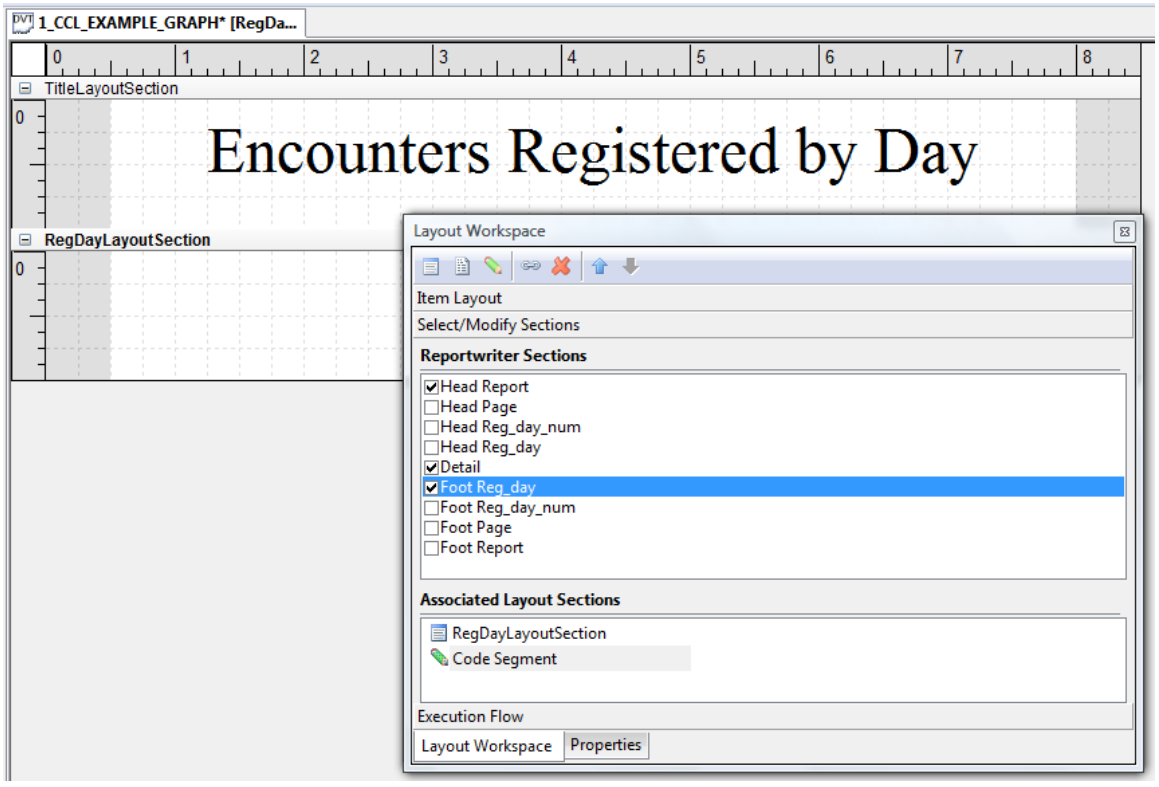
32. From the Select/Modify Section of the Workspace dialog box, select the Foot Reg_day Reportwriter option. Click Yes to create a layout section associated to the Reportwriter section. FootReg_DaySection is created and listed in the Associated Layout Sections for the section.



33. Double-click the Foot Reg_DaySection title bar. The Properties dialog box is displayed.

34. On the Properties for the new section, modify the Name to **RegDayLayoutSection**.

35. Click the New Code Segment  button on the Layout Workspace toolbar. The code segment will be automatically inserted in to the Reportwriter Section. You can expect your layout to be similar to the following screen:

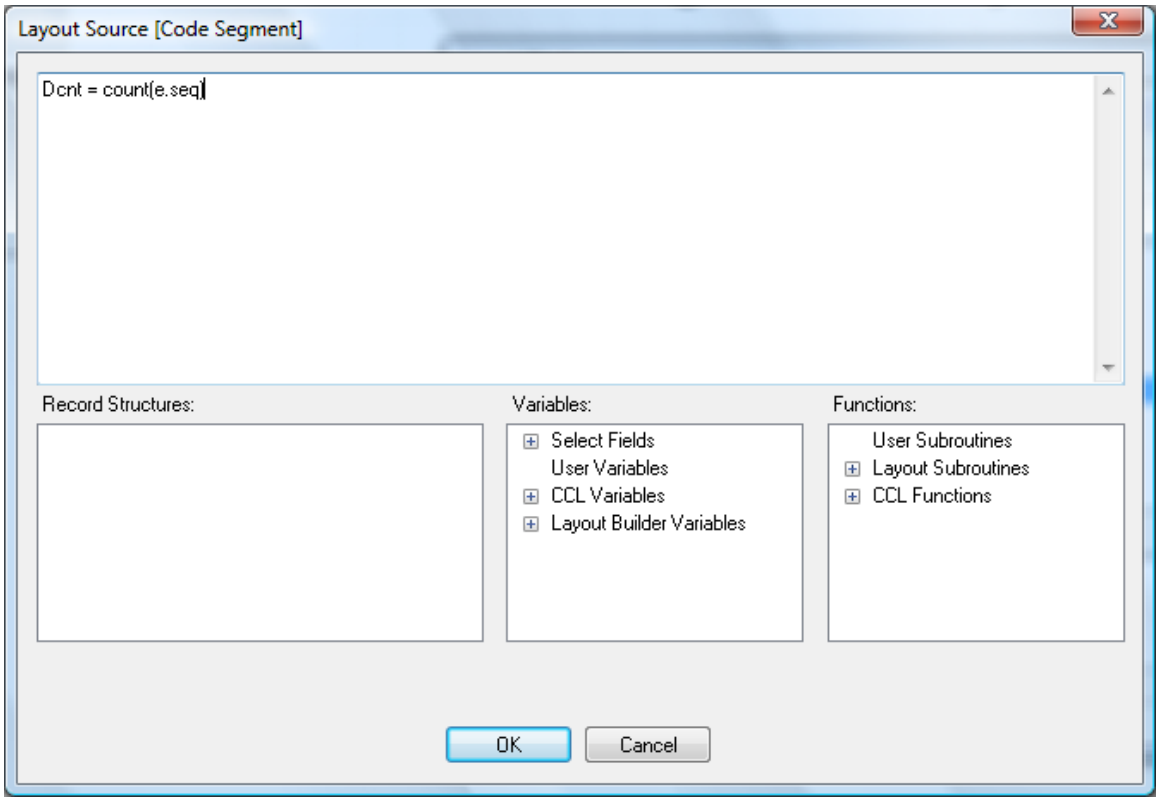


36. In the Associated Layout Sections, double-click the Code Segment to open the Layout Source [Code Segment] dialog and enter the following command in the source window:

Doc ID: A1001001A08G07B34433A94851

```
Dcnt = count(e.seq)
```

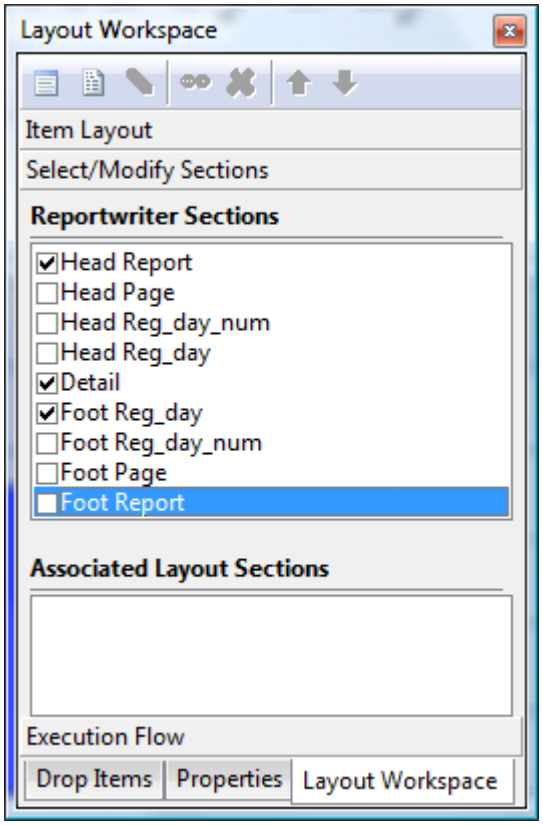
You can expect the Layout Source [Code Segment] dialog box to display similar to the following example:



Adding Dcnt = (e.seq) using the steps above creates a variable named Dcnt that is set equal to the number of encounters registered on each day. Since this command is placed in the Foot Reg_day reportwriter section, Dcnt is reset each time the day changes. Displaying Dcnt in the Foot Reg_day reportwriter section shows the number of encounters registered on each day in textual format.

37. Click OK to close the Layout Source [Code Segment] dialog box.

38. Click Code Segment in the Associated Layout Section and click the up Arrow button to move it to the top of the list. Code Segments and Layout Sections are executed in the order that they appear in the Associated Layout Sections list. Moving the Code Segment to the top of the list, causes the commands in the Code Segment to be executed before the RegDayLayoutSection is rendered.



39. Use the Text Tool  to place a text item in the RegDayLayoutSection.

40. In the Properties dialog box, name this item **Reg_day_label**.

41. Use the following command as the source for this item:

```
concat("The number of encounters registered on ", trim(Reg_day,3), ":")
```

Make sure to size the text item appropriately to display the entire string.

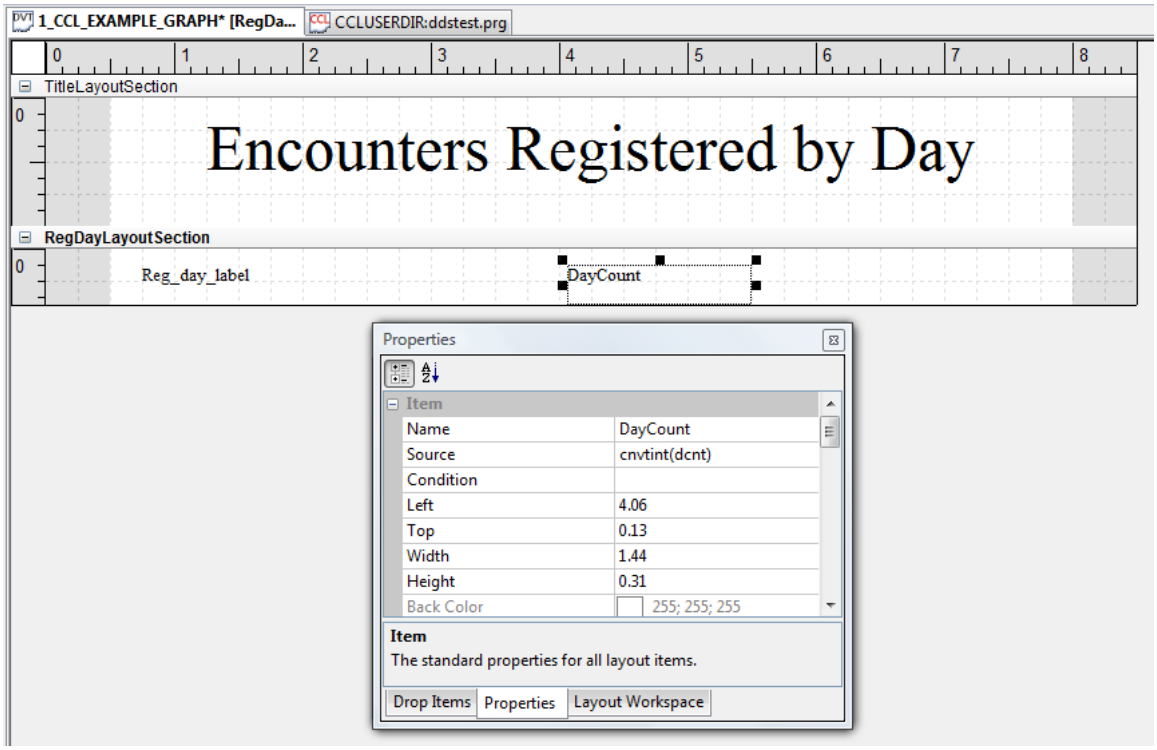
42. Use the Text Tool  to place another text item to the right of Reg_day_label in the RegDayLayoutSection.

43. In the Properties dialog box, name this item **DayCount**.

44. Use cnvtint(dcnt) as the source for this item.

45. Use the vertical resize to click and drag the bottom of the RegDayLayoutSection up to just below the two items you added above. You can expect your layout to be similar to the following screen:

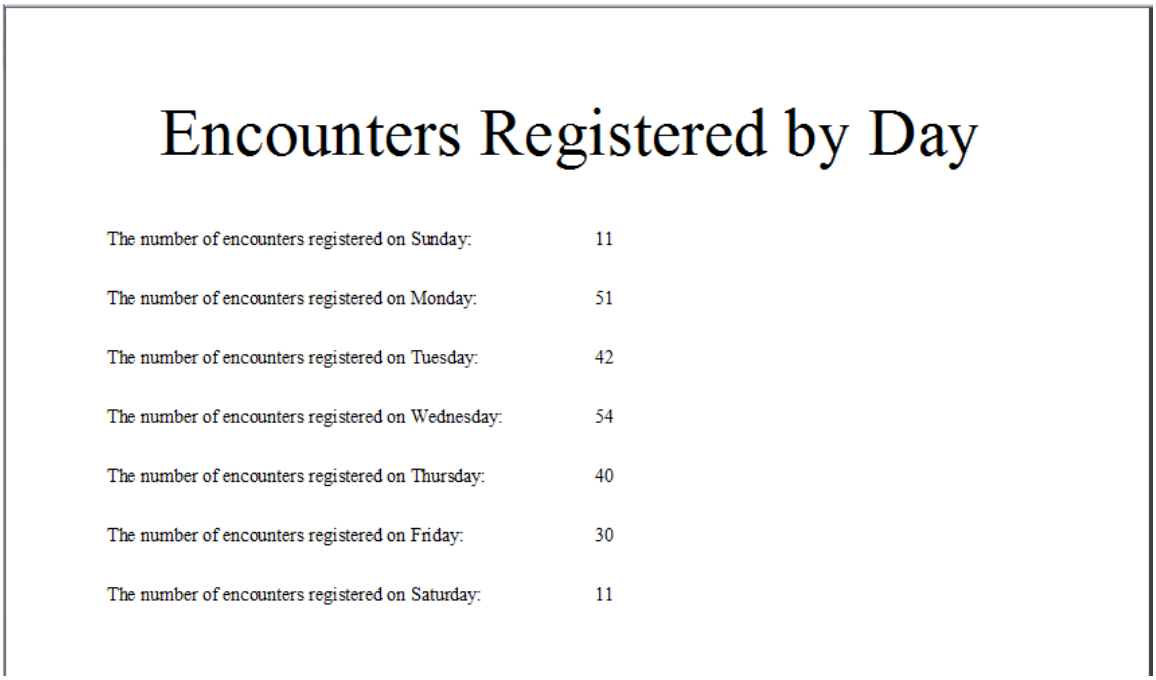
Doc ID: A1001001A08G07B34433A94851



46. From the Build menu, select Run "1_*Your_Initials*_Example_Graph", or press CTRL+F5 to execute your layout program.

47. Click Yes when prompted to save the layout. The prompt form opens.

48. At the prompt, use MINE for the output device and enter starting and ending dates that will return records that were registered over several different days. Your output should resemble the following screen:



Encounters Registered by Day

The number of encounters registered on Sunday:	11
The number of encounters registered on Monday:	51
The number of encounters registered on Tuesday:	42
The number of encounters registered on Wednesday:	54
The number of encounters registered on Thursday:	40
The number of encounters registered on Friday:	30
The number of encounters registered on Saturday:	11

The example above shows the output in textual format. We will now display the output in a graphical format.

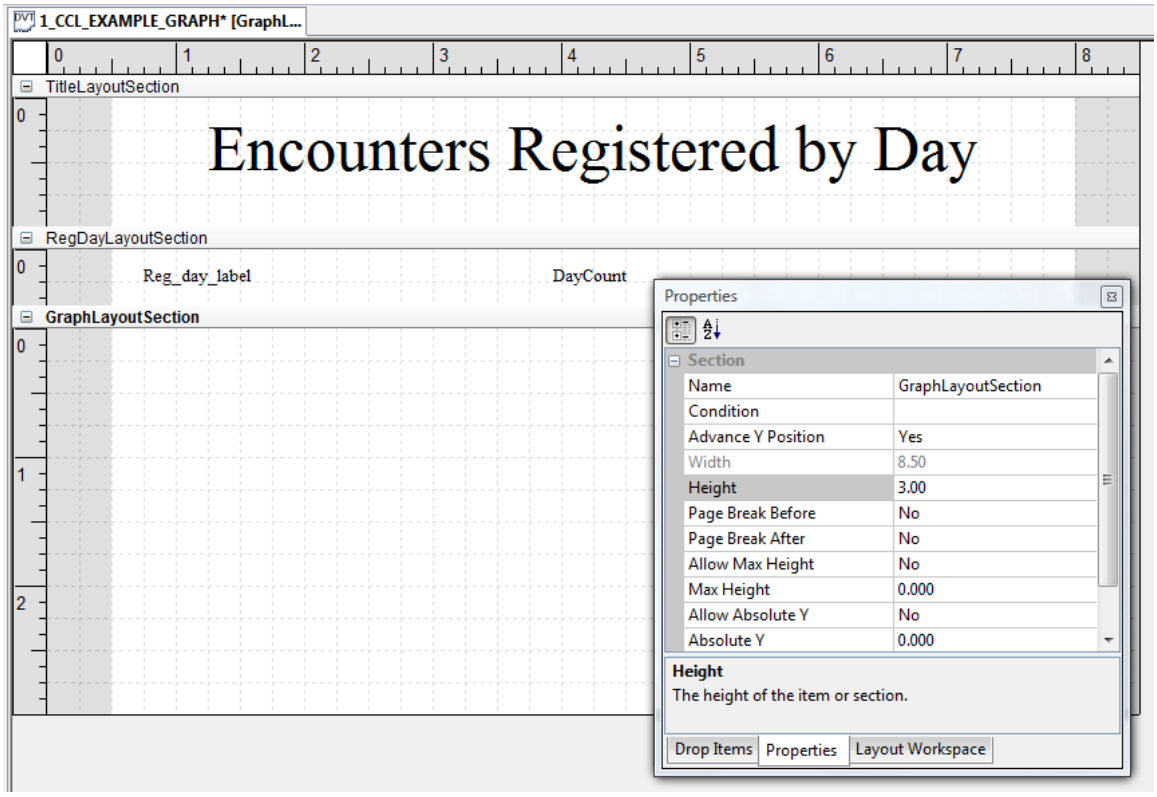
49. Close the output display.

50. From the Select/Modify Section of the Workspace dialog box, select the Foot Report Reportwriter section check box.

51. Click Yes to create a new layout section.

52. In the Properties for the new section modify the name to **GraphLayoutSection**.

53. Modify the height of the GraphSection to **3.0**. You can expect your layout to be similar to the following screen:



54. Click the Graph Tool  and then click and drag in the GraphLayoutSection to create a rectangular graph area that covers most of the section. The Graph dialog box will open.

The Graph Type tab on the Graph dialog box is used to determine the type of graph that will be generated.

55. Verify the Column Graph Type is selected on the Graph Type tab.

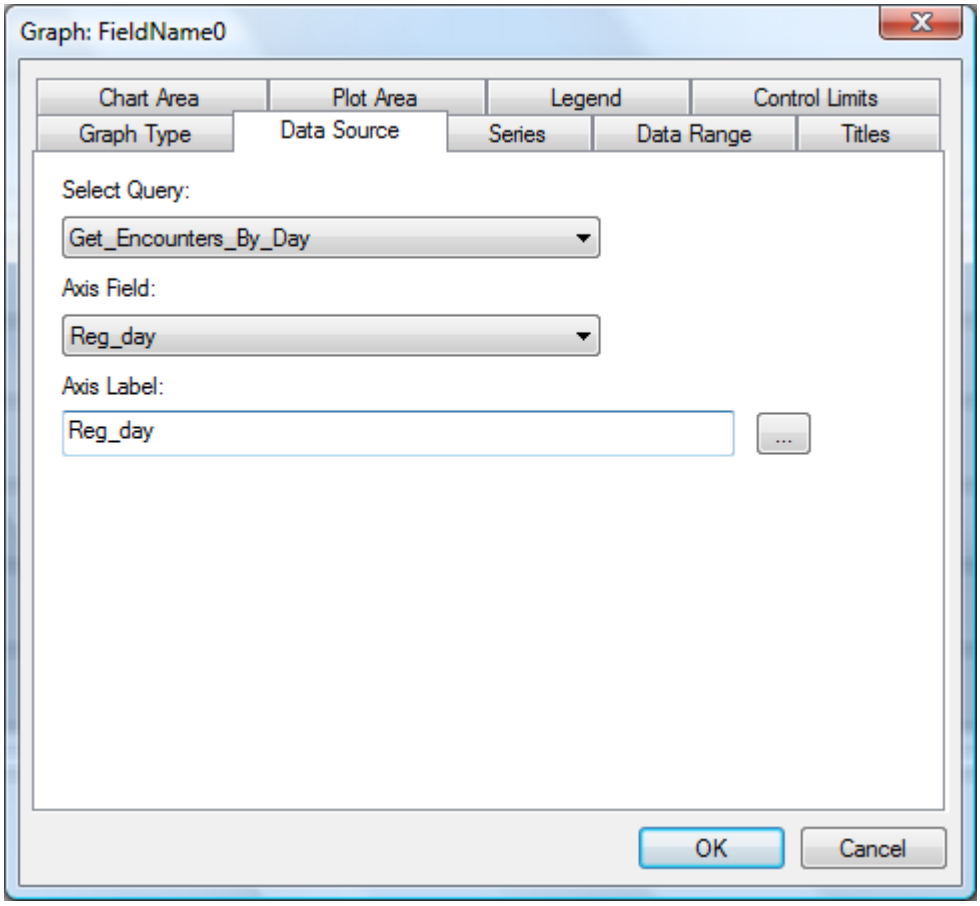
The Data Source tab on the Graph dialog box is used to select the query that supplies the data to be graphed and the major break points for graphing the data.

56. On the Data Source tab, use the Select Query list to select the Get_Encounters_By_Day query.

Because the Get_Encounters_By_Day query is sorted by Reg_day_num and Reg_day, those expressions are displayed as options in the Axis Field: list. The Axis Field determines the major break points for the graph. In our example we want to display a new column on the graph each time the day changes. Selecting Reg_day as the Axis Field generates a new column on the graph each time the registration day changes.

57. Select Reg_day as the Axis Field.

58. Enter **Reg_day** as the Axis Label.



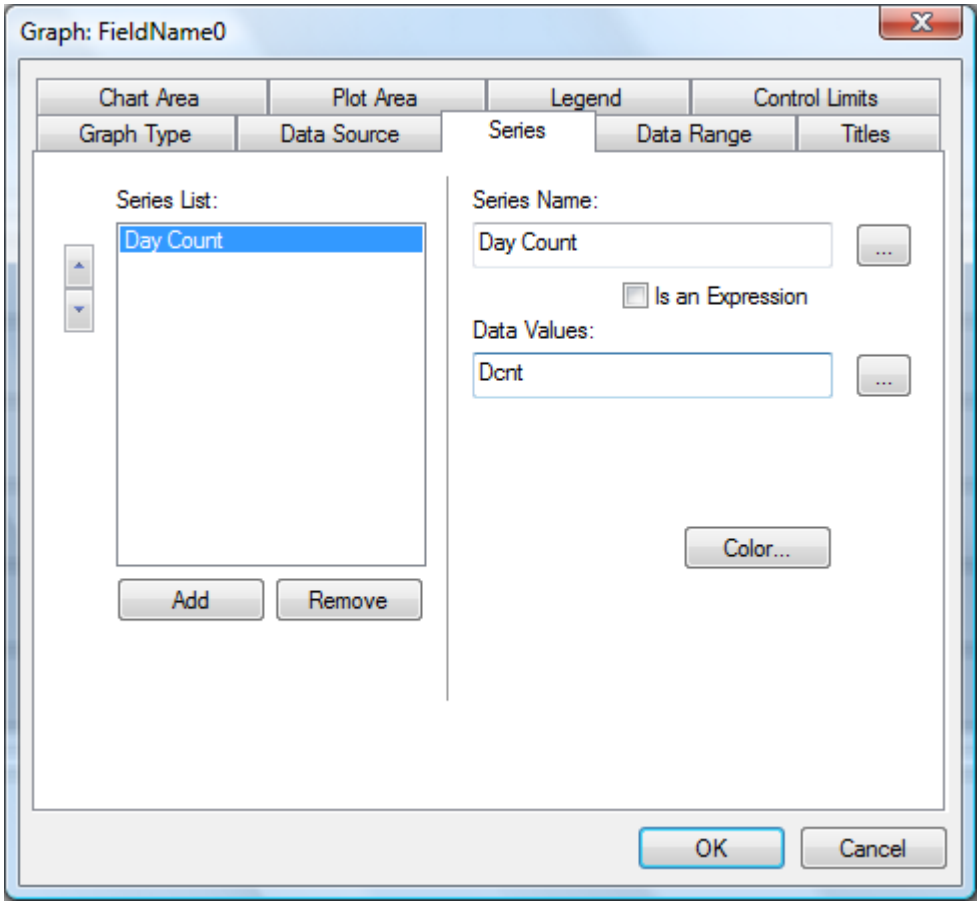
The Series tab is used to determine the data values to be graphed. By default a single series named Series1 is created on the Series tab. For our example we want the graph to show the number of encounters registered on each day of the week. That number is stored in the DCNT variable created earlier.

59. On the Series Tab, verify Series1 is selected in the Series List.

60. Modify the Series Name from Series1 to **Day Count**. The Series Name is displayed in the Legend if the Show Legend option is selected on the Legend tab.

61. Enter DCNT in the Data Values: control.

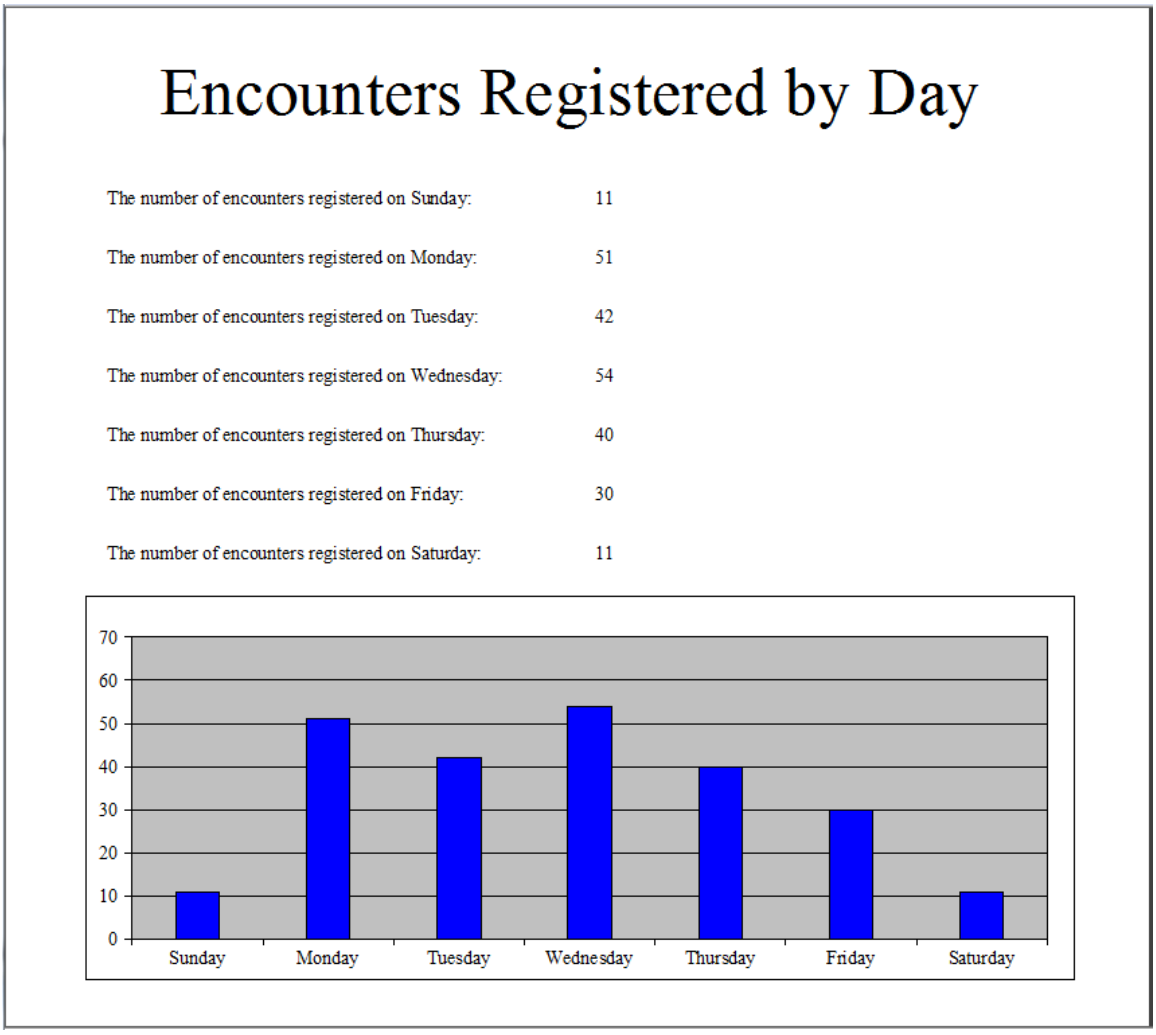
Doc ID: A1001001A08G07B34433A94851



62. Click OK to close the Graph Properties dialog box.

63. From the Build menu, select Run "1_*Your_Initials*_Example_Graph" or press CTRL+F5 to execute your layout program.

64. At the prompt, use MINE for the output device and enter starting and ending dates that will return records that were registered over several different days. Your output should resemble the following:



In the example above, the Y axis defaulted to a minimum value of 0 and a maximum value of 70. Your output will be different based on the data in your database. The Data Range tab is used to control the appearance of the Y axis on the graph, which you can use to control the size of your graph and for other cosmetic purposes.

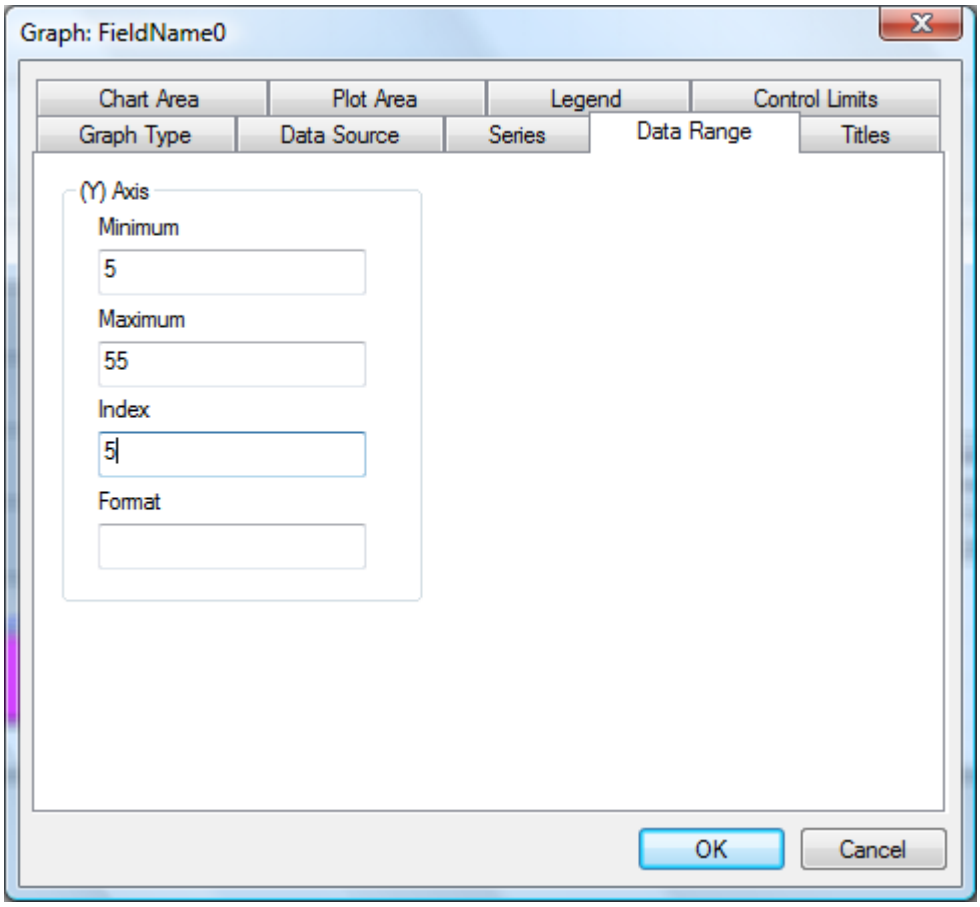
65. Close the output window to return to your layout.

66. Double-click the graph item to open the Graph dialog box.

67. On the Data Range tab:

- Set the Minimum value to a number that is slightly less than the lowest count your query is returning. For our example, we will set the Minimum value to 5.
- Set the Maximum value to a number that is slightly higher than the highest count your query is returning. For our example, we will set the Maximum value to 55.
- Set the Index value to a number that will change the breaks on the Y axis. For our example, we will set the Index value to 5.

Your Graph dialog box should look similar to the following example:



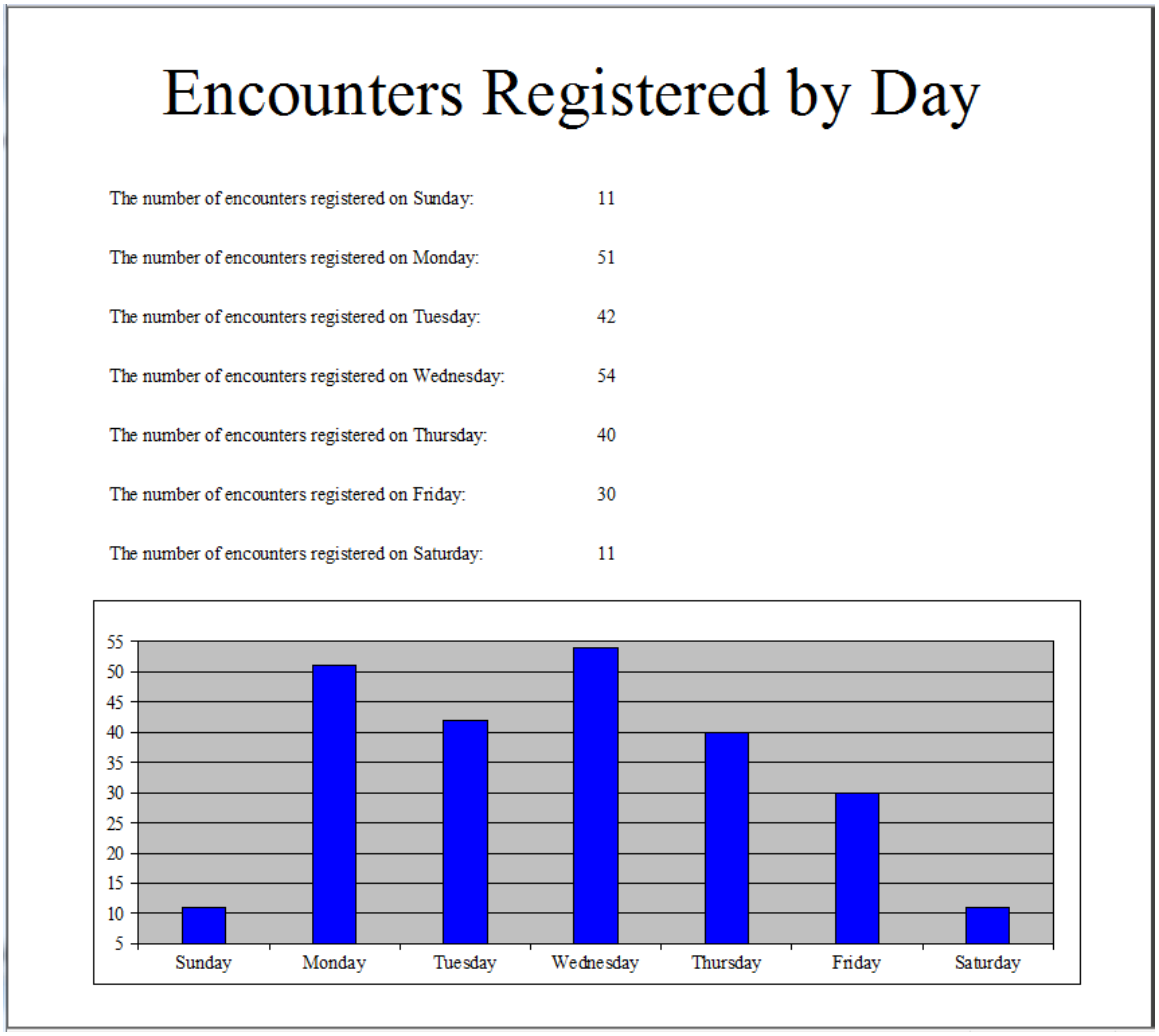
The minimum and maximum fields can be used to force the Y axis to always have a specific range. The Index field is used to control the scale of the intermediate lines on the graph. If these fields are left blank, Layout Builder generates the Y axis based on the data being graphed. Setting a minimum and maximum value causes the Y axis to be the same regardless of the data graphed. Using a minimum and maximum can be useful when you want the range of the graph to be the same across multiple executions. One caveat to consider when setting the minimum and maximum values is that if the data graphed exceeds the maximum value, or is less than the minimum value, the Y-axis will not be extended. For example, if the maximum is set to 50 and the data to be graphed is 55, the column would end at 50. If the minimum is set to 5 and the data to be graphed is 4, the column would not be displayed. The Format field can be used to format the index values that are displayed on the Y-axis. A display option can be entered directly in the Format field. For more information on display options see Discern Explorer Help (DiscernExplorerHelp.exe) > Contents Tab > Command Reference > Display Options.

68. Click OK to close the Graph Properties dialog box.

69. From the Build menu, select Run "1_*Your_Initials*_Example_Graph" or press CTRL+F5 to execute your layout program.

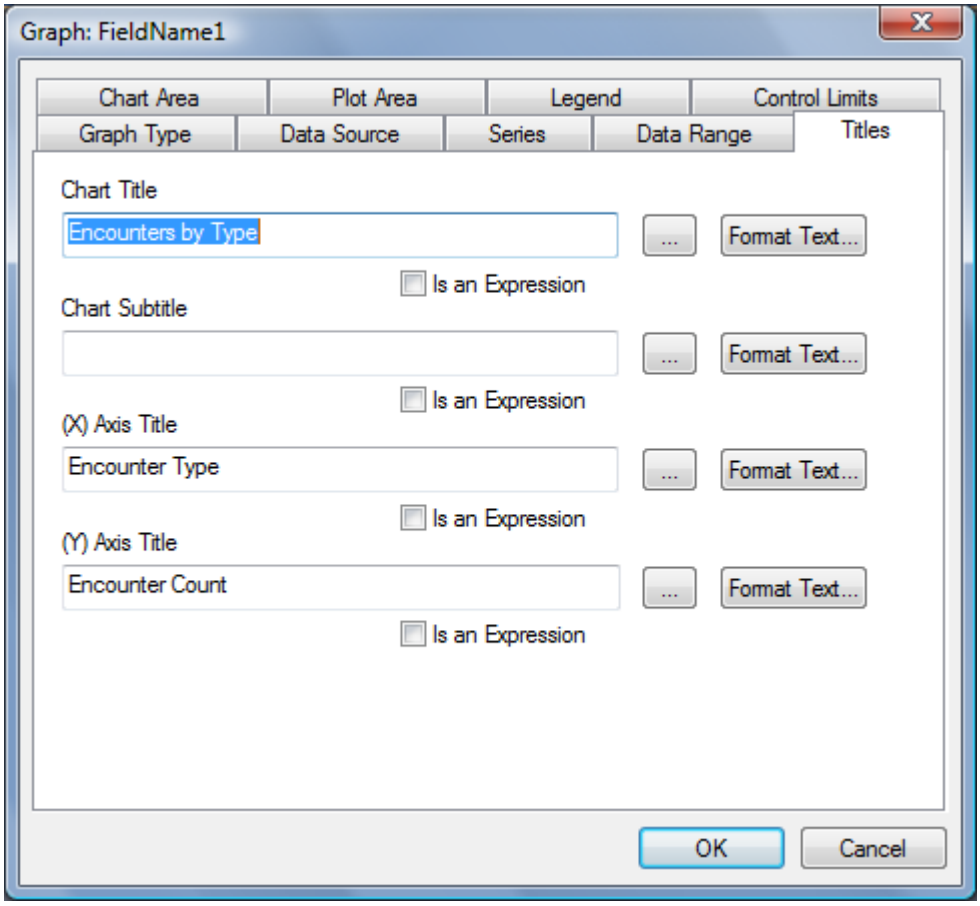
70. At the prompt, use MINE for the output device and enter starting and ending dates that will return records that were registered over several different days.

Your output is displayed similar to the following example. Your numbers will be different based on the data that exists in your environment.



Adding titles to the graph can make it more meaningful and easier to read.

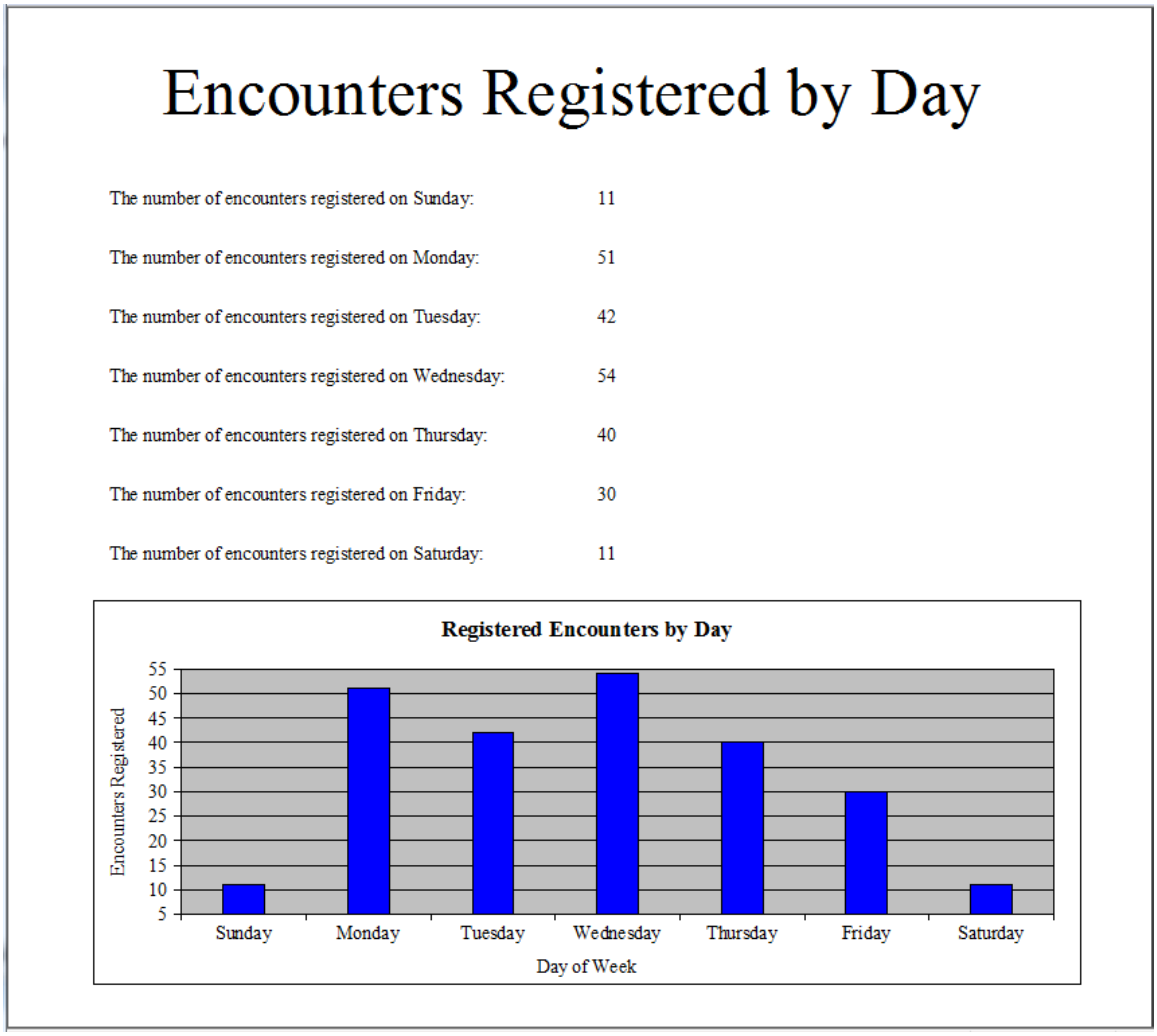
71. Close the output window to return to your layout.
72. Double-click the graph item to open the Graph Properties dialog box.
73. On the Titles tab:
    - Enter Registered Encounters by Day as the Chart Title.
    - Enter Day of Week as the (X) Axis Title.
    - Enter Encounters Registered as the (Y) Axis Title.



74. Click OK to close the Graph Properties dialog box.

75. From the Build menu, select Run "1_*Your_Initials*_Example_Graph" or press CTRL+F5 to execute your layout program.

76. At the prompt, use MINE for the output device and enter starting and ending dates that will return records that were registered over several different days.



77. Close the output window to return to your layout.

So far the Encounters Registered by Day graph has been a single series graph. Suppose we want to subdivide the encounters registered on a day by the encounter type. We can accomplish this by creating separate counter variables for the different encounter types and then use those variables as a series in the graph.

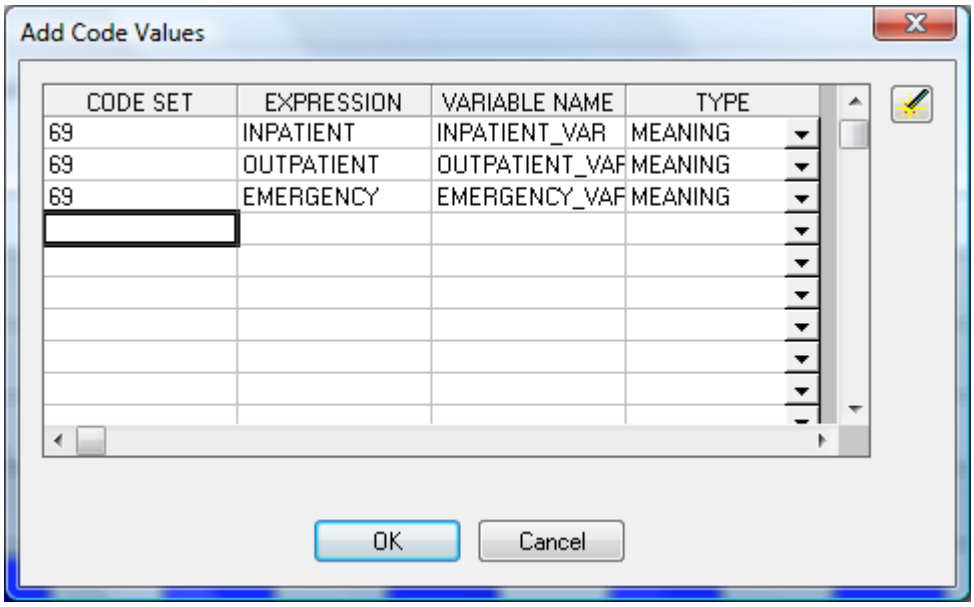
To keep this example fairly simple we will limit the query to only return inpatient, outpatient, and emergency encounter types. If your environment does not have rows on the ENCOUNTER table with these encounter types, select different encounter types that exist in your environment.

78. From the Tools menu, select Add Code Values. The Add Code Values dialog box opens.

79. Enter the following information to create global variables that can be set equal to the Code_Values for inpatient, outpatient, and emergency encounters.

```
69 INPATIENT  INPATIENT_VAR  MEANING
69 OUTPATIENT OUTPATIENT_VAR MEANING
69 EMERGENCY  EMERGENCY_VAR  MEANING
```

Your Add Code Values dialog box should display similar to the following example:



80. Click OK to close the Add Code Values dialog box. Depending on the data in your environment, you may need to select some different encounter types.

81. From the Tools menu, select Query Builder. The Add Queries dialog box opens.

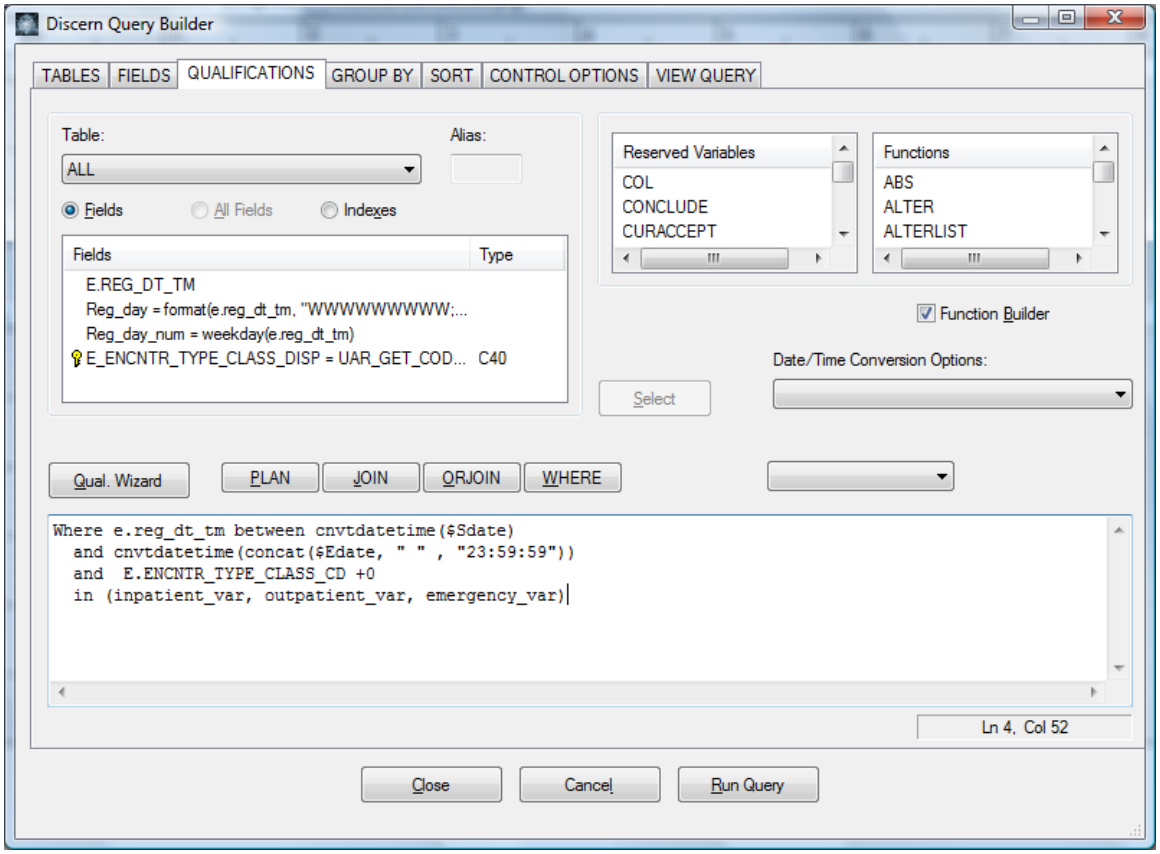
82. Verify the Get_Encounters_by_Day query is selected and click Edit Query. The Query Builder dialog box opens.

83. On the Fields tab select the ENCNTR_TYPE_CLASS_CD field.

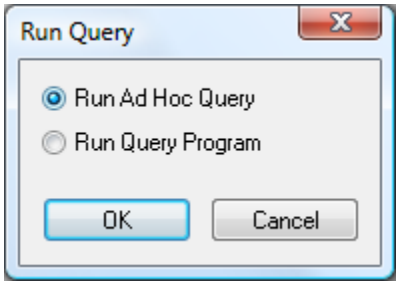
84. On the Qualifications tab, add the following code:

```
and  E.ENCNTR_TYPE_CLASS_CD +0
in (inpatient_var, outpatient_var, emergency_var)
```

Your qualifications look similar to the following example:



85. Click the Run Query button. The Run Query dialog box shown below will open.



The Run Query dialog box allows you to either execute just the current query using the Run Ad Hoc Query option, or to execute the commands from the beginning of the program through the current query using the Run Query Program option. Since your current query uses values from the prompts in the qualifications, it cannot be executed as a stand alone query. To execute, the query needs the beginning and ending dates from the prompts. Choosing the Run Query Program option will display the prompt form and then execute the program through the current query.

86. Select the Run Query Program option and click OK.

87. At the prompt, use MINE for the output device and enter starting and ending dates that will return records that were registered over several different days.

Verify your output includes inpatient, outpatient, and emergency encounters.

88. Close the output window.

89. Close Query Builder.

90. Click OK to close the Add Queries dialog box.

91. From the Select/Modify Section of the Workspace dialog box, select the Foot Reg_day Reportwriter section (Click the label, not the check box).

92. Double-click the Code Segment. The Layout Source [Code Segment] dialog box opens.

Currently, you have the following code in the Layout Source [Foot Reg_day] dialog box:

```
Dcnt = count(e.seq)
```

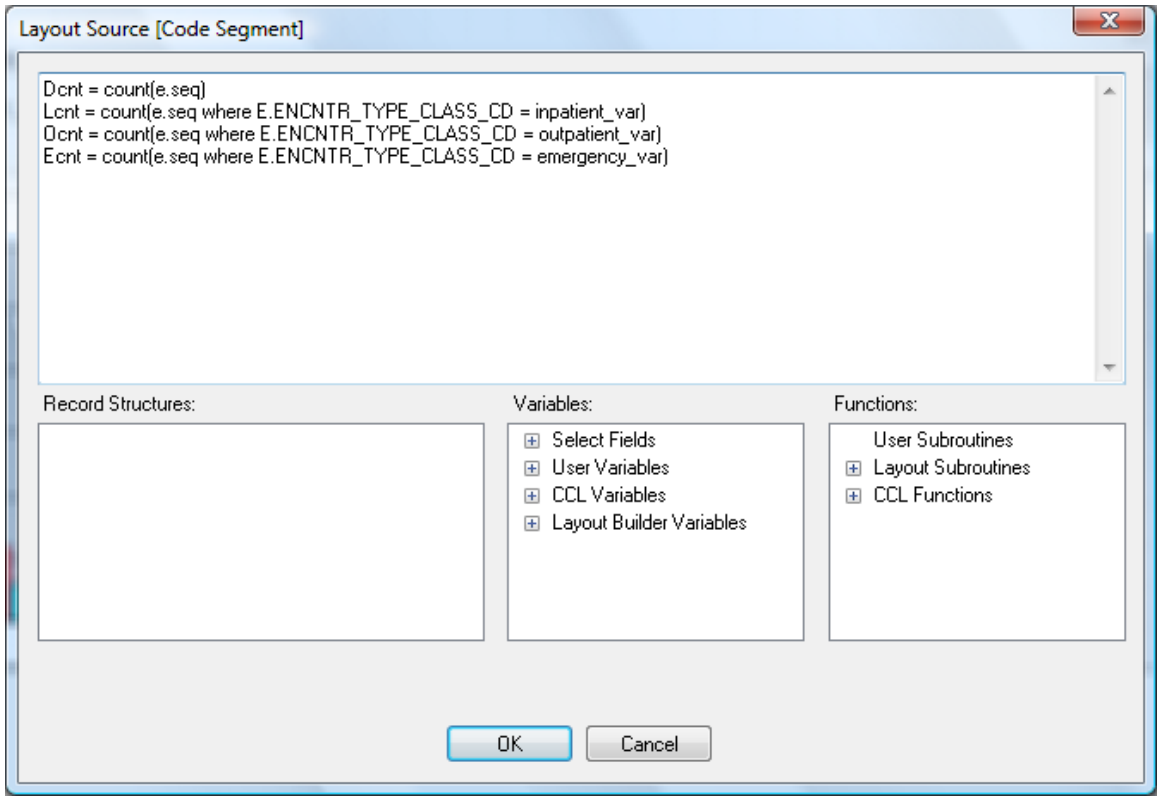
Using the code above, Dcnt is set equal to the total number of encounters registered on each day of the week.

93. Add the following code directly below the Dcnt = count(e.seq) code:

```
Icnt = count(e.seq where E.ENCNTR_TYPE_CLASS_CD = inpatient_var)
Ocnt = count(e.seq where E.ENCNTR_TYPE_CLASS_CD = outpatient_var)
Ecnt = count(e.seq where E.ENCNTR_TYPE_CLASS_CD = emergency_var)
```

In the above code, only encounters with the specific encntr_type_class_cd are counted when setting the Icnt, Ocnt, and Ecnt variables.

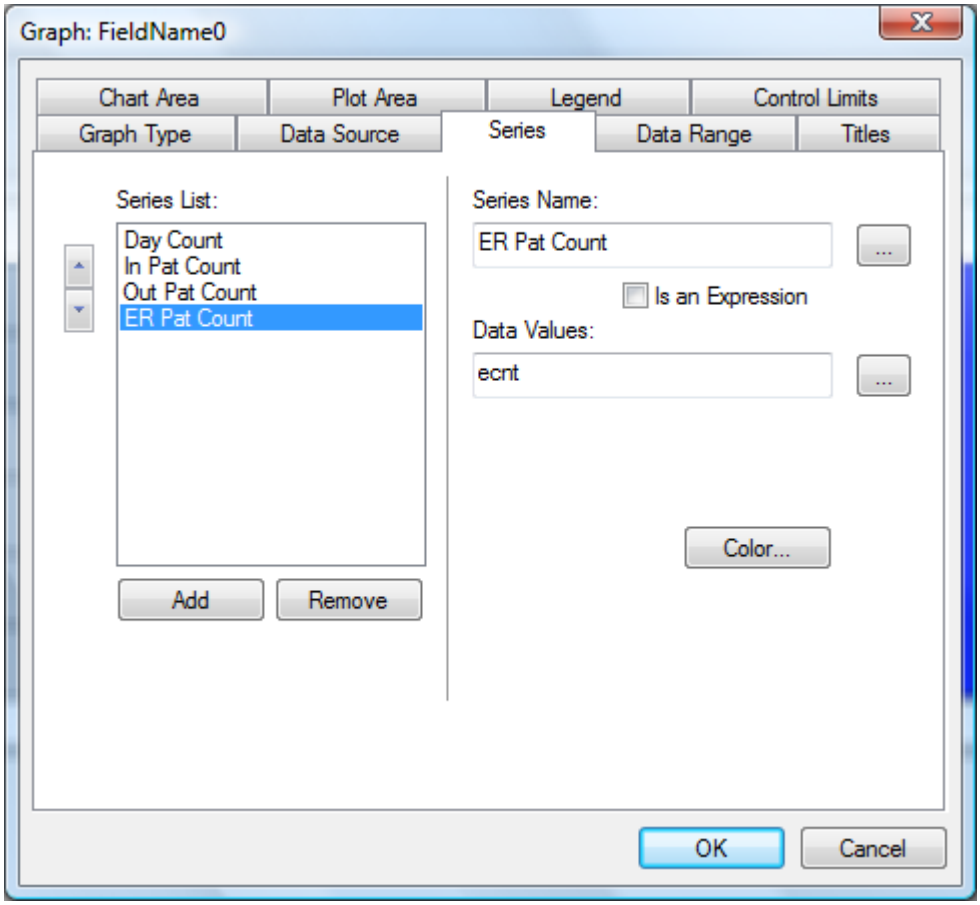
Your Layout Source [Code Segment] dialog box should look similar to the following example:



Doc ID: A1001001A08G07B34433A94851

94. Click OK to close the Layout Source [Code Segment] dialog box.
95. Double-click your graph item to open the Graph dialog box.
96. On the Series tab, click Add to add a series for Icnt.
97. In the Series Name box, enter **In Pat Count**; in the Data Values box enter **Icnt**.
98. Click Add to add a series for Ocnt.
99. In the Series Name box, enter **Out Pat Count**; in the Data Values box, enter **Ocnt**.
100. Click Add to add a series for Ecnt.
101. In the Series Name box, enter **ER Pat Count** and in the Data Values box, enter **Ecnt**.

You Graph dialog box should look similar to the following example:

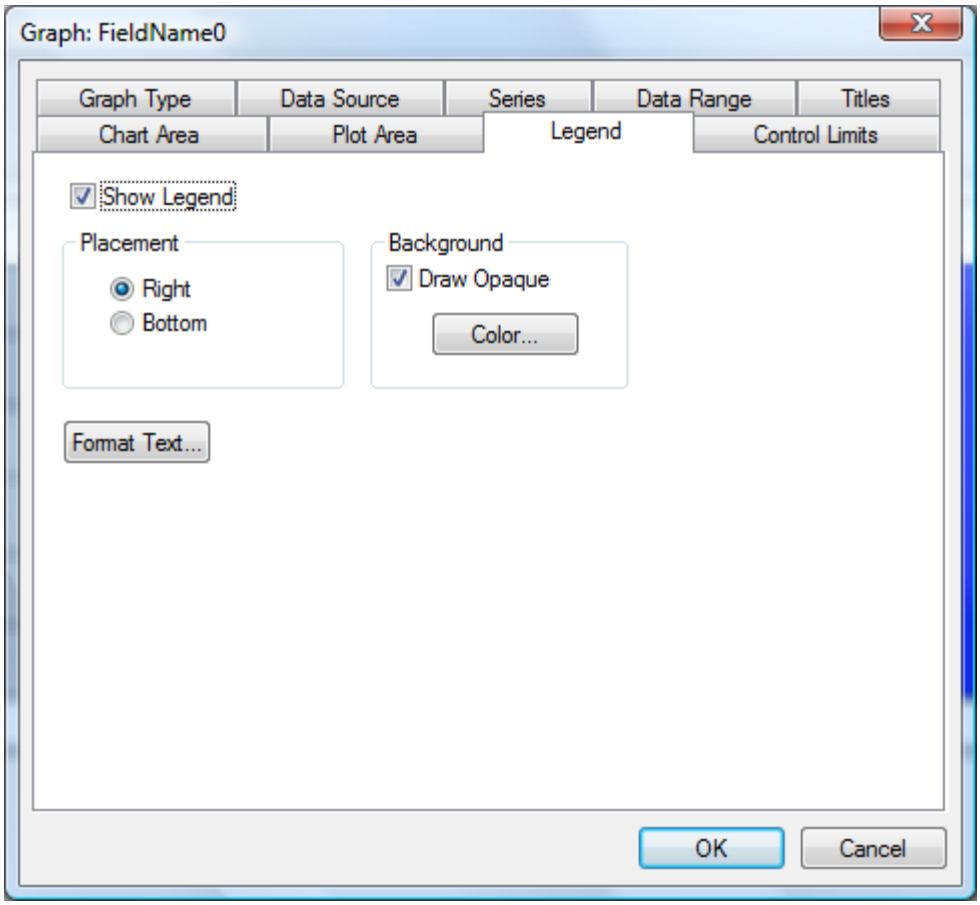


Adding Icnt, Ocnt, and Ecnt to the Series tab will display a column on the graph for each of those variables. Because Reg_day is used as the Axis Field on the Data Source tab, the graph will display four columns for each day of the week. The first column displays the total number of encounters, the second column will show the number of inpatient encounters, the third column displays the outpatient encounters, and the fourth column

displays the emergency encounters. You can use the Legend tab to display a legend that indicates what each of the columns represents.

102. Click the Legend tab and select the Show Legend option.

Your Graph dialog box should display similar to the following example:



103. Click the Data Range tab on the Graph dialog box.

104. Delete the Minimum, Maximum, and Index values. This will ensure that all values are displayed on the graph.

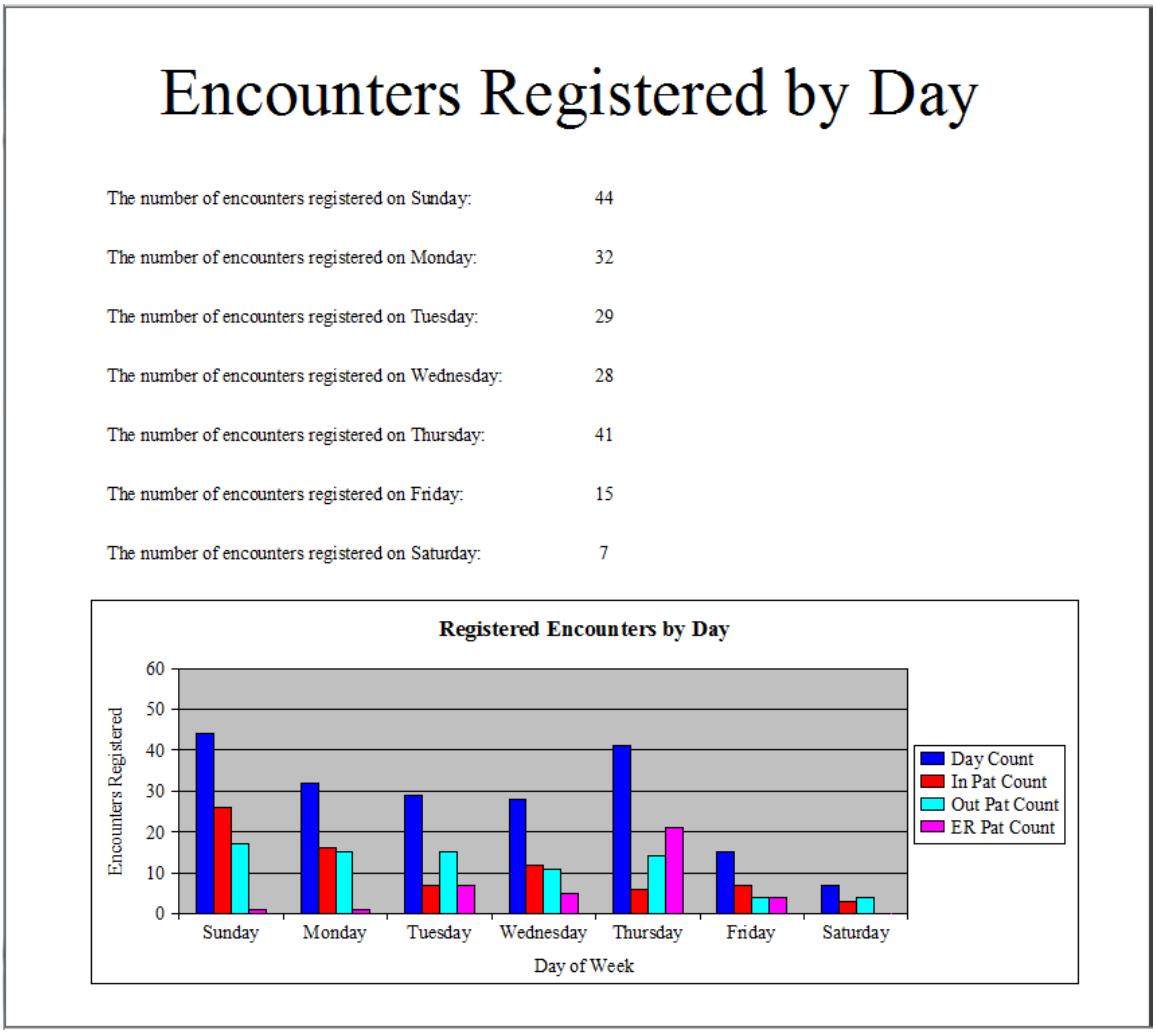
105. Click OK to close the Graph dialog box.

106. From the Build menu, select Run "1_*Your_Initials*_Example_Graph" or press CTRL+F5 to execute your layout program.

107. Click Yes when prompted to save the layout. The prompt form opens.

108. At the prompt, use MINE for the output device and enter starting and ending dates that will return records that were registered over several different days.

Your output is displayed similar the following example. Your numbers will be different based on the data that exists in your environment.



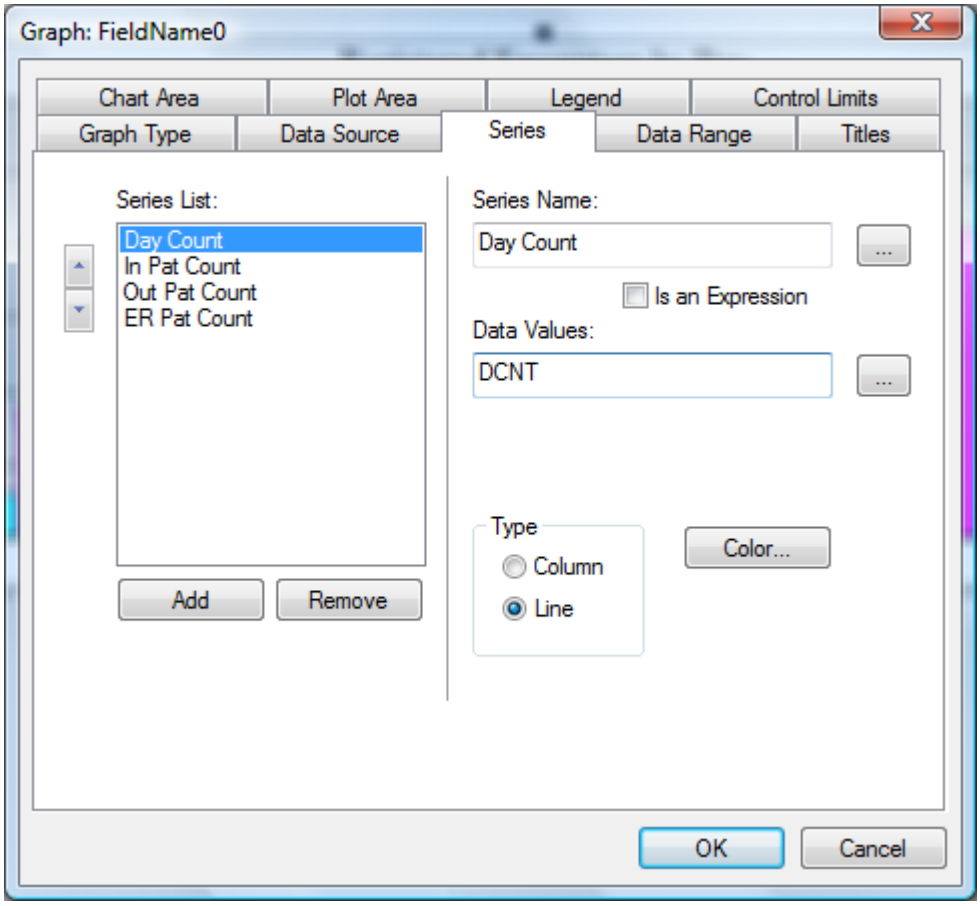
In the above report, Day Count represents the total number of encounters registered on each day of the week. The sum of the other three counts equals Day Count. Rather than expressing Day Count as a column, the graph could be more meaningful if Day Count was expressed as a line to represent the total number of encounters and the other three counts continued to be displayed as columns. This is done by changing the graph type to Line – Column.

109. Close the output window to return to the layout.

110. Double-click your graph item. The Graph dialog box opens.

111. On the Graph Type tab select Line - Column.

112. On the Series tab select the Day Count series and modify the Type to Line.

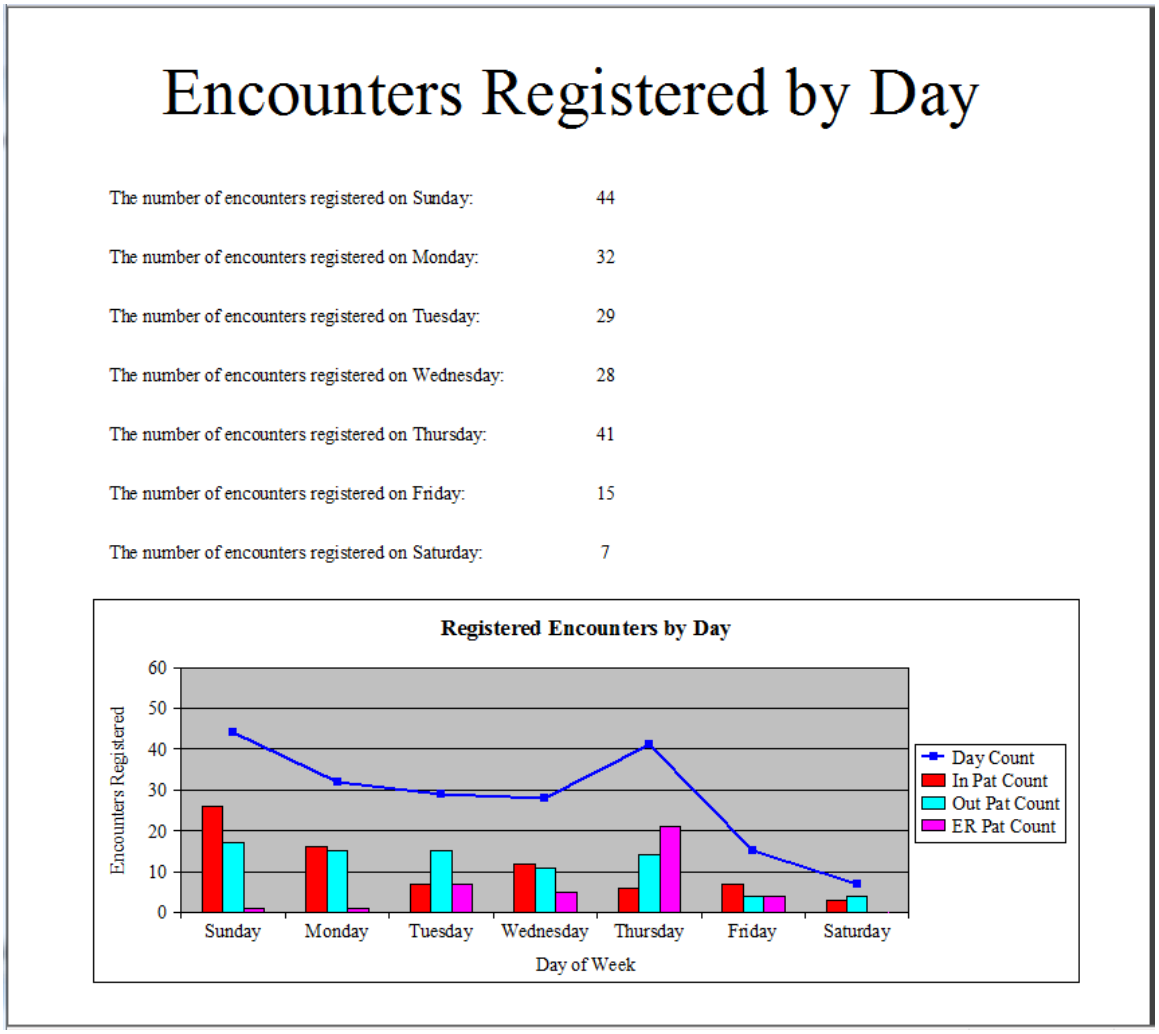


113. Click OK to close the Graph Properties dialog box.

114. From the Build menu, select Run "1_*Your_Initials*_Example_Graph" or press CTRL+F5 to execute your layout program. Click Yes when prompted to save the layout. The prompt form opens.

115. At the prompt, use MINE for the output device and enter starting and ending dates that will return records that were registered over several different days.

Your output is displayed similar to the following example. Your numbers will be different based on the data that exists in your environment.



Graphing Data From Multiple Queries

There may be times when you want to graph data from several sources. The example above used *Discern Explorer* Reportwriter sections to create count variables and display them in graphical format. Aggregates created in the selection list of a query can also be displayed in graphical format. Layout Builder enables you to create layout sections that are not associated with a reportwriter section of a specific query. It also enables you to create multiple queries in a single layout program. The Graph Tool enables you to select a query as the data source for the graph. Combining this functionality enables you to create multiple graphs from different queries and display them in a single layout program. To demonstrate this functionality, we will create a report that contains a graph displaying the number of encounters with specific encounter types over a time period that will be specified by the user with a prompt. The report also will contain a graph showing the counts of specific orders over the same time period.

1. Using DVDev, from the File menu, select New. The New dialog box opens.

2. From the File Type list, select Layout Program.

3. In the Program Name box, enter **1_*Your_Initials*_Graph_Ords_Encs** and click OK. The New Layout Program dialog box opens.

4. Verify the Standard Layout Report Layout and PostScript Output Type options are selected and click Next. The Paper Size dialog box is displayed.

5. For our report, keep the defaulted values and click Finish. When a new layout program is created, DVDev places an empty layout section on the layout entitled DetailSection.

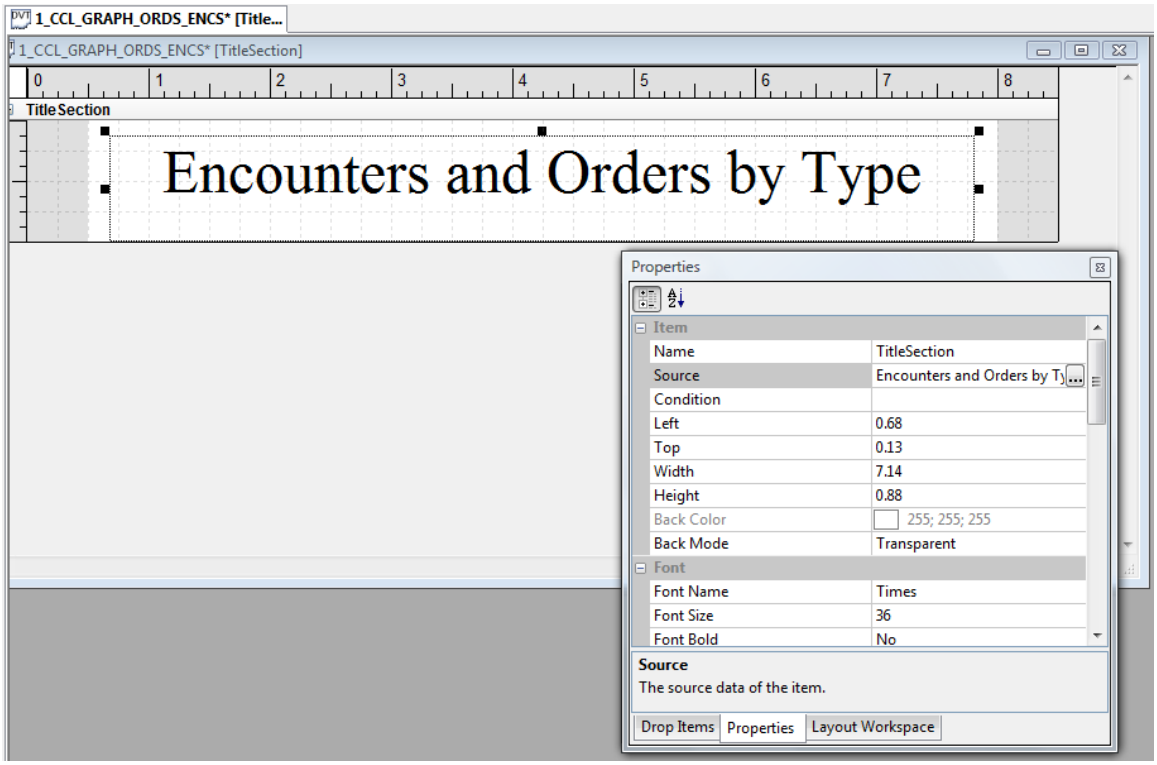
6. In the Properties dialog box, modify the name to **TitleSection** (if the Properties dialog box is not displayed, open it using the Properties option on the View menu).

7. Use the Label Tool  to add the report title to the TitleSection.

8. Use the Formatting toolbar shown below to set:

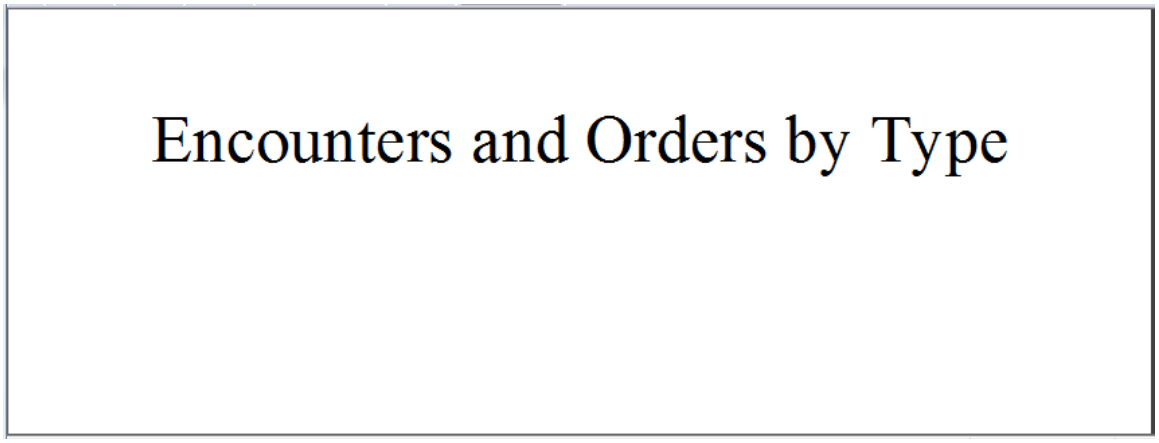
   - The font to Times
   - The font size to 36
   - Center the text

If the Formatting toolbar is not displayed, from the View menu, select Toolbars > Formatting. You can expect your layout to be similar to the following screen:



9. From the Build menu, select Run "1_*Your_Initials*_Graph_Ords_Encs" or press CTRL+F5 to execute your layout program.

10. Use MINE at the prompt for the output device and click Execute. Your output should be similar to the following:



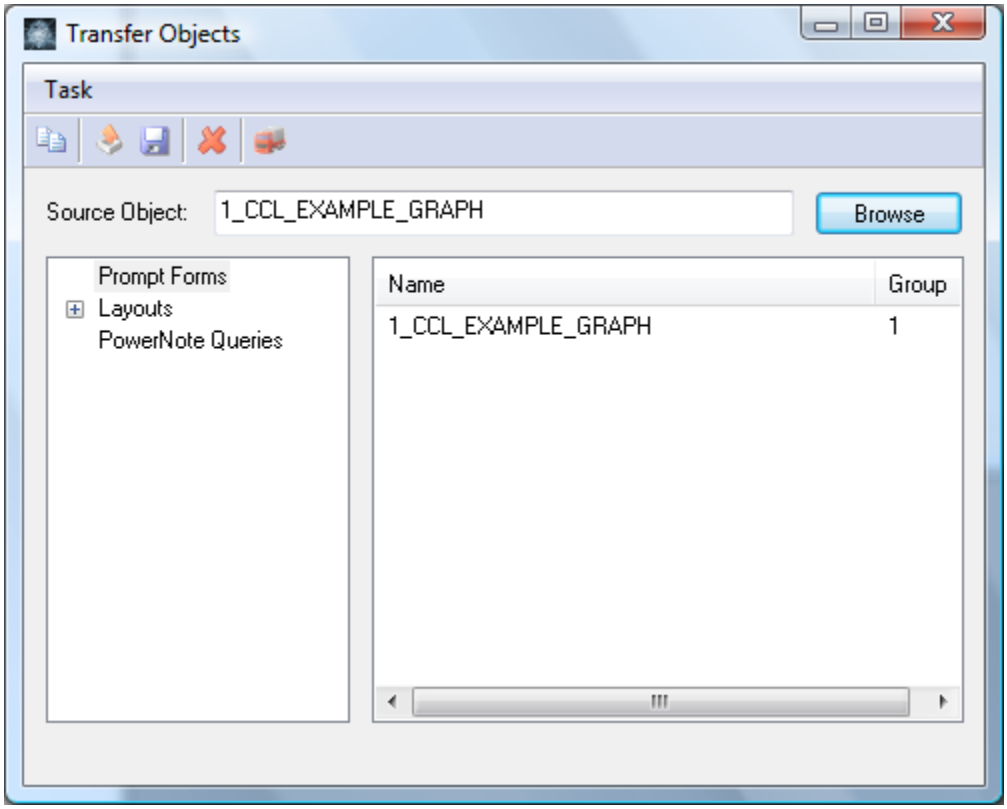
At this point it is important to understand that when a layout is associated with a query, rendering a layout section requires the layout section to be added to at least one of the Reportwriter sections of the associated query. If the layout is not associated with a query, each layout section is rendered in the order they are displayed on the layout. Because there is no query associated with the layout in this example, the TitleSection layout section is rendered when the program is executed.

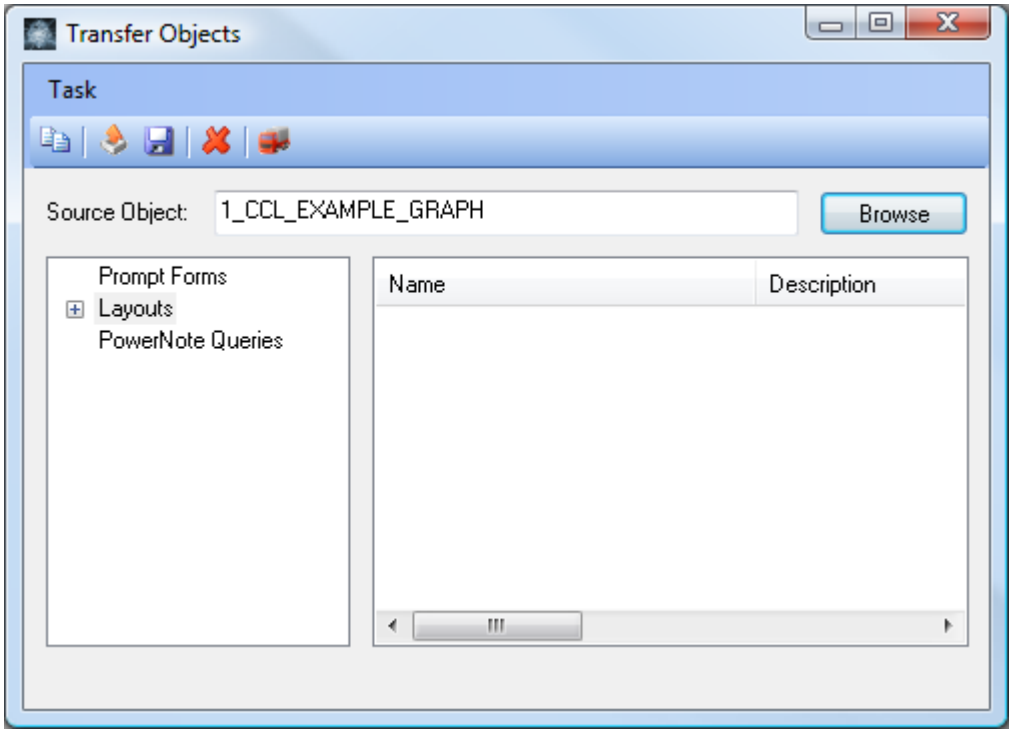
For this report we want to use prompts for a starting and ending date that are very similar to the prompts we used in the previous example where you created the graph using data from a single query. We could recreate these prompts using the Prompt Builder, however, it is easier to copy the prompts from the earlier example program to this example program. If you did not complete the earlier example, see the Graphing Data from a Single Query section and follow the steps for creating the prompts.

11. Close the Report Output window.

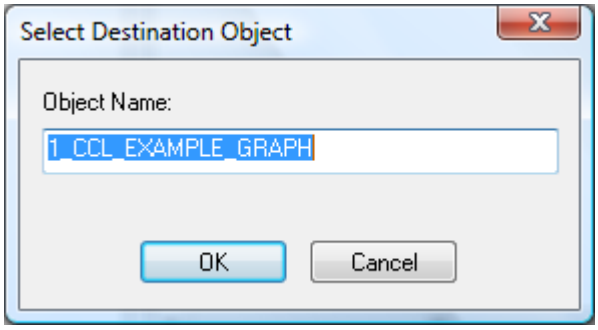
12. From the Tools menu, select Transfer Objects. The Transfer Objects dialog box opens.

13. In the Source Object box, enter **1_*Your_Initials*_Example_Graph** and click Browse. Your prompt form opens in the search result box.

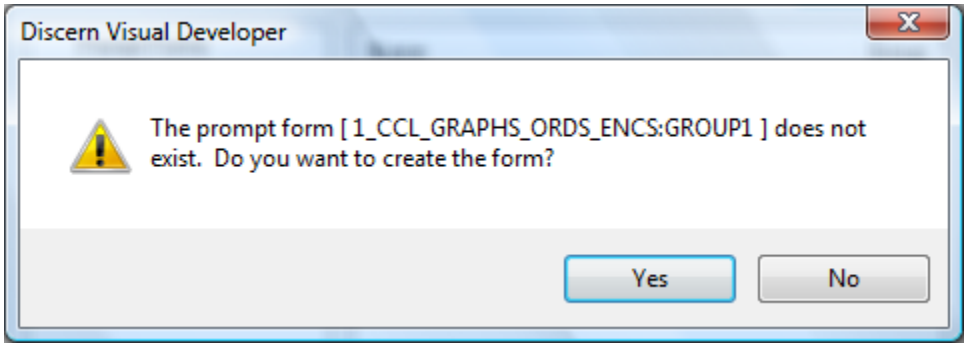




14. Right-click your prompt form and select Copy Object, or select Copy Object from the Task menu. The Select Destination Object dialog box opens.

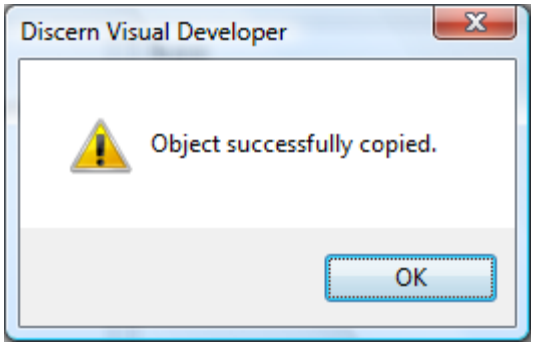


15. In the Object Name box enter **1_*Your_Initials*_Graph_Ords_Encs** and click OK. A warning message similar to the following example is displayed.



When you executed the layout program earlier in this exercise, a default prompt form with an output device control was used for your program.

16. Click Yes to create a prompt form for the 1_*Your_Initials*_Graph_Ords_Encs program that is a copy of the prompt form from the 1_*Your_Initials*_Example_Graph program. A message similar to the following example is displayed:



17. Click OK and close the message dialog box.

18. Close the Transfer Objects dialog.

19. From the Build menu, select Run "1_*Your_Initials*_Graph_Ords_Encs" or press CTRL+F5 to execute your layout program. The new prompt form is displayed with a control for an output device, a control for the starting date, and a control for the ending dates.

20. Click Cancel to close the prompt form.

On the prompt form you just copied there is a control named Sdate. It has a Date Time control with a Prompt Type of String and a Command Line Format of "dd-mmm-yyyy". There also is a control named Edate. It has a Date Time control with a Prompt Type of String and a Command Line Format of "dd-mmm-yyyy". A common request is to have the values entered at the prompts displayed in the output.

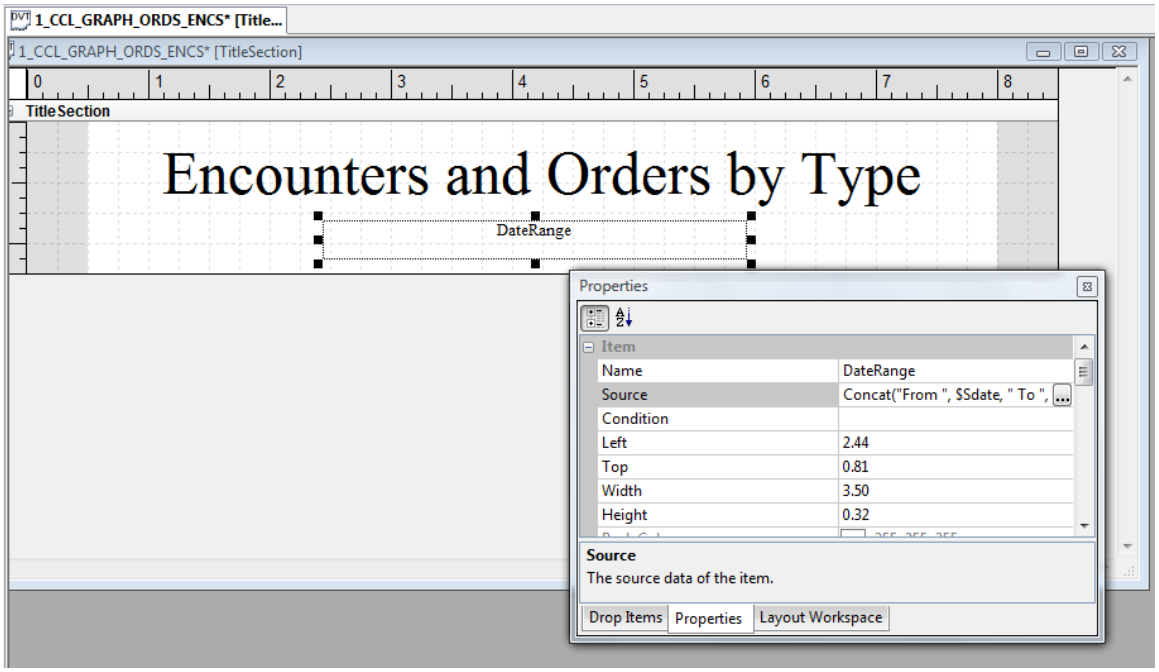
21. On the Properties for the TitleSection change the Height to 1.25.

22. Use the Text Tool  to place a text field in the TitleSection layout section.

23. In the Properties dialog box, enter **DateRange** as the Name.

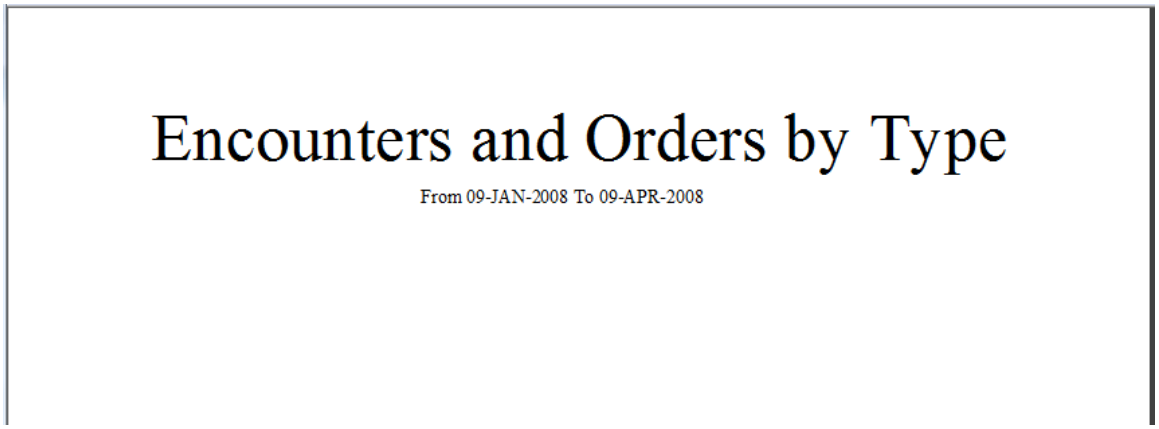
24. Enter **Concat("From ", $Sdate, " To ", $Edate)** as the source.

25. Center the text. You can expect your layout to be similar to the following screen:



26. Select Run "1_*Your_Initials*_Graph_Ords_Encs" from the Build menu or press CTRL+F5 to execute your layout program. Click Yes when prompted to save the layout. The prompt form opens.

27. At the prompt, use MINE for the output device and enter starting and ending dates. Your output should be similar to the following example:



For this example we want to create a graph to show the counts of orders with specific catalog codes and a graph to show the counts of encounters with specific encounter type class codes. To qualify on the specific orders and encounters, you need to create and set variables equal to the code values of the specific catalog and encounter type class codes.

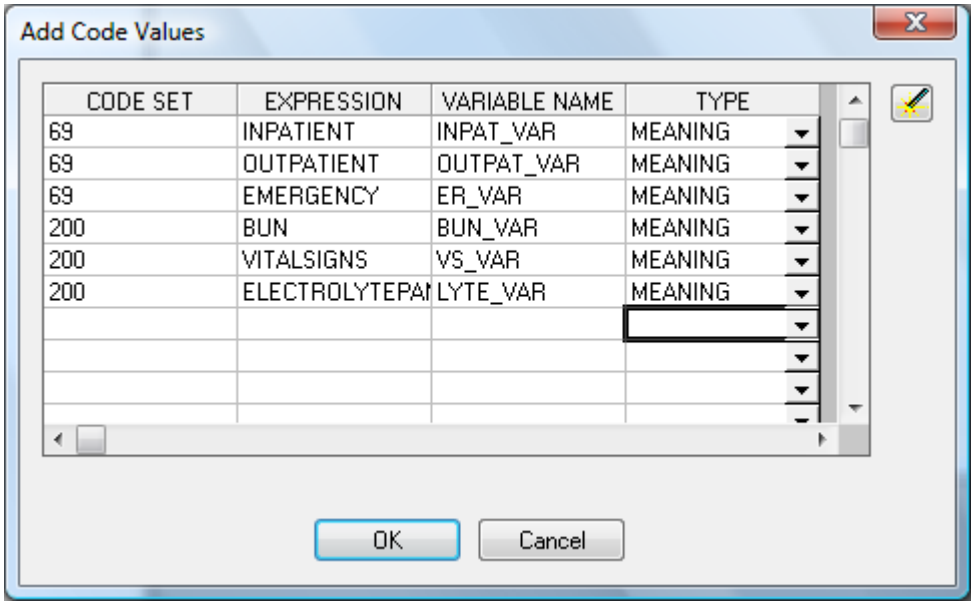
28. Close the Report Output window.

29. From the Tools menu, select Add Code Values. The Add Code Values dialog box opens.

30. Enter the following codes to create global variables that can be set equal to the Code_Values for inpatient, outpatient, and emergency encounters. By default, the Add Code Values dialog box appends _VAR to the value you enter as the Expression to create the Variable Name. You will need to modify the name of the variable by double-clicking the Variable Name column.

```
69  INPATIENT          INPAT_VAR  MEANING
69  OUTPATIENT         OUTPAT_VAR MEANING
69  EMERGENCY          ER_VAR     MEANING
200 BUN                BUN_VAR    DISPLAYKEY
200 VITALSIGNS         VS_VAR     DISPLAYKEY
200 ELECTROLYTEPANEL   LYTE_VAR   DISPLAYKEY
```

For Code Set 200, the MEANING may not be populated on the code_value table. You will need to modify the TYPE to use DISPLAYKEY or another type that reflects what you placed in the EXPRESSION column. Your Add Code Values dialog box should look similar to the following example:



NOTE: You may need to execute a couple of ad hoc queries to determine encounter type class codes and order catalog codes to use and modify your program appropriately.

31. Click OK to close the Add Code Values dialog box.

Next, we want to add two queries, one to count the number of each type of encounter and one to count number of each type of orders.

32. From the Tools menu, select Query Builder. The Add Queries dialog box opens.

33. Click Add. The Add Query dialog box opens.

34. In the Query Name box, enter **Get_Orders**, verify that the Associate Layout option is deselected, and click OK. The Discern Query Builder dialog box opens.

35. From the Tables list select ORDERS.

36. Click the Fields tab and click Code Values button. The Code Value Displays dialog box opens.

37. Select CODE_VALUE field and DISPLAY field, and then click OK.

38. From the Fields list, select CATALOG_CD.

39. Add the expression, **OCOUNT = COUNT(O.Catalog_CD)**.

40. Click the Qualifications tab and enter the following code:

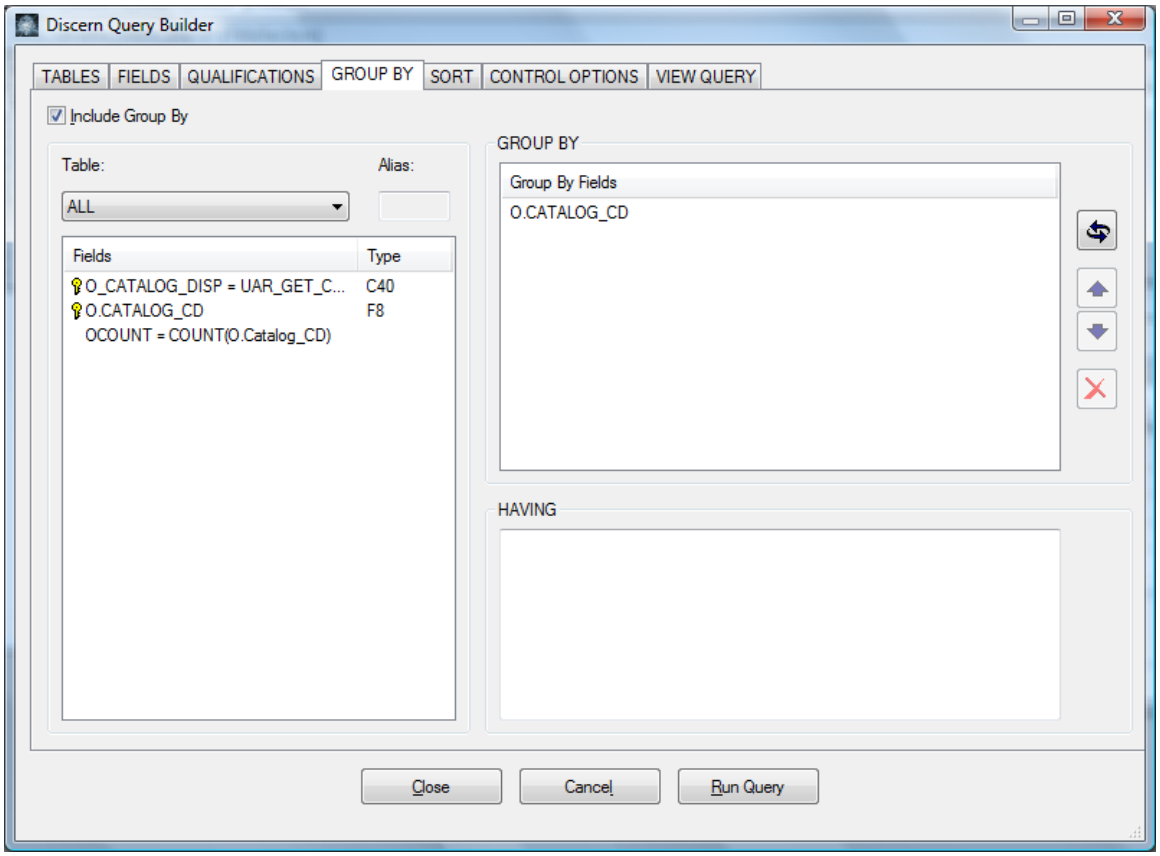
```
WHERE  O.CATALOG_CD +0 IN (LYTE_VAR, VS_VAR, BUN_VAR)
  AND  O.ORIG_ORDER_DT_TM BETWEEN
       CNVTDATETIME($SDATE) AND
       CNVTDATETIME(concat($Edate, " " , "23:59:59"))
```


Doc ID: A1001001A08G07B34433A94851

In the above qualification, we used the variables you created from Code Set 200 in the IN clause. Also in the qualification, the + 0 used after the O.Catalog_CD, O.Catalog_CD +0 forces the query to use the index on O.Orig_Order_DT_TM.

41. Click the Group By tab, select the Include Group By option, and verify O.Catalog_Cd is displayed in the Group By Fields list.

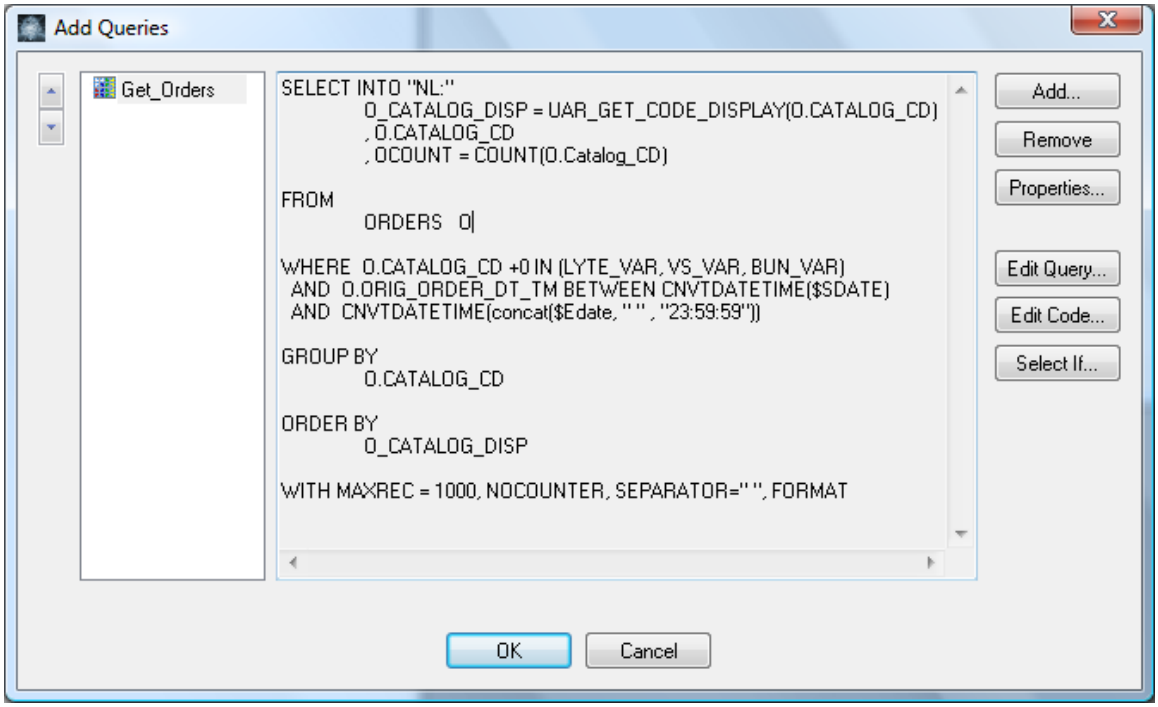
The Query Builder dialog box should look similar to the following example:



42. Click the Sort tab and select O_CATALOG_DISP from the Fields list.

43. Click the Control Options tab, set the Max Records to **1000** and select the Into "NL:" option.

44. Click Close to exit Discern Query Builder and return to the Add Queries dialog box. The code for your Get_Orders query that is displayed on the Add Queries dialog box should look similar to the following example:



45. Click Add to start the next query that will get encounter information. The Add Query dialog box opens.

46. In the Query Name box, enter **Get_Encounters**, verify the Associate Layout option is deselected and click OK. The Discern Query Builder dialog box opens.

47. From the Tables list, select ENCOUNTER.

48. Click the Fields tab and click the Code Values button. The Code Value Displays dialog box opens.

49. Select the CODE_VALUE field, the DISPLAY field and click OK.

50. Select the Encntr_Type_Class_CD field.

51. Add the expression, ECOUNT = COUNT(E.Encntr_Type_Class_CD)

52. Click the Qualifications tab and enter the following code:

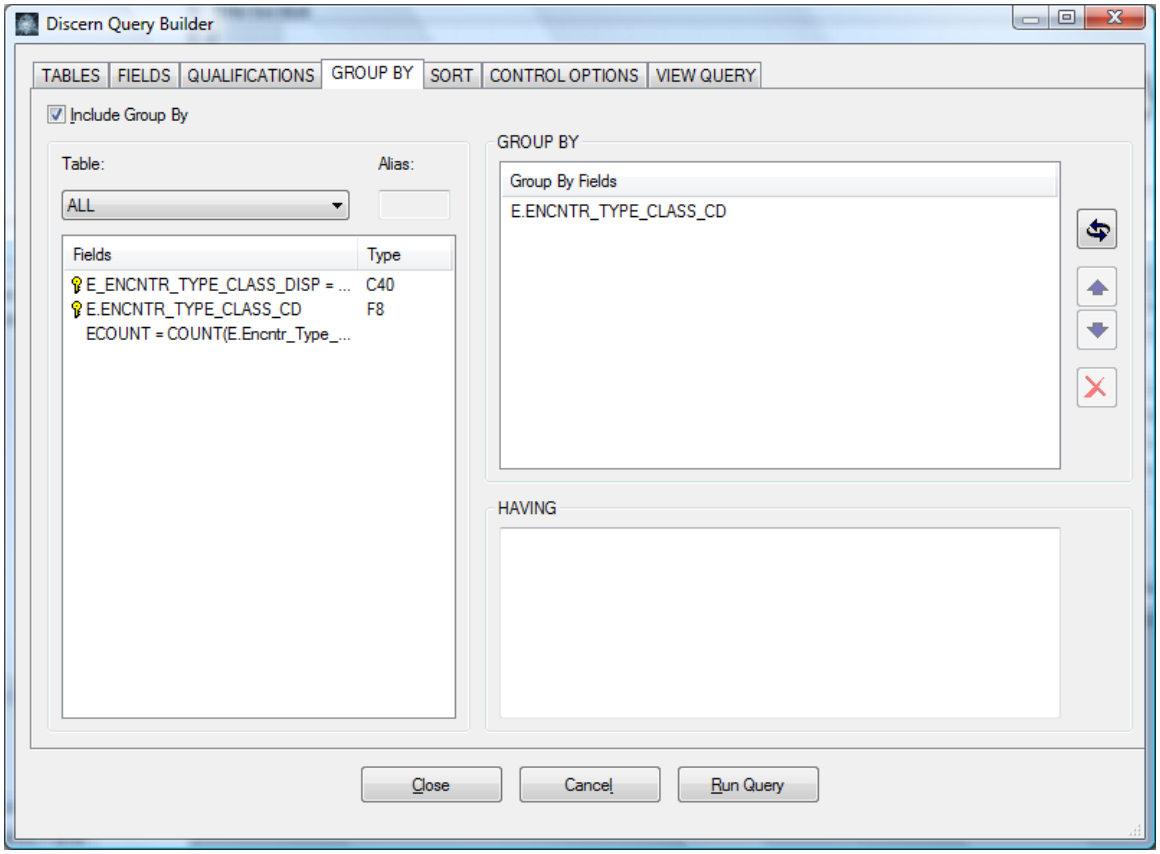
```
WHERE  E.ENCNTR_TYPE_CLASS_CD +0 IN (INPAT_VAR, OUTPAT_VAR, ER_VAR)
       AND E.REG_DT_TM BETWEEN CNVTDATETIME($SDATE)
       AND CNVTDATETIME(concat($Edate, " " , "23:59:59"))
```

In the example above, we used the variables you created from Code Set 69 in the IN clause. In this example, E.Encntr_Type_Class_CD +0 forces the query to use the index on E.Reg_DT_TM. Also notice the prompt symbols $Sdate and $Edate are used in both queries for this program.

53. Click the Group By tab and select the Include Group By option and verify E.Encntr_Type_Class_CD is displayed in the Group by Fields list.

The Query Builder dialog box should look similar to the following example:

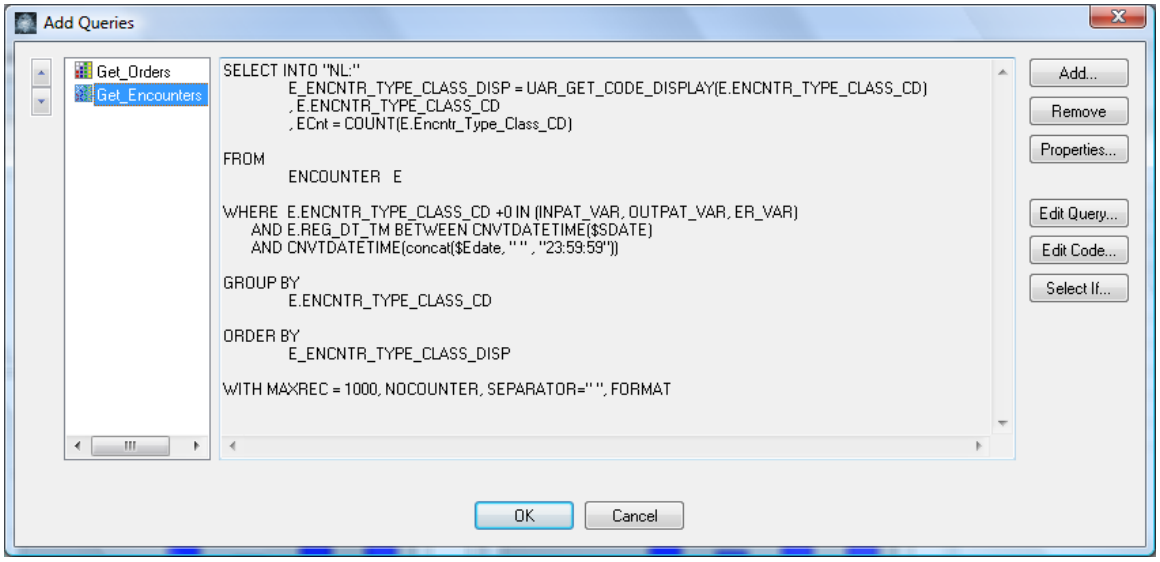
Doc ID: A1001001A08G07B34433A94851



54. Click the Sort tab, and select E_ENCTR_TYPE_CLASS_DISP from the Fields list.

55. Click the Control Options tab, set the Max Records to 1000, select the Into "NL:" option and click Close to exit Discern Query Builder and return to the Add Queries dialog box.

The code for your Get_Encounters query that is displayed on the Add Queries dialog box should look similar to the following example:



56. Click OK to close the Add Queries Dialog box.

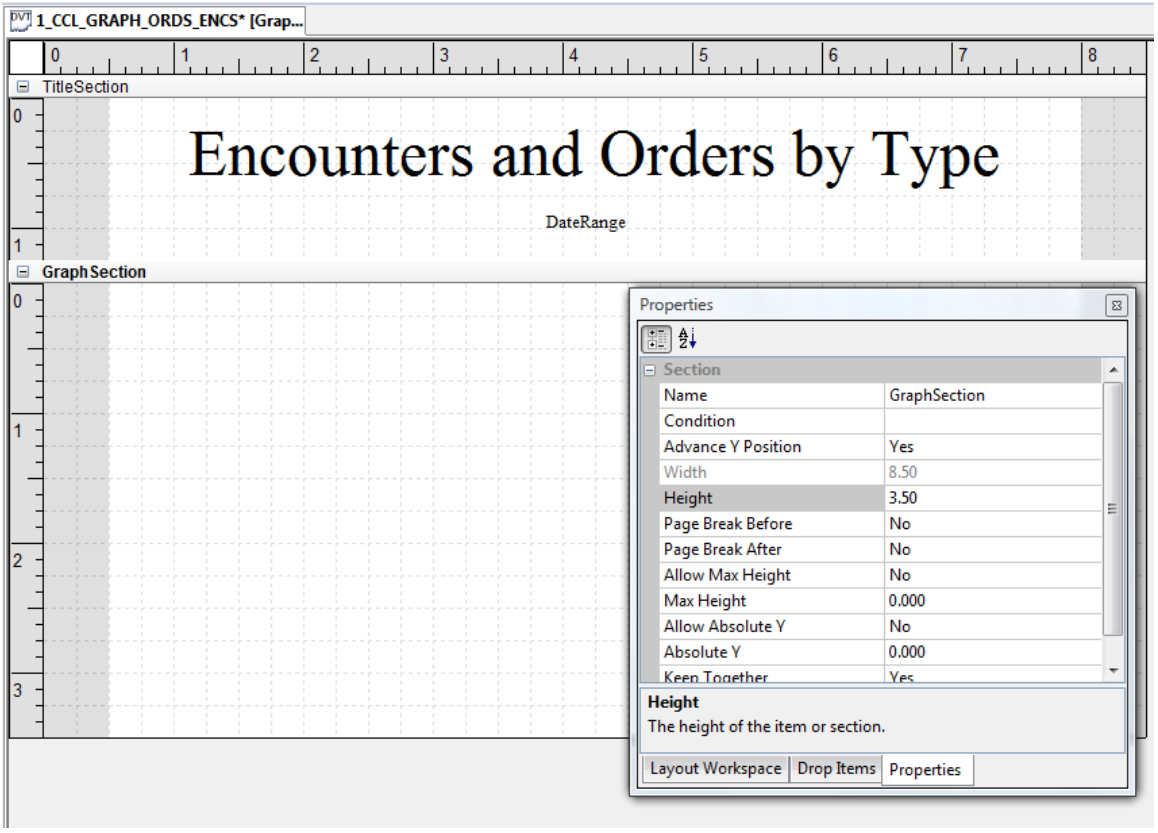
We are now ready to add graphs to the layout to display the output of the queries in graphical form.

57. From the Edit menu, select Insert > Section to insert a new layout section.

58. Modify the name of the new section to **GraphSection**.

59. Drag the GraphSection down below the TitleSection.

60. Modify the Height of the GraphSection to **3.5**. You can expect your layout to be similar to the following screen:



61. Use the Graph Tool to create a graph item over most of the left side of the GraphSection layout section. The Graph dialog box opens.

62. Click the Data Source tab, and from the Select Query list, select Get_Orders.

63. From the Axis Field list select O_Catalog_Disp, and in the Axis Label box enter **O_Catalog_Disp**, or click the ellipsis button to open the Layout Source dialog box and select O_Catalog_Disp from the Variables list.

64. Click the Series tab, and in the Series Name box, modify the name to **Order Count**.

65. In the Data Values box enter **Ocount**, or click the ellipsis button to open the Layout Source dialog box and from the Select Fields list, select Ocount.

66. Click the Titles tab. In the Chart Title box, enter **Orders by Type**, in the (X) Axis Title box, enter **Order Type**, and in the (Y) Axis Title box, enter **Order Count**. Click OK to close the Graph dialog box.

67. Use the Graph Tool to create a graph item over most of the right side of the GraphSection layout section. The Graph dialog box opens.

68. Click the Data Source tab, and from the Select Query list, select Get_Encounters.

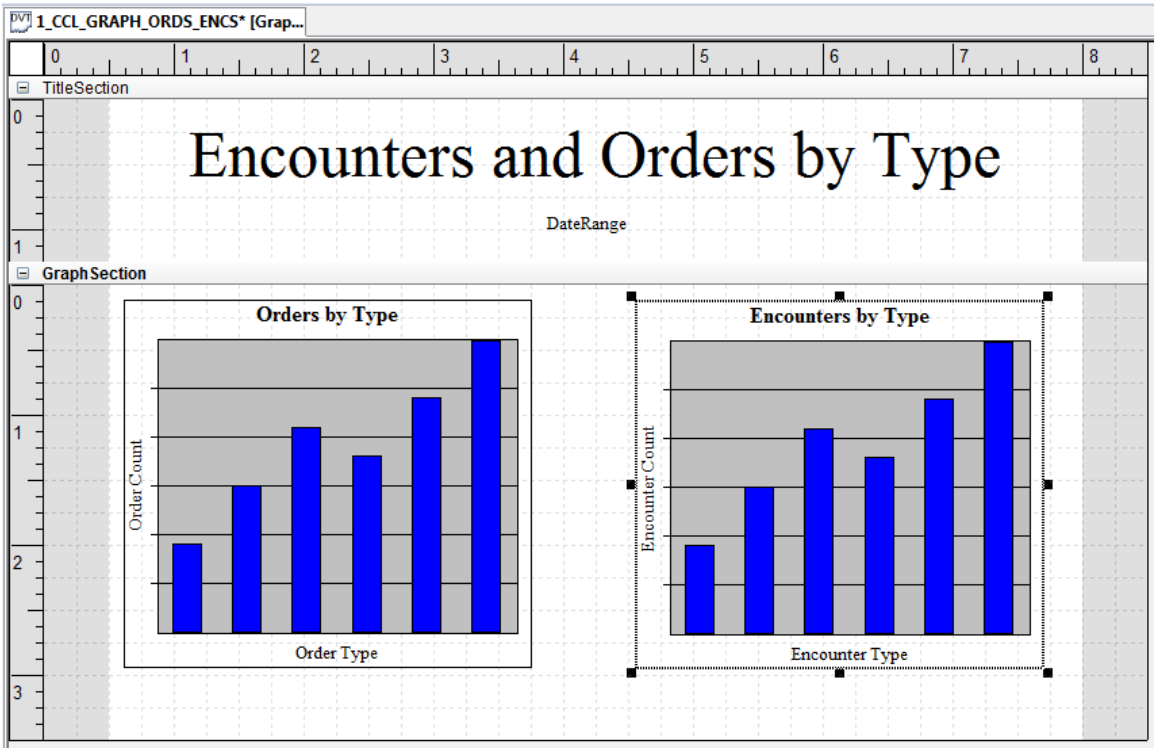
69. From the Axis Field list select E_ENCTR_TYPE_CLASS_DISP. In the Axis Label box, enter **E_ENCTR_TYPE_CLASS_DISP**, or click the ellipsis button to open the Layout Source dialog box and from the Variables list, select E_ENCTR_TYPE_CLASS_DISP.

70. Click the Series tab and in the Series Name box, modify the name to **Encounter Count**.

71. In the Data Values box, enter **Ecount**, or click the ellipsis button to open the Layout Source dialog box and from the Select Fields list, select Ecount.

72. Click the Titles tab. In the Chart Title box, enter **Encounters by Type**, in the (X) Axis Title box, enter **Encounter Type**, and in the (Y) Axis Title box, enter **Encounter Count** and click OK to close the Graph dialog box.

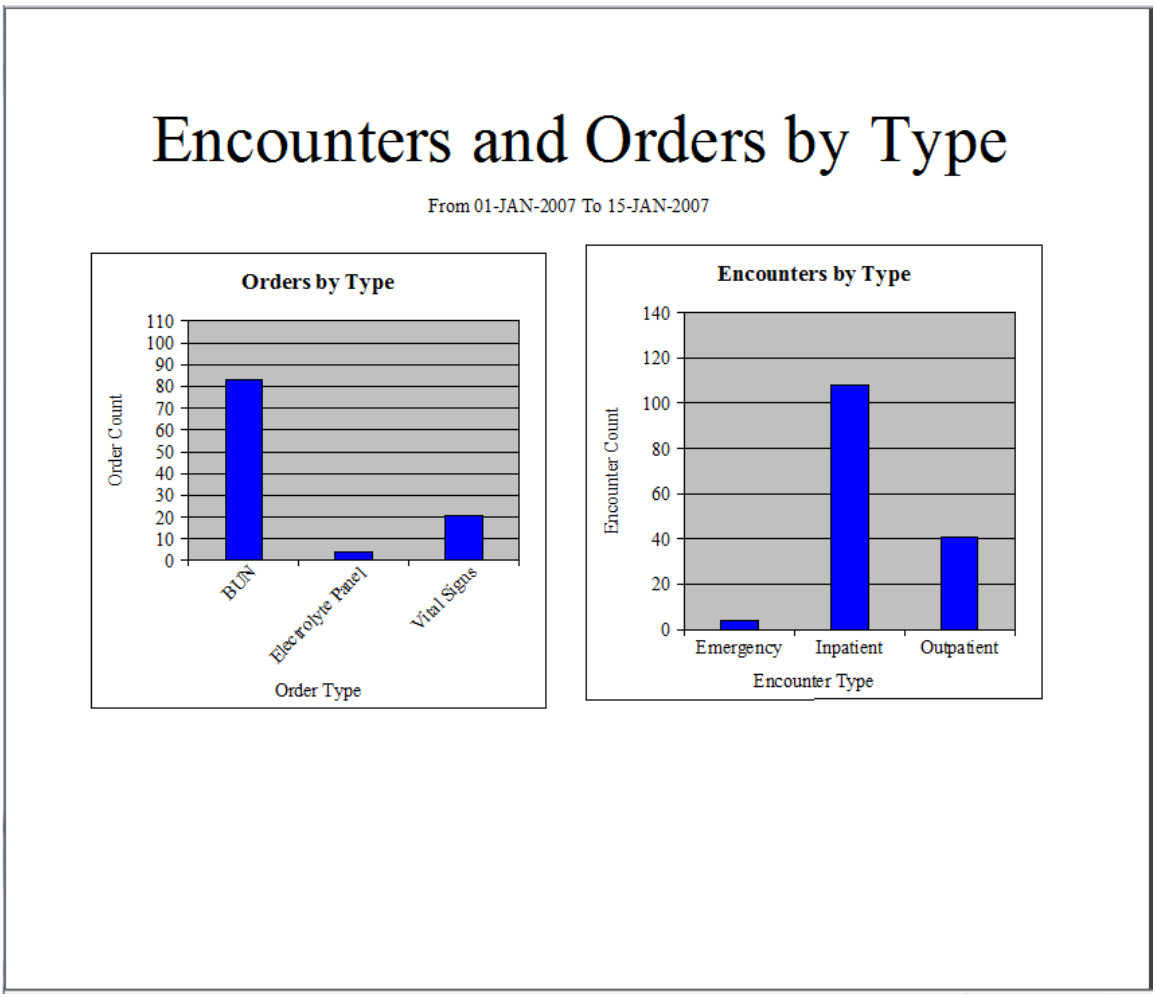
73. Click OK to close the Graph Properties dialog box. You your layout should be similar to the following screen:



74. From the Build menu, select Run "1_*Your_Initials*_Graph_Ords_Encs", or press CTRL+F5 to execute your layout program. Click Yes when prompted to save the layout. The prompt form opens.

75. At the prompt, use MINE for the output device and enter the starting and ending dates.

Your output should display similar to the following example. Your numbers, encounter types, and order types will differ based on the data that exists in your environment and the code values that you created and qualified on earlier in this example.



Creating a Table View Layout

A Table View is a layout program that generates its data in a table format with the data displayed using rows and columns. Using the Table View layout integrates the entire query that is associated with the layout. This means that each row of the query will be associated to a Reportwriter section. The flow of the program is dependent and connected to the rows on the table. One advantage of the Table View is that the table can be created as HTML. To demonstrate some of the common table functionality, we will create a report to display information about people, their orders, and order comments in table format. The final output will be a report formatted like the following example:

Person Order Comment Report

Page: 1 of 2

ID:	Name:	
3508235.00	Davis , Shawna	
	Dextrose 5% with 0.3% NaCl	4/10/2008 14:58 Dextrose 5% & NaCl 0.3% 17.2414 mg/kg/dose (weight: 58 kg) Potassium Chloride 2 mEq/mL intravenous solution 1.7241 mEq/kg/dose (weight: 58 kg)
	acetaminophen- codeine	2 tablets every 8 hours as needed Abdominal Cramps for 30
	acetaminophen-codeine	Contains codeine or a synthetic codeine
	acetaminophen-diphenhydramine	Take 1 tablet every night at bedtime 1 30 w
	acetaminophen_rk	mar notes
	dexamethasone	Take 20 tablets every 4 hours as needed Anxiety 4 Doses
	metoclopramide	1 tablet every 4 hours
	penicillin	5 mL 4 times a day for 7
	tobramycin	*STD CONC=2MG/ML* EXP=4 DAYS REFR ABX START DATE =__-__-__
	vecuronium	WARNING: This drug may cause respiratory depression. Facilities for artificial respiration must be immediately available.
ID:	**Name:**	
8035694.00	Deshpande, Neha	
	Consult to Social Services	Order entered secondary to documention family and or social concerns

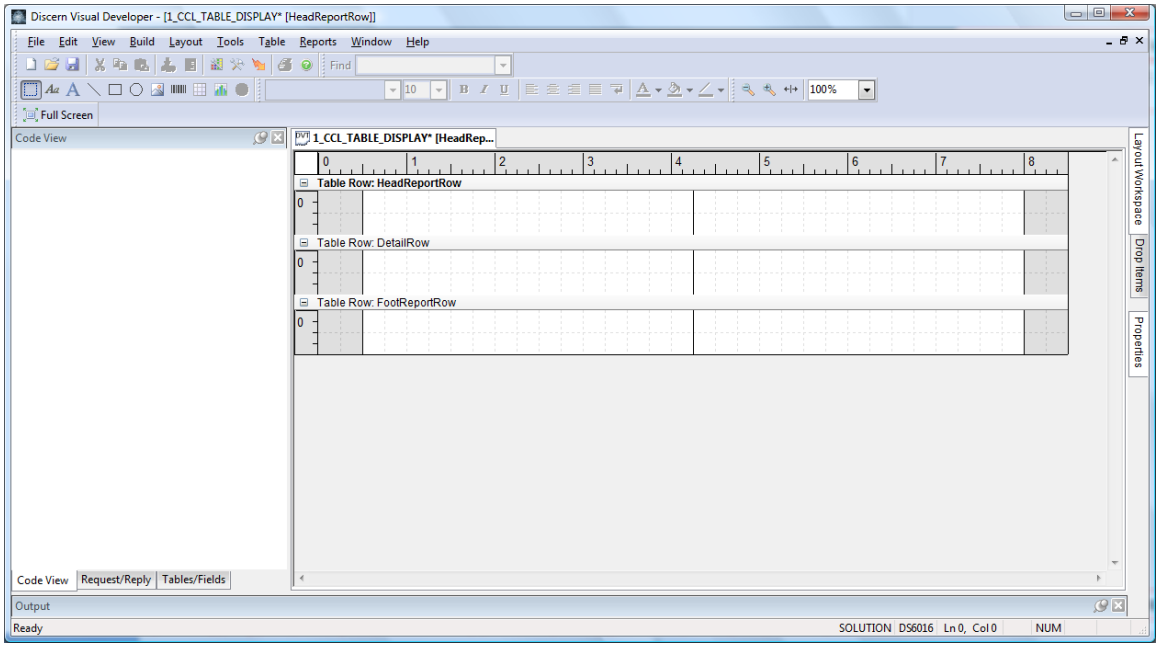
1. Using DVDev, from the File menu, select New. The New dialog box opens.

2. From the File Type list, select Layout Program.

3. In the Program Name box, enter **1_*your_initials*_table_display**, and click OK. The New Layout Program dialog opens.

4. Select Table View from the Report Layout option. Enter **2** for the Number of Columns: option. This option enables you to set the number of columns to place on the table.

5. Verify the PostScript option is selected on the Output Type option. After clicking the Finish button, three table rows will be created: HeadReportRow, DetailRow and FootReportRow. The Table should look like this:



By selecting the Table View, the initial table size is two columns and three rows, which are automatically associated to Reportwriter sections. The rows and columns are evenly spaced within the initial size of table.

The following query selects information about people and orders with order comments:

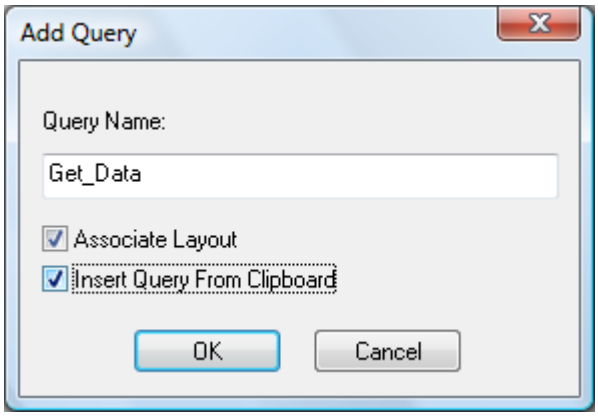
```
SELECT into "NL:"
  P.PERSON_ID,
  P.NAME_FULL_FORMATTED,
  O_CATALOG_DISP = UAR_GET_CODE_DISPLAY( O.CATALOG_CD ),
  L.LONG_TEXT,
  L.LONG_TEXT_ID,
  L.PARENT_ENTITY_NAME

FROM
  LONG_TEXT  L,
  ORDER_COMMENT  OC,
  ORDERS  O,
  PERSON  P

PLAN L  WHERE  L.PARENT_ENTITY_NAME = "ORDER_COMMENT"
JOIN OC WHERE  OC.LONG_TEXT_ID = L.LONG_TEXT_ID
JOIN O  WHERE  O.ORDER_ID = OC.ORDER_ID
JOIN P  WHERE  P.PERSON_ID = O.PERSON_ID

ORDER BY
  P.NAME_FULL_FORMATTED,
  P.PERSON_ID,
  O_CATALOG_DISP,
  0

WITH MAXREC = 100, NOCOUNTER, SEPARATOR=" ", FORMAT
```

6. Copy the query to the clipboard.

7. From the Tools menu, select Query Builder. The Add Queries dialog box opens.

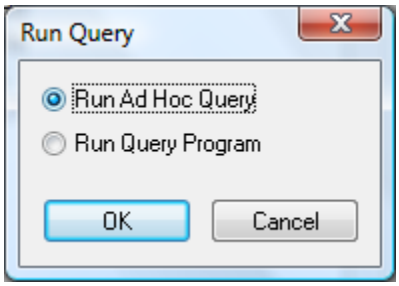
8. Click Add. The Add Query dialog box opens.

9. In the Query name box, enter **Get_Data**, select the Associate Layout and Insert Query From Clipboard options. Your Add Query dialog box should display similar to the following:



10. Click OK. The Discern Query Builder dialog box opens with the query populated from the clipboard.

11. Click Run Query. The following Run Query dialog box opens:



12. Select the Run Ad Hoc Query option and click OK. The Query Output window opens.

13. Verify your query returns information about people, their orders, and order comments, and then close the Query Output window. If necessary, modify the qualifications to get a good sample data set.

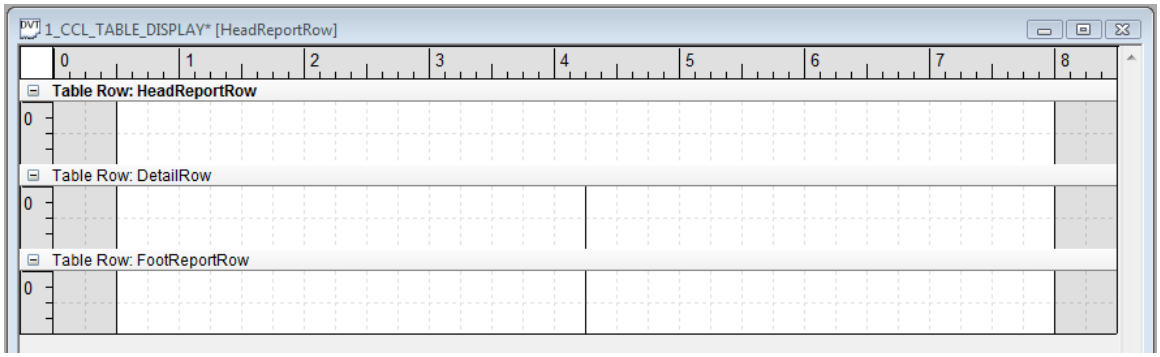
14. Once you have a good sample data set, close the Discern Query Builder window.

15. Click OK to close the Add Queries dialog box.

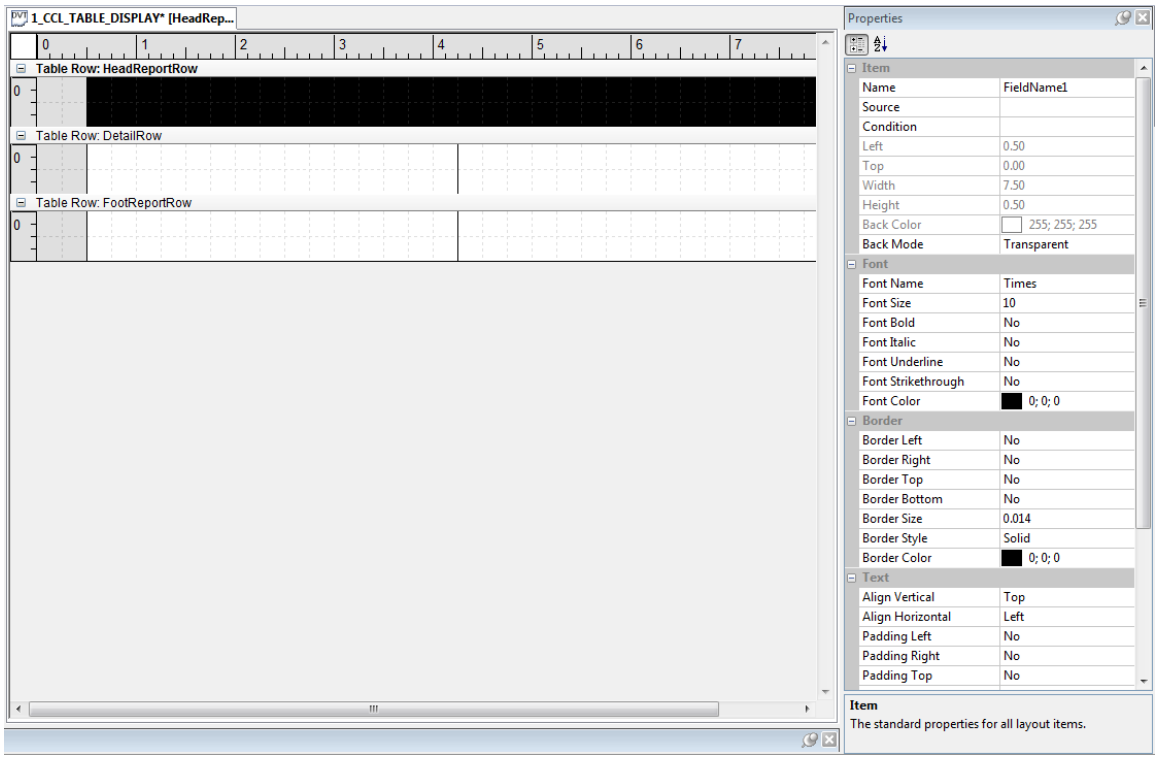
    Currently there are two cells in the HeadReportRow of your table. We want to use this row to display a report title, for that, we only need one cell.

16. Select the first cell in the HeadReportRow of your table.

17. Right-click and select Remove Cell from the context menu. The remaining cell fills in the entire width of the first row in your table.



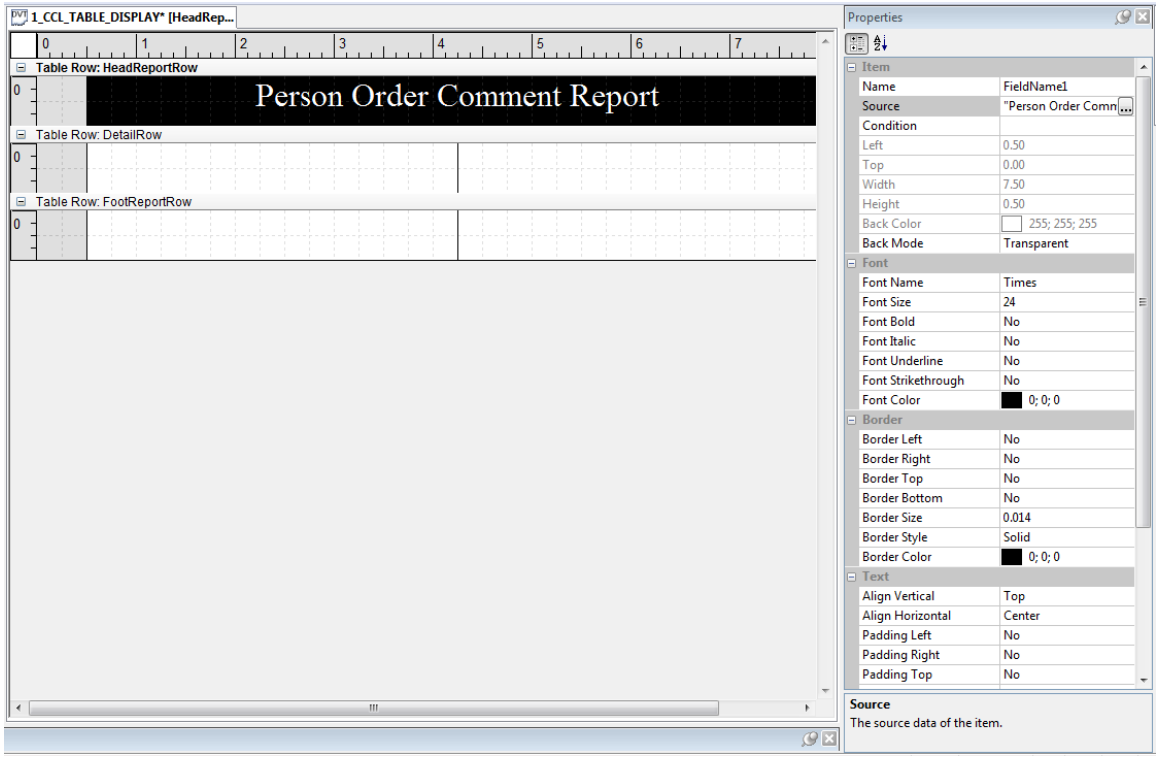
18. Double-click the remaining cell in the HeadReportRow of the table. The properties for that cell are displayed in the Properties dialog box as in the following example. The default name for the Name property varies, depending on which cell you removed.



19. In the Source box enter **"Person Order Comment Report"**. You can also double-click in the cell again and enter the text. The quotes must be placed around the text.

20. Modify the Font Size to **24** and the Align Horizontal property to **Center**

21. The following layout is displayed:



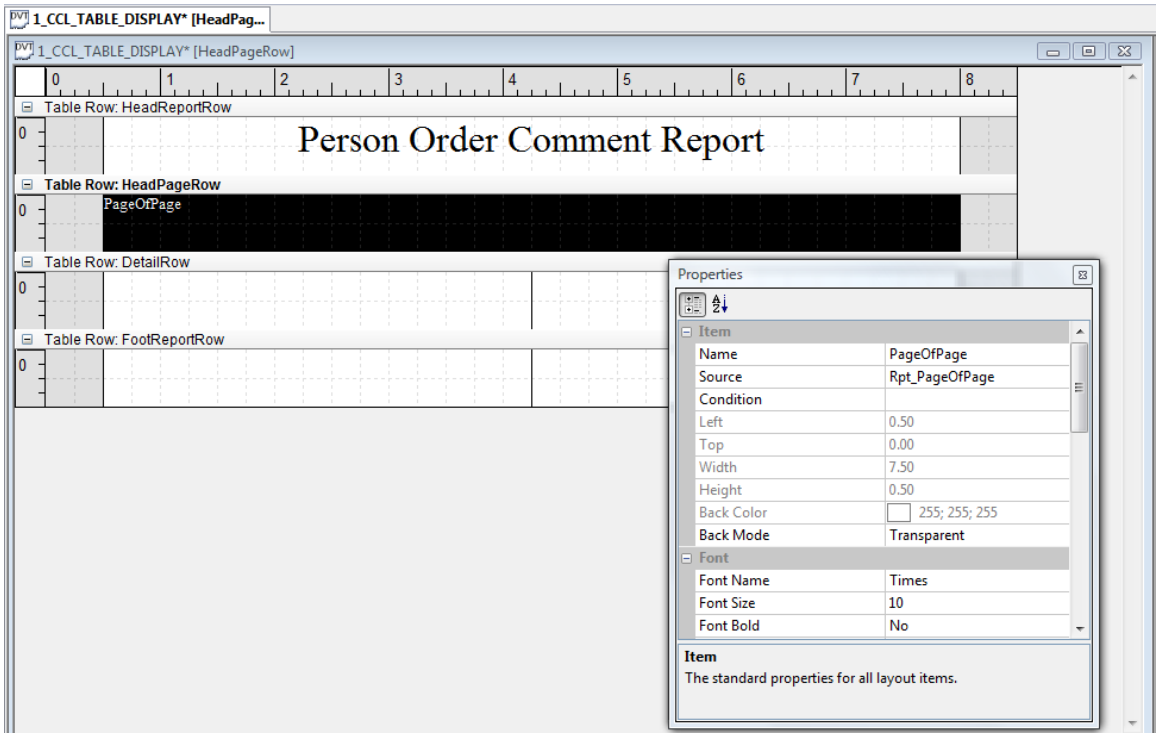
22. Check Head Page from the Select/Modify Section of the Workspace dialog box. If the Layout Workspace dialog box is not already displayed, select it from the View menu.

23. Select Yes to the message to create a new row that is associated with the Head Page Reportwriter section. A new row, called HeadPageRow, is created after the HeadReportRow. We want to use the HeadPageRow to display the page number for each page of the output.

24. Right-click the first cell in the HeadPageRow, and from the context menu, select Remove Cell. The remaining cell fills in the entire width of the HeadPageRow in your table.

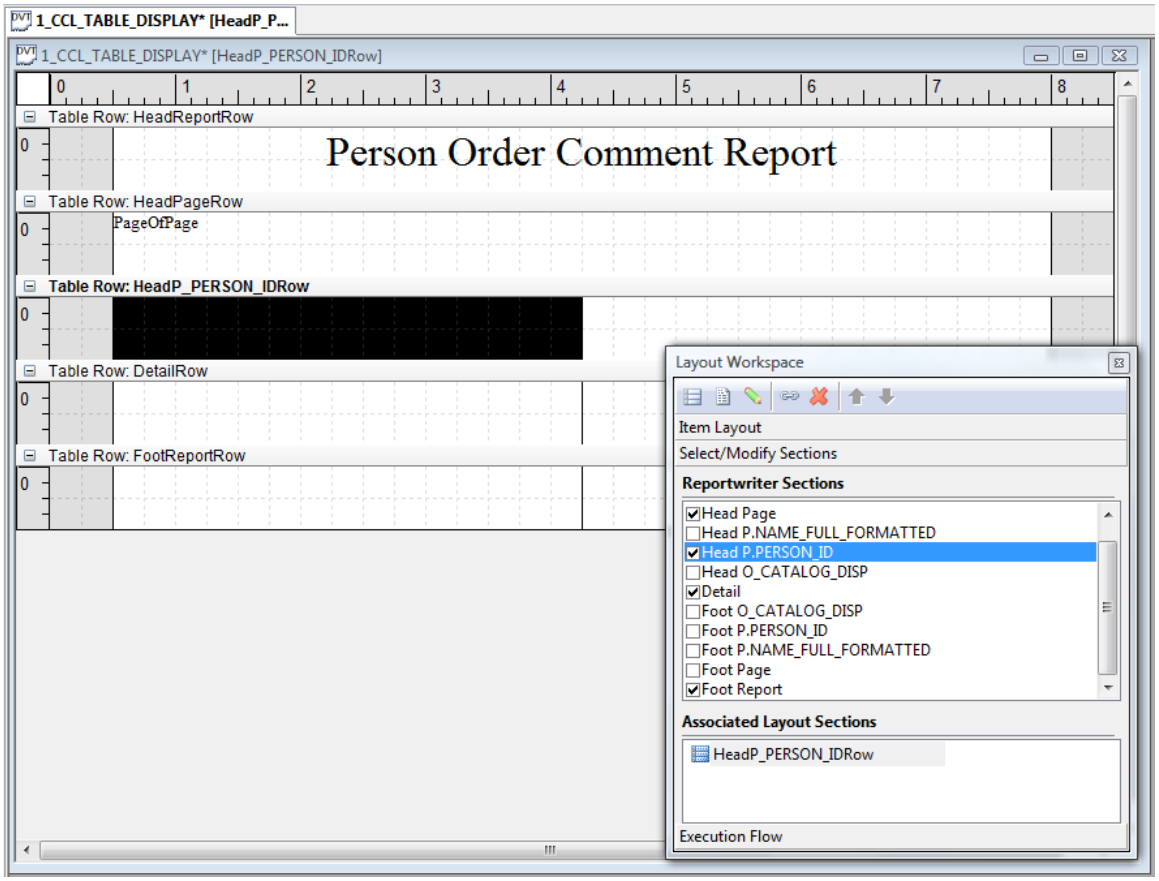
25. Click the remaining cell in the HeadPageRow row of the table and click the Properties dialog box. Modify the Name property to **PageOfPage**.

26. In the Source box, enter **Rpt_PageOfPage**, or double-click the cell to enter the variable. Layout Builder creates the special variable Rpt_PageOfPage to display the current page number and the total number of pages. The following layout is displayed:



27. Check the Head P.PERSON_ID from the Select/Modify Section of the Workspace dialog box.

28. Click Yes to the message to create a new row called HeadP_PERSON_IDRow that is associated to the Head P.PERSON_ID Reportwriter section. The following layout is displayed:

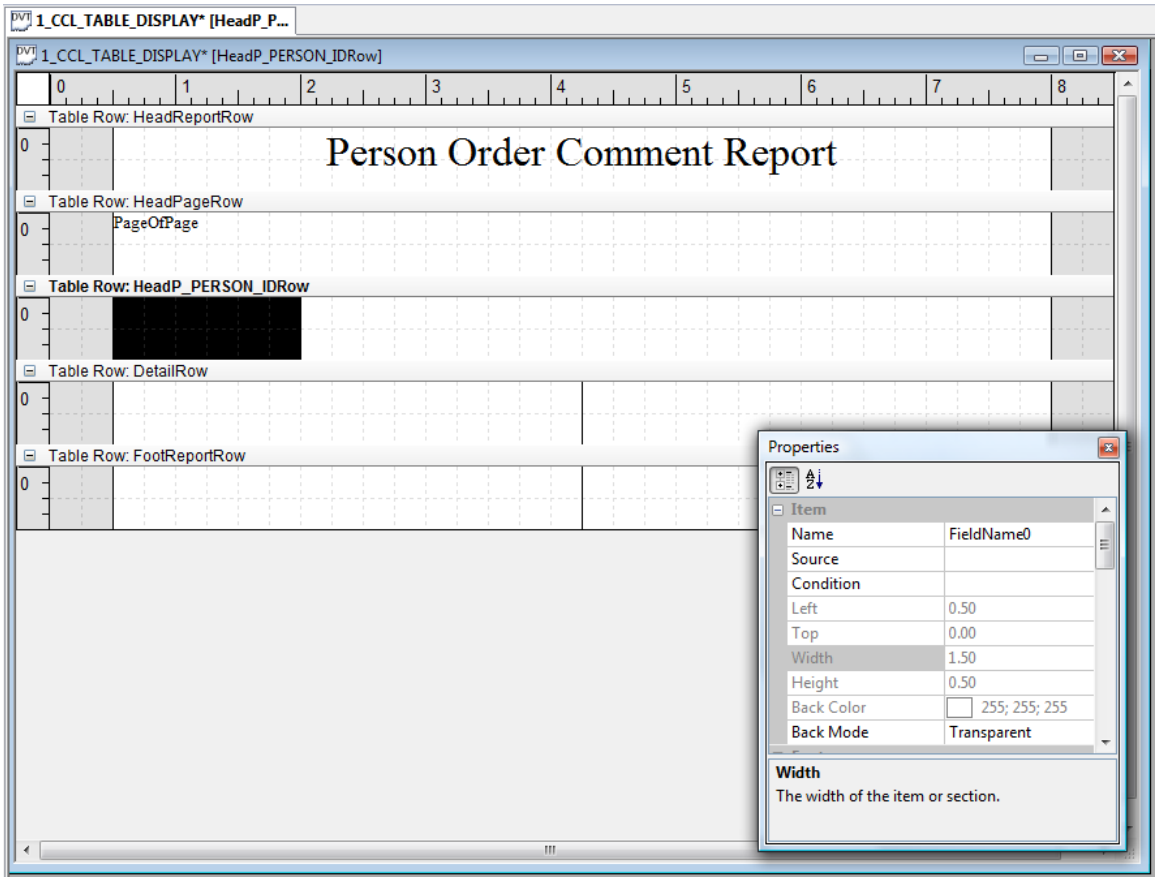


We want to use the HeadP_PERSON_IDRow of the table to display the column headings "ID:" and "Name:" for each person that is returned by the query. We want to use the DetailRow to display the person ID and name of each person returned by the query. Currently these rows are divided into two cells of equal size. The cells used to display the "ID:" column heading and the person ID do not need to be as large as the cell to display the actual person's name.

We have several options for changing the size of the cells. First if a single cell is selected and the pointer is moved over a vertical edge of that cell, the pointer changes to the horizontal resize ↔. When the pointer changes to the horizontal resize you can click and drag the vertical edge of the cell to change the size of the cell. If multiple, or no, cells are selected, the horizontal resize can be used to click and drag the vertical edge of all cells that share a vertical edge

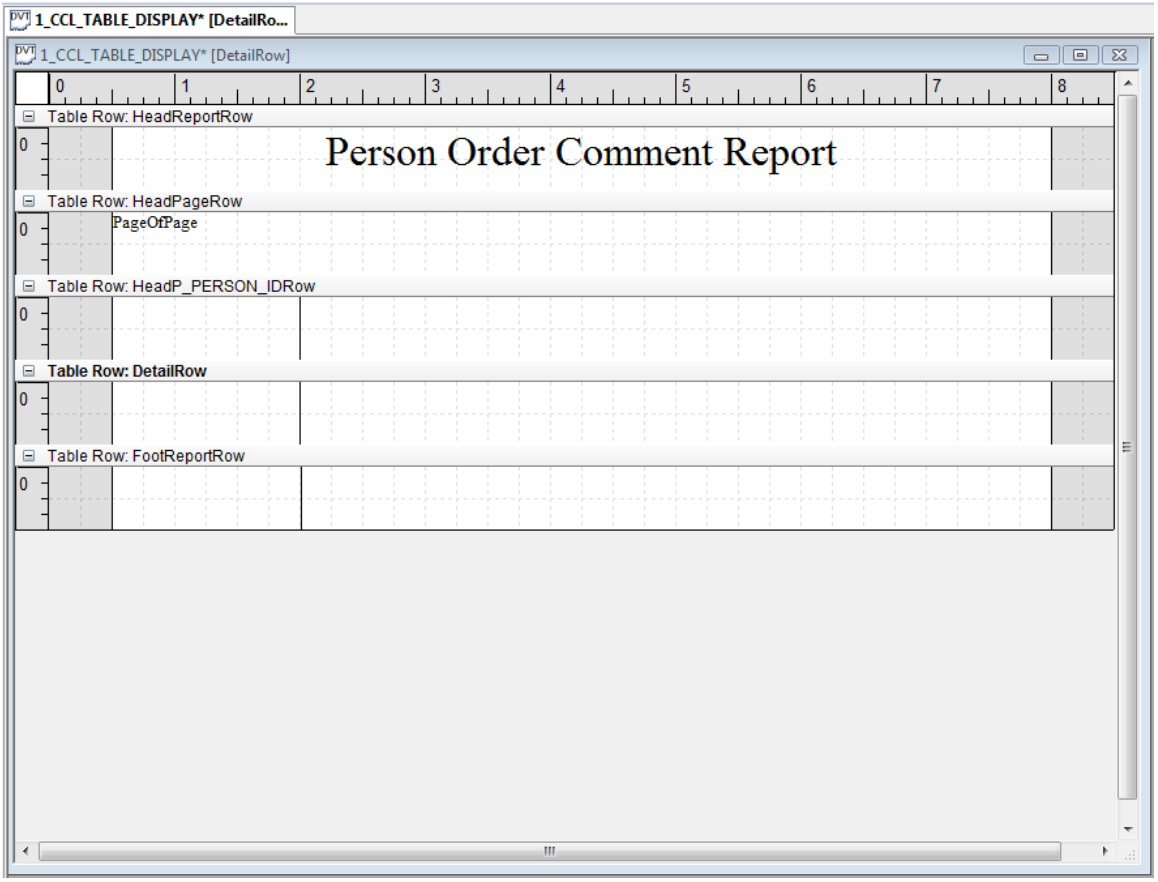
29. Click in the first cell of the HeadP_PERSON_IDRow to select it.

30. Move the pointer over the right vertical edge of the selected cell. When the pointer changes to the horizontal resize ↔, click and drag the vertical edge of the cell to the left to make the size of the cell about one and a half inches. (The grid lines on the layout are one fourth of an inch apart.) The following layout is displayed:



31. Move the pointer over the right vertical edge of the first cell in the DetailRow cell.

    When the pointer changes to the horizontal resize ↔, click and drag the vertical edge of the cell to the left to make the size of the cell about one and a half inches. No specific cell was selected, so all cells that share the same vertical edge are resized. The following layout is displayed:

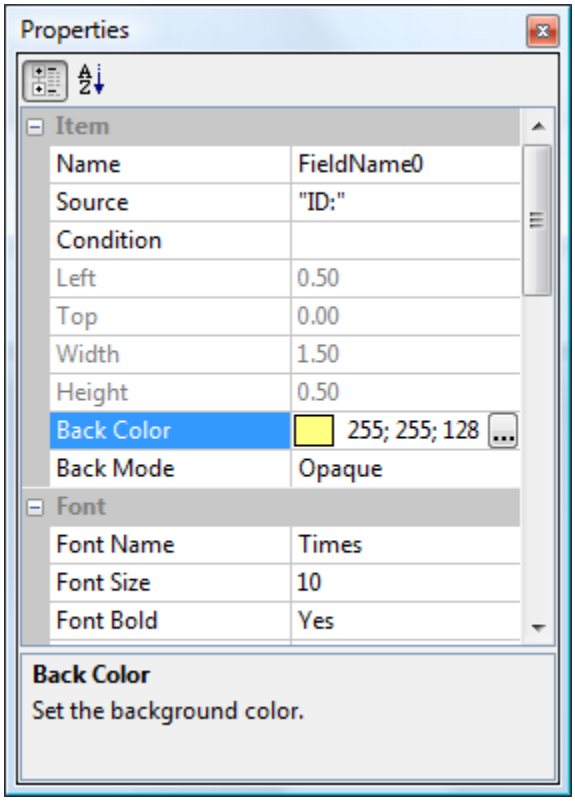


32. Click in the first cell of the HeadP_PERSON_IDRow. Click again and enter the text **"ID:"**. The text is added to the Source property.

33. Modify the Font Bold property to Yes.

34. Modify the Back Mode property to Opaque.

35. Modify the Back Color property to a light color. The following Properties dialog box is displayed:

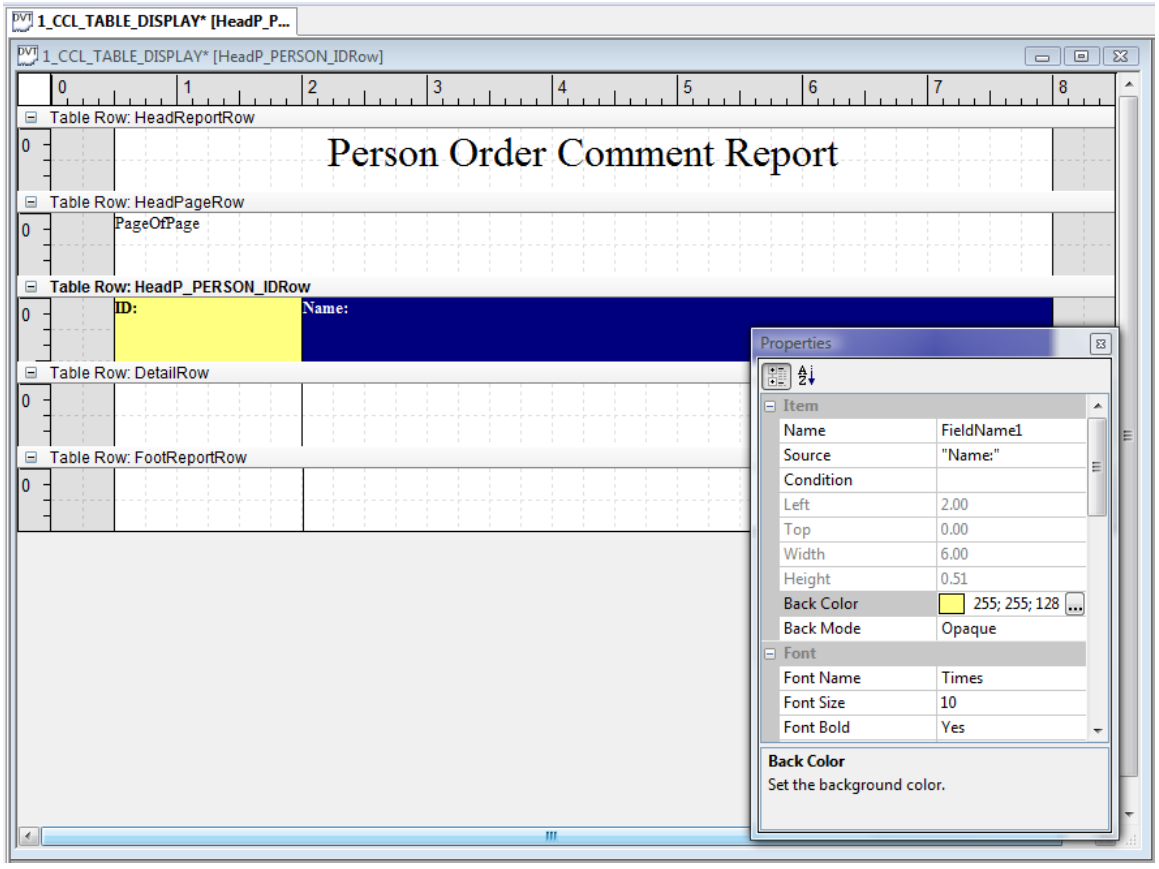


36. Click the cell to the right on the HeadP_PERSON_ID row. Click again and enter **"Name:"** . The text is added to the Source property.

37. Modify the Font Bold property to **Yes**.

38. Modify the Back Mode property to **Opaque**.

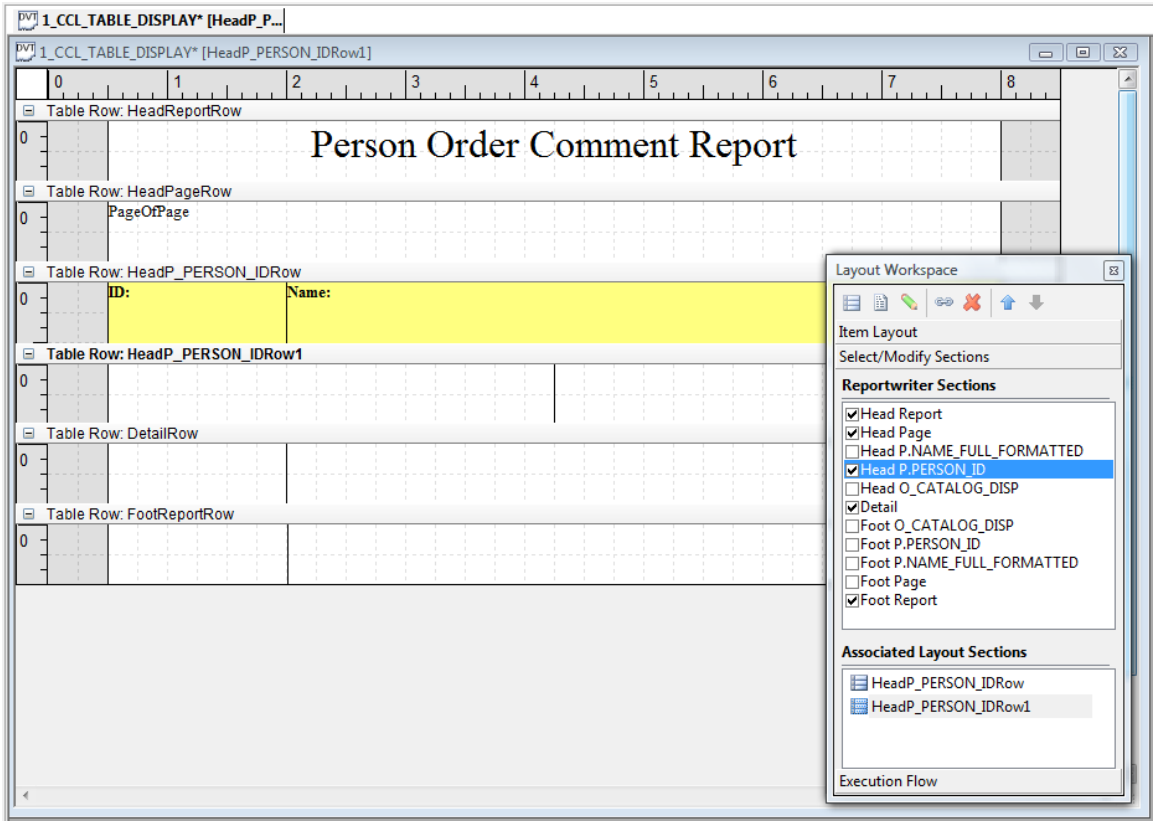
39. Modify the Back Color property to a light color. The following layout is displayed:



Note: The Name cell is displayed in black background and white font if it is still selected. The actual cell name in the Name property will vary based on which cells you removed in the earlier steps.

We want to only display the person ID and name one time per person. To do this, we need to create another row that is associated with Head P.Person_ID Reportwriter section.

40. Select the Head P.PERSON_ID (select the label, not the check box) from the Select/Modify Section of the Workspace dialog box. Click the New Table Row button,  . A new table row called HeadP_PERSON_IDRow1 is created that is associated to the Head P.PERSON_ID Reportwriter section and immediately follows the HeadP_PERSON_IDSection. The layout will look similar the following:



Since the person's ID will be in the first cell of the HeadP_PERSON_IDRow1, the size of the cell can be reduced to one and a half inch.

41. Move the pointer over the right vertical edge of the cell. When the pointer changes to the horizontal resize ↔, click and drag the vertical edge of the cell to the left to make the size of the cell about one and a half inches.

42. Click in the first cell of the HeadP_PERSON_IDSection row to select it. Access the Properties for the cell.

43. In the Name box, enter **PID**.

44. In the Source box, enter **P.PERSON_ID**, or click in the Source box to activate the ellipsis [...] button to open the Layout Source [PID] dialog box and select P.PERSON_ID from the Select Fields tree in the Variables column.

45. Click in the second cell of the HeadP_PERSON_IDRow1 to select it. Access the Properties for the cell.

46. In the Name property, enter **Name**.

47. In the Source box, enter **P.NAME_FULL_FORMATTED** or click in the Source box to activate the ellipsis [...] button to open the Layout Source [Name] dialog box and

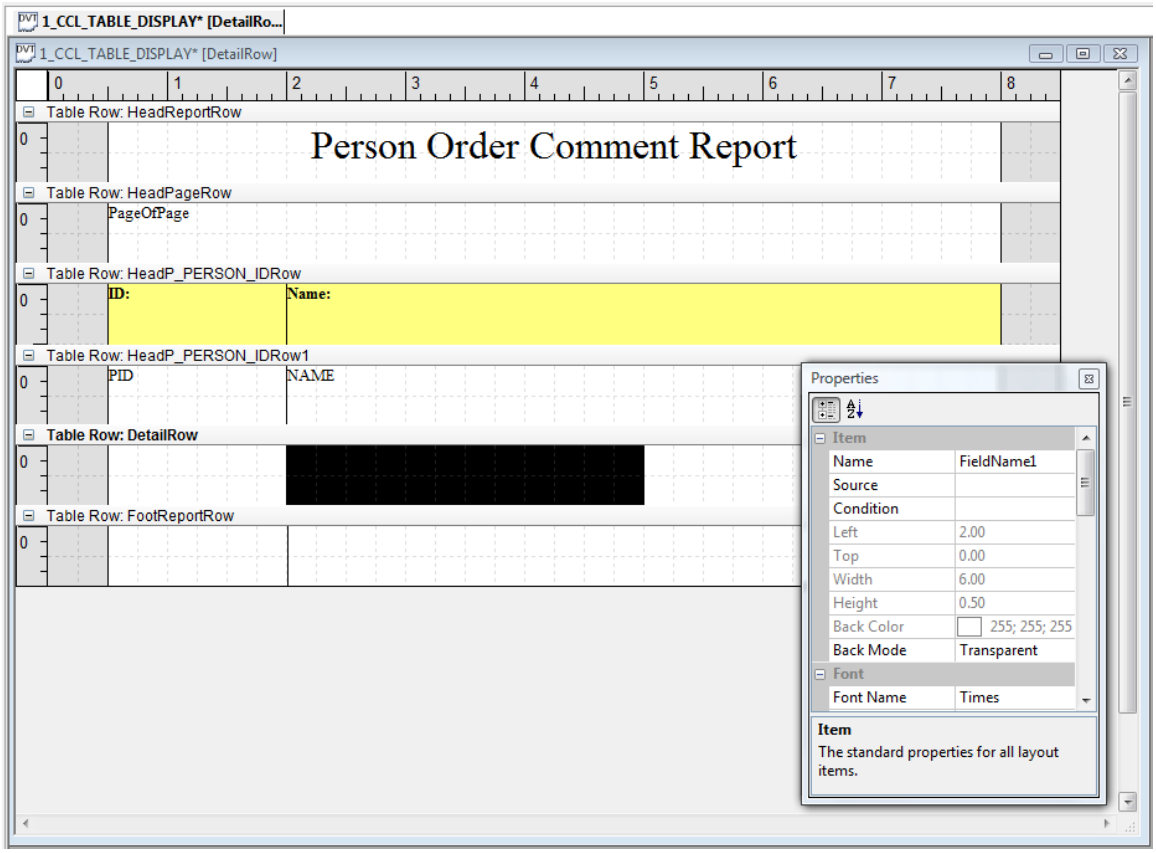
select P.NAME_FULL_FORMATTED from the Select Fields tree in the Variables column.

We want to use the DetailRow of the table to display all of the display values of the catalog codes and the order comments. We also want to indent this information so the display value of the catalog code is under the person's name and the order comments are to the right of that. Because rows on the table must extend across the entire table, in order to indent the order information we need to leave an empty cell on the left side of the row. Since the last row of the table consists of only two cells, and since we need three cells to accomplish our objective, we need to split one of the cells into two cells.

48. Click in the cell on the right side of the DetailRow to select it.

    From the Table menu, select Split Cell, or right click in the cell and select Split Cell from the context menu. The Split Cell option divides the existing cell into two cells of equal size.

49. Click in the middle cell on the DetailRow of the table. Access the Properties for the cell.. The following layout is displayed:



50. In the Name box, enter **ODisp**.

51. In the Source box, enter **O_CATALOG_DISP** or click in the Source box to activate the ellipsis  button to open the Layout Source [ODisp] dialog box and select O_CATALOG_DISP from the Select Fields list in the Variables column.

52. Move your pointer over the right vertical edge of the ODisp cell. When the pointer changes to the horizontal resize  , right-click and drag the vertical edge of the cell to the left to make the size of the cell about two inches.

53. Click in the cell to the right of the ODisp cell to select it. The properties for that cell are displayed in the Properties dialog box.

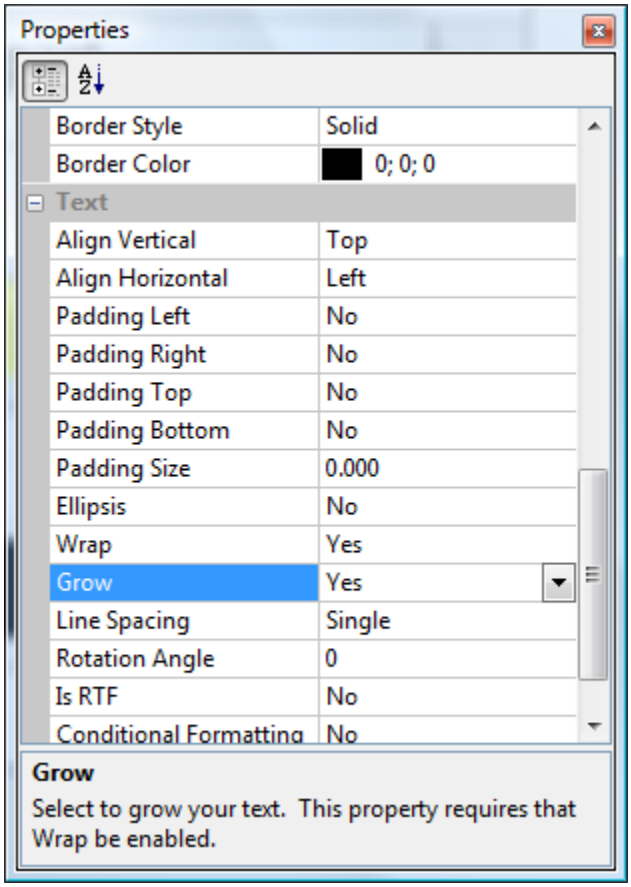
54. In the Name box, enter **OComment**.

55. In the Source box, enter **L.LONG_TEXT**, or click in the Source property box to activate the ellipsis  button to open the Layout Source [Name] dialog box and select L.LONG_TEXT from the Select Fields list in the Variables column.

56. From the Text category, modify the Wrap property to Yes.

57. From the Text category, modify the Grow property to Yes.

    Setting the Wrap and Grow properties to Yes enables the Long_Text to be wrapped across multiple lines within the cell and to grow the size of the cell if needed to display all of the Long_Text.



58. From the Build menu, select Run "*Your_Layout_Program*", click the run Prompt Program button, or press CTRL+F5 to execute your layout program.

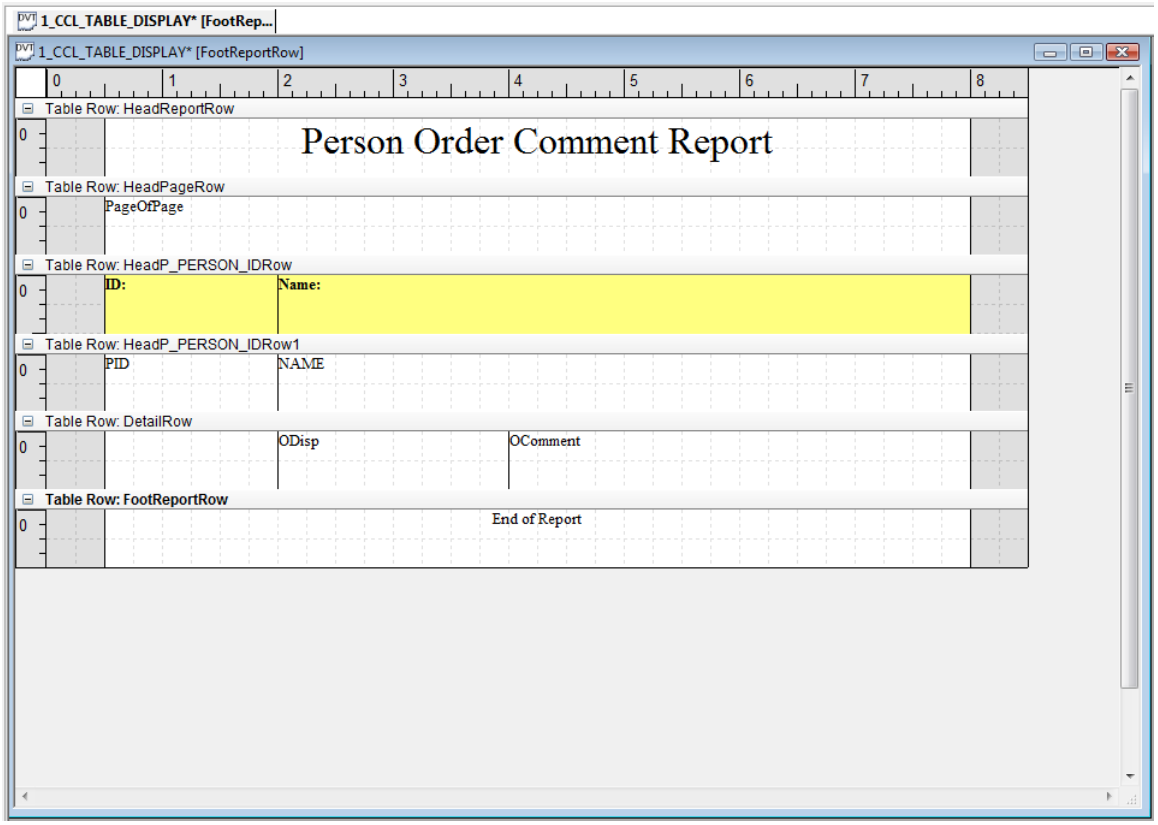
59. Select Yes when prompted to save the layout program. Your output should resemble the following output:

Person Order Comment Report

Page: 1 of 2

ID:	Name:	
3508235.00	Davis , Shawna	
	Dextrose 5% with 0.3% NaCl	4/10/2008 14:58 Dextrose 5% & NaCl 0.3% 17.2414 mg/kg/dose (weight: 58 kg) Potassium Chloride 2 mEq/mL intravenous solution 1.7241 mEq/kg/dose (weight: 58 kg)
	acetaminophen- codeine	2 tablets every 8 hours as needed Abdominal Cramps for 30
	acetaminophen-codeine	Contains codeine or a synthetic codeine
	acetaminophen-diphenhydramine	Take 1 tablet every night at bedtime 1 30 w
	acetaminophen_rk	mar notes
	dexamethasone	Take 20 tablets every 4 hours as needed Anxiety 4 Doses
	metoclopramide	1 tablet every 4 hours
	penicillin	5 mL 4 times a day for 7
	tobramycin	*STD CONC=2MG/ML* EXP=4 DAYS REFR ABX START DATE =__-__-__
	vecuronium	WARNING: This drug may cause respiratory depression. Facilities for artificial respiration must be immediately available.
ID:	**Name:**	
8035694.00	Deshpande, Neha	
	Consult to Social Services	Order entered secondary to documention family and or social concerns

60. Close the output display window.
61. Click the first cell of the FootReportRow to select the properties.
62. Enter **"End of Report"** as the source for this cell.
63. Set the Align Horizontal property to Center.
64. To remove the other cell from FootReportRow, right-click in the second cell, and select Remove Cell from the context menu. The following layout should be displayed:



65. From the Build menu, select Run “Your_Layout_Program”. Click the Run Prompt Program button, or press CTRL+F5 to execute your layout program.

66. Select Yes when prompted to save the layout program. The Report Output window opens.

67. Click the Next Page button to go to the last page of the report and scroll to the bottom of the page to verify the End of Report label is displayed. Your output should resemble the following:

ID:	Name:	
3508235.00	Davis , Shawna	
	acetaminophen_rk	mar notes
	metoclopramide	1 tablet every 4 hours
	penicillin	5 mL 4 times a day for 7
	vecuronium	WARNING: This drug may cause respiratory depression. Facilities for artificial respiration must be immediately available.
End of Report		

Creating HTML Output

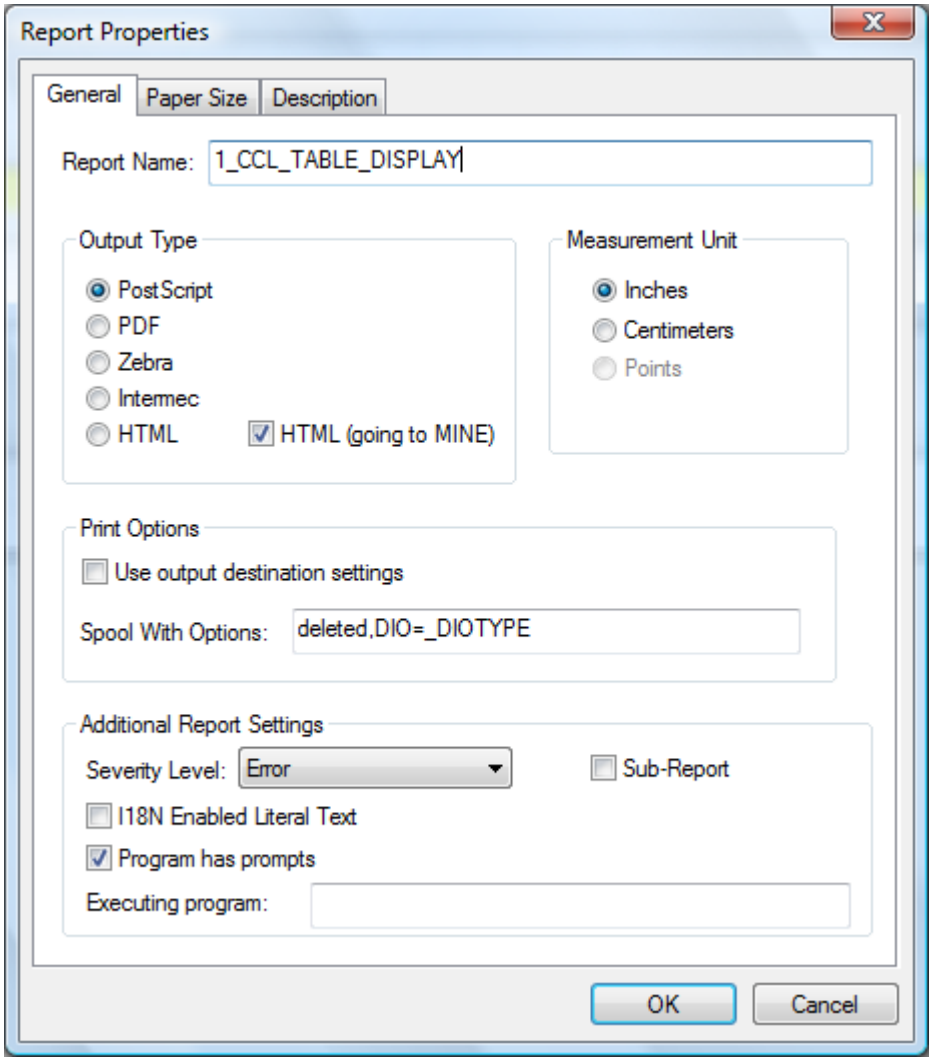
This section assumes you have already completed the Creating a Table View section of this tutorial.

There may be times when you want to create a report that supplies high level information, but then offers the ability to navigate down to a finer level of detail. Layout Builder enables you to create Table Views and layout programs that produce output in HTML format. Using HTML format enables you to create links which can then be used to execute further programs allowing the user to navigate down to see additional details. Using Table Views is the recommended type of layout to use when creating HTML output. Table Views have intrinsic tabular type logic which lends itself to creating HTML output. However, using a normal layout program is acceptable when consideration is given to the tabular type logic flow.

For existing layout programs, the Report Properties dialog box provides the option to create output in HTML table format.

One way to create HTML output is by selecting the HTML (going to MINE) option for the Report Property.

Doc ID: A1001001A08G07B34433A94851

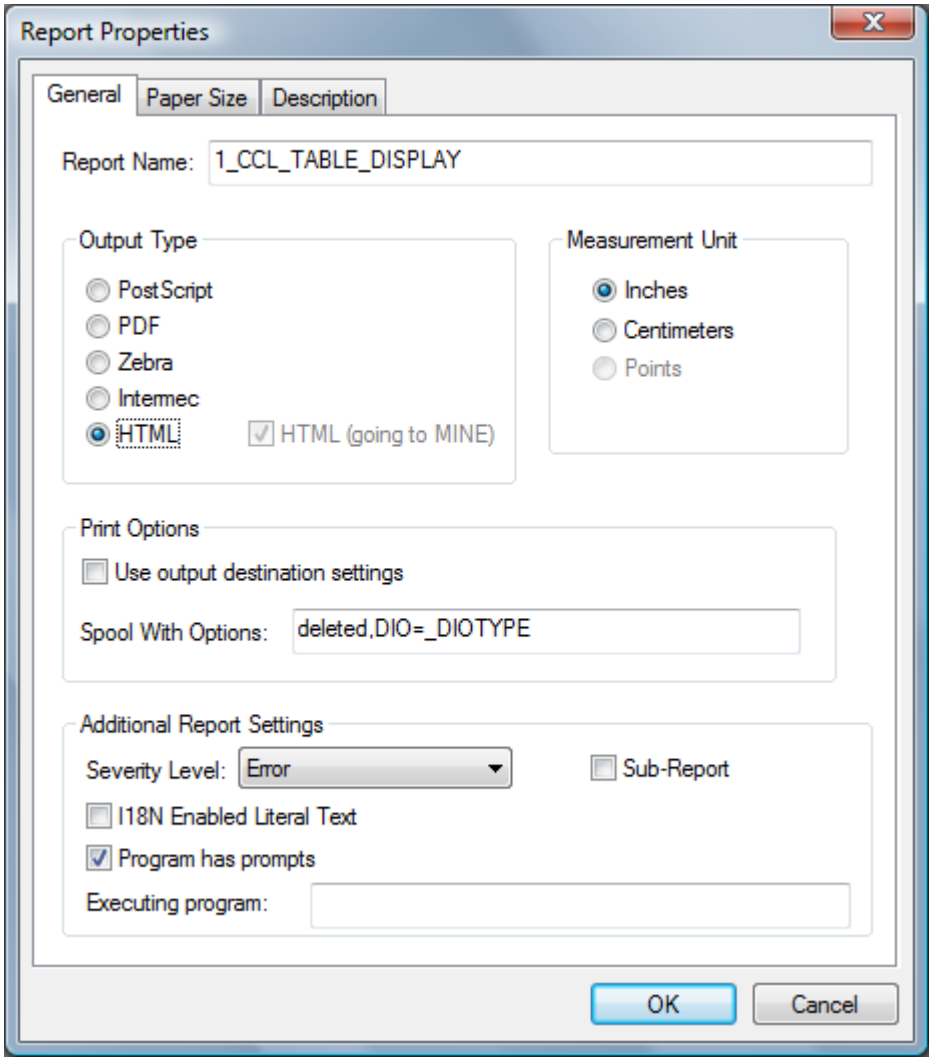


Using this method allows you to flex the output to different types of formats.

When a layout program is created, DVDev creates a prompt form for the layout program. The prompt form contains a prompt for the output device. The output of the layout is sent to the output device. If a layout program has the Output Type option of HTML (going to MINE) selected and if MINE (in uppercase) is used as the output device, the output of the layout program is displayed on your screen in HTML format. If any value other than uppercase MINE is used as the output device, the output is sent to that device in the specified output format.

For example, if the PostScript Output option is selected and the HTML (going to MINE) option is checked and MINE is entered or selected at the prompt for output device, the output of the layout program is displayed on the screen in HTML format. If a PostScript printer queue name is used as the output device, the output of the layout program is sent to that PostScript printer queue in PostScript format. It is important to understand that MINE must be entered with uppercase characters to create HTML output. If Mine (not all uppercase) is entered as the output device, the results of the layout program would be displayed on the screen in PostScript format.

The alternative way to create HTML output is to select the HTML option for the Output Type.



Selecting the HTML option locks in the format type. When selecting MINE for the output device on the first prompt, the display of the output will be in HTML format. Typing a file name for the output device on the first prompt sends the output to a file in HTML format.

For any new programs, the selection to use the HTML options will available on the New Layout Program dialog box.

Using HTML output enables you to create linked reports you can use to navigate down through multiple levels to see additional detail. To demonstrate this functionality, we will create a report that displays information about people. For each person that is displayed, we will create a link to a report that displays information about that person's encounters. For each encounter we will create a link to a report to display information about the orders for that encounter. To build these reports, we need to identify the person IDs of some people, and the encounter IDs of some of their encounters for use in testing. The following query selects information about people, their encounters, and orders and can be used to find some testing data:

```
SELECT
  P.PERSON_ID,
  P.NAME_FULL_FORMATTED,
  P.NAME_LAST_KEY,
  P.NAME_FIRST_KEY,
  E.ENCNTR_ID,
  E_ENCNTR_TYPE_CLASS_DISP = UAR_GET_CODE_DISPLAY(E.ENCNTR_TYPE_CLASS_CD),
  O.ORDER_ID,
  O_CATALOG_DISP = UAR_GET_CODE_DISPLAY(O.CATALOG_CD)
FROM
  PERSON P,
  ENCOUNTER E,
  ORDERS O
PLAN P WHERE P.PERSON_ID > 0.0
JOIN E WHERE P.PERSON_ID = E.PERSON_ID
JOIN O WHERE E.ENCNTR_ID = O.ENCNTR_ID
ORDER BY
  P.PERSON_ID,
  E.ENCNTR_ID,
  O_CATALOG_DISP
WITH MAXREC = 100, NOCOUNTER, SEPARATOR=" ", FORMAT
```

1. Using DVDev, from the File menu, select New. The New dialog box opens.

2. From the File Type list, select Blank and click OK.

3. Copy the select statement from above and paste it into the blank file in DVDev.

4. To execute the select statement, Press Ctrl+Q, or from the Build menu, select the Run Ad Hoc Query.

5. Review the output from the query and record several person_ids, encntr_ ids, last_name_keys, and first_name_keys. We will use this information while building and testing the linked reports.

   Now that you have identified some values to use for testing, we can begin building the reports that will eventually be linked to provide the navigate down capability. First, we will create the report that displays information about orders for an encounter. To test this report we will create it as a Table View layout program that prompts for an output device and encntr_id and displays the results in a table.

6. Using DVDev, from the File menu, select New. The New dialog box opens.

7. From the File Type list, select Layout Program.

8. In the Program Name box, enter **1_*Your_Initials*_Linked_Orders**, and click OK. The New Layout Program dialog box opens.

9. Select Table View from the Report Layout option. Enter **3** for the Number of Columns: option.

10. Select the HTML Output Type. After clicking the Finish button, three rows will be created: HeadReportRow, DetailRow and FootReportRow. Each row will contain three equally spaced cells.

11. From the Tools menu, select Prompt Builder. The Prompt Builder dialog box opens with a single control for an output device.

12. Click Add to add a second prompt to ask for an encntr_id from the user.

13. On the General Tab, set the following:

Prompt Display:	Enter an Encntr_ID
Prompt Name:	EID
Control Type:	Text Edit
Prompt Type:	Expression

14. Click Save to save the prompt form and close the Prompt Builder dialog box.

The following query selects information about orders for a specific encntr_id that is entered at the EID prompt you created.

```
SELECT into "NL:"
  O.ORDER_ID,
  O_CATALOG_DISP = UAR_GET_CODE_DISPLAY(O.CATALOG_CD),
  O.ORIG_ORDER_DT_TM ";;q"
FROM ORDERS  O
WHERE  O.ENCNTR_ID =  $EID
ORDER BY
  O.ORIG_ORDER_DT_TM,
  O_CATALOG_DISP
WITH MAXREC = 100, NOCOUNTER, SEPARATOR=" ", FORMAT
```

15. Copy the above query to the clipboard.

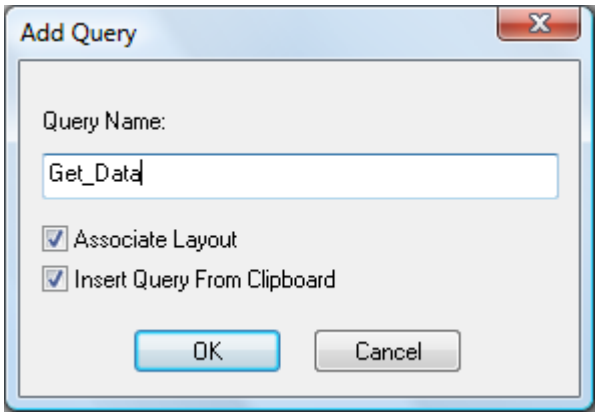
16. From the Tools menu, select Query Builder. The Add Queries dialog box opens.

17. Click Add to add a new query. The Add Query dialog box opens.

18. In the Query Name box, enter **Get_Data**.

19. Select the Associate Layout and Insert Query From Clipboard options.

Your Add Query dialog should be similar the following:



20. Click OK to add the query. The Discern Query Builder dialog box opens.

Doc ID: A1001001A08G07B34433A94851

21. Click Close to close the Discern Query Builder dialog box.

22. Click OK to close the Add Queries dialog box.

23. Click in the first cell of the HeadReportRow on your table. The Properties dialog box displays the information for the call you selected.

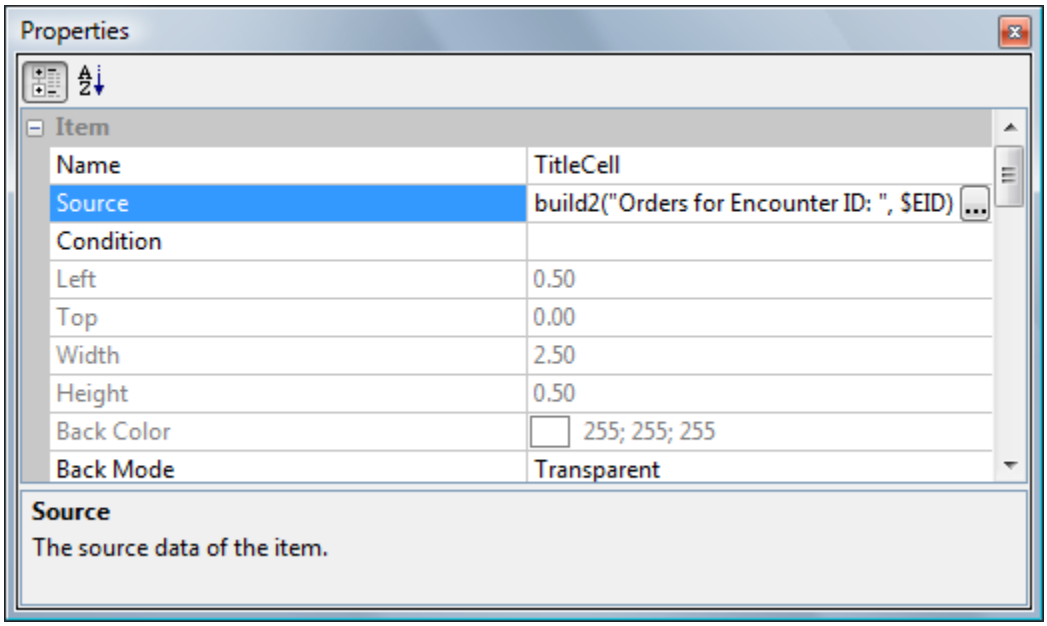
24. Modify the Name to **TitleCell**.

25. In the Source box, enter the following command:

```
build2("Orders for Encounter ID: ", $EID)
```

To create this command, click in the Source box to activate the ellipsis button, and click the ellipsis button to open the Layout Source [TitleCell] dialog. The Build2() function can be selected from the CCL Functions list in the Functions column.

Your Properties dialog box should resemble the following:



26. Click in the first cell of the DetailRow on your table.

27. In the Properties dialog box, modify the Name property to **Order_ID**.

28. In the Source box, enter **O.ORDER_ID**, or click in the Source property box to activate the ellipsis button to open the Layout Source [Order_ID] dialog box and select O.ORDER_ID from the Select Fields list in the Variables column.

29. Click in the middle cell of the DetailRow on your table.

30. In the Properties dialog box, modify the Name property to **Cat_Disp**.

31. In the Source box, enter **O_CATALOG_DISP,** or click in the Source property box to activate the ellipsis button to open the Layout Source [Cat_Disp] dialog box and select O_CATALOG_DISP from the Select Fields list in the Variables column.

32. Click in the cell on the right side of the DetailRow on your table.

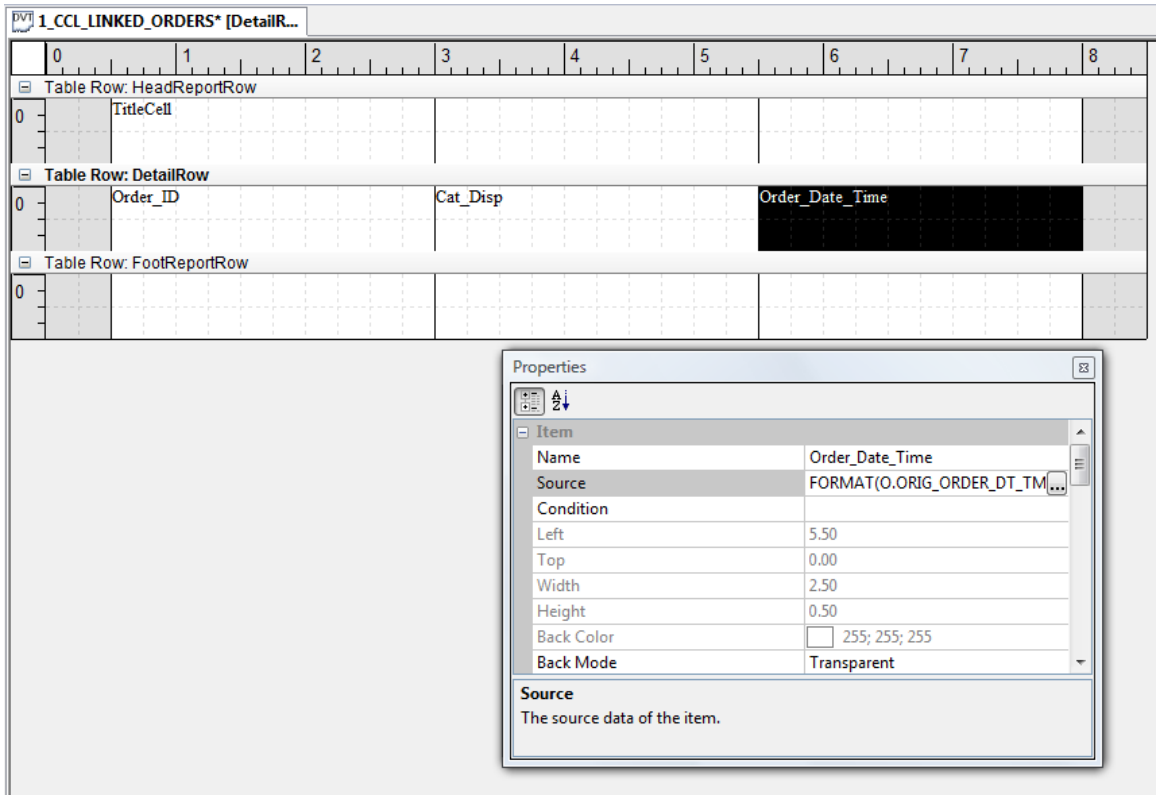
33. In the Properties dialog box, modify the Name property to **Order_Date_Time**.

34. In the Source box, enter the following command:

```
FORMAT(O.ORIG_ORDER_DT_TM, "@SHORTDATETIME")
```

    To create this command, click in the Source box to activate the ellipsis button, and click the ellipsis button to open the Layout Source [Order_Date_Time] dialog. The Format() function can be selected from the CCL Functions list in the Functions column and the O.Orig_Order_DT_TM can be selected from the Select Fields list in the Variables column.

The following layout is displayed:



35. Save your 1_*Your_Initials*_Linked_Orders layout.

36. From the Build menu, select Run "1_*your_initials*_Linked_Orders", or press CTRL+F5 to execute your layout program. The Discern Prompt dialog box opens.

Doc ID: A1001001A08G07B34433A94851

37. Select MINE as the output device, enter one of the encntr_ids you recorded earlier at the *Enter an Encntr_ID* prompt, and click Execute.

The Report Output window opens and the output of your 1_*Your_Initials*_Linked_Orders layout looks something like the following:

Orders for Encounter ID: 590102		
592919.00	Hemogram	11/26/03 08:40:28
592920.00	Prostate Specific Antigen	11/26/03 08:40:31
595609.00	hepatitis A vaccine	01/06/04 03:34:49
595614.00	hepatitis A-hepatitis B vaccine	01/06/04 03:42:00
597619.00	amoxicillin	01/06/04 07:49:08

The 1_Y*our_Initials*_Linked_Orders layout program is the lowest level of detail in our linked reports. We will now create the intermediate level of our linked reports. For this level, we will create a report that displays information about all encounters for a specific person and provides a link that will pass the encntr_id and execute the 1_Y*our_Initials*_Linked_Orders report created above. To test the intermediate report, we will create it as a layout program that prompts for an output device and person _id and, displays the results in a table.

38. From the File menu, select New. The New dialog box opens.

39. From the File Type list, select Layout Program.

40. In the Program Name box, enter **1_*Your_Initials*_Linked_Encntrs**, and click OK. The New Layout Program dialog box opens.

41. Select Table View from the Report Layout option. Enter **3** for the Number of Columns: option.

42. Select the HTML option for the Output Type. After clicking the Finish button, three rows will be created: HeadReportRow, DetailRow and FootReportRow. Each row will contain three equally spaced cells.

43. From the Tools menu, select Prompt Builder. The Prompt Builder dialog box opens with a single control for an output device.

44. Click Add to add a second prompt to ask for a person_id from the user.

45. On the General Tab set the following:

Prompt Display:	Enter a Person_ID
Prompt Name:	PID
Control Type:	Text Edit
Prompt Type:	Expression

46. Click Save to save the prompt form and close the Prompt Builder dialog box.

The following query selects information about encounters for a specific person_id that is entered at the PID prompt you created.

```
SELECT INTO "NL:"
  E.ENCNTR_ID,
  E_ENCNTR_TYPE_CLASS_DISP = UAR_GET_CODE_DISPLAY(E.ENCNTR_TYPE_CLASS_CD)
FROM
  ENCOUNTER  E
WHERE E.PERSON_ID = $PID
WITH MAXREC = 100, NOCOUNTER, SEPARATOR=" ", FORMAT
```

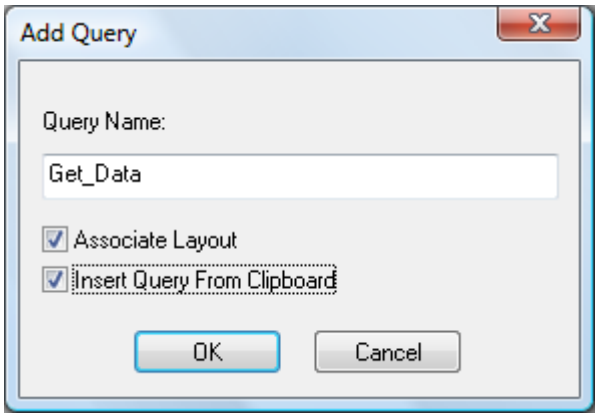
47. Copy this query to the clipboard.

48. From the Tools menu, select Query Builder. The Add Queries dialog box opens.

49. Click Add to add a new query. The Add Query dialog box opens.

50. In the Query Name box, enter **Get_Data**.

51. Select the Associate Layout and the Insert Query From Clipboard options. The following Add Query dialog box is displayed:



52. Click OK to add the query. The Discern Query Builder dialog box opens.

53. Click Close to close the Discern Query Builder dialog box.

54. Click OK to close the Add Queries dialog.

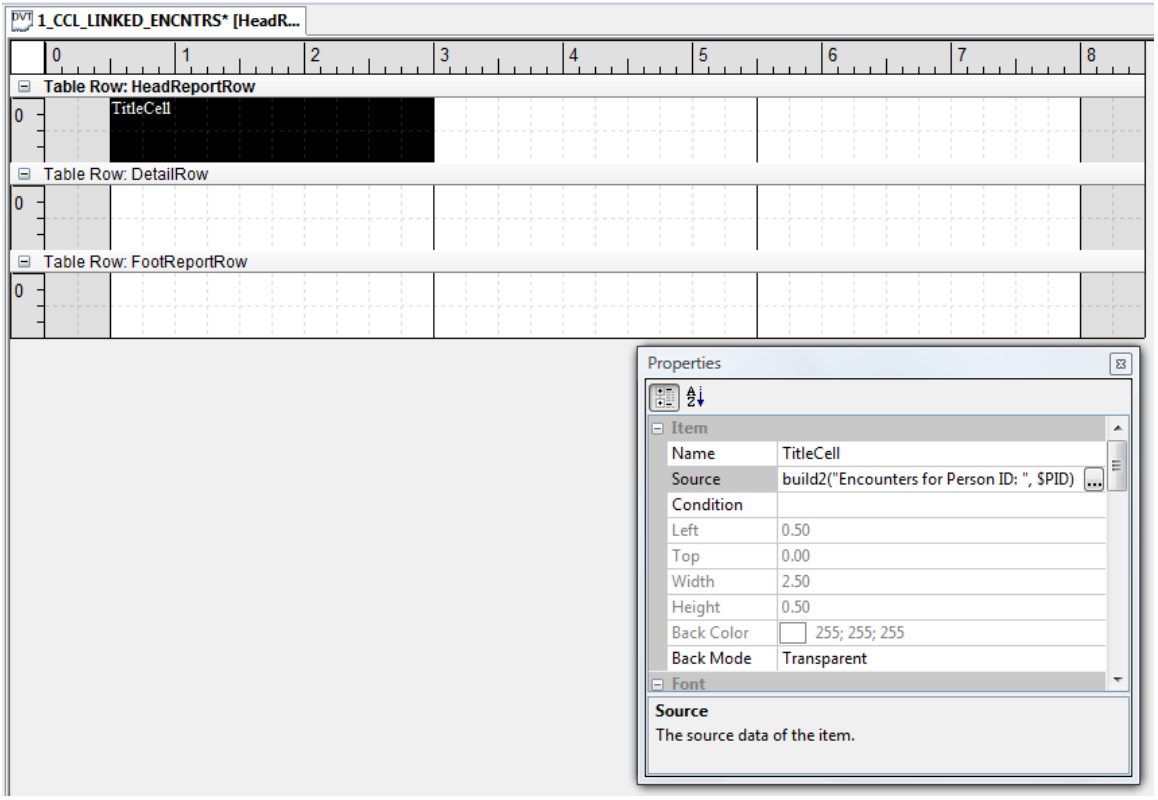
55. Click in the first cell of the HeadReportRow on your table. The Properties dialog box displays the information for the cell you selected.

56. Modify the Name property to **TitleCell**.

57. In the Source property, enter the following command:

```
build2("Encounters for Person ID: ", $PID)
```

The following layout is displayed:



58. Click in the first cell of the DetailRow on your table.

59. In the Properties dialog box, modify the Name to **Encntr_ID**.

60. In the Source box, enter **E.ENCNTR_ID**, or click in the Source box to activate the ellipsis  button to open the Layout Source [Encntr_ID] dialog box and select E.ENCNTR_ID from the Select Fields list in the Variables column.

61. Click in the middle cell of the DetailRow on your table.

62. In the Properties dialog box, modify the Name to **EType_Disp**.

63. In the Source box, enter **E_ENCNTR_TYPE_CLASS_DISP**, or click in the Source property box, then click the ellipsis  button to open the Layout Source [EType_Disp] dialog box and select E_ENCNTR_TYPE_CLASS_DISP from the Select Fields list in the Variables column.

We will use the cell on the right side of the DetailRow on the table to create the link to execute the 1_*Your_Initials*_Linked_Orders program you created earlier. To create the link we will use the special CCLBuildHLink() subroutine. The syntax for this subroutine is:

```
CCLBuildHLink(Program_Name, Prompt_Values, Display_Mode,
Link_Text)
```

Program_Name – Name of program to execute. A literal value must be enclosed in quotation marks. A VC data type variable that is equal to the name of the program to execute can be used without the quotation marks.

Prompt_Values – Prompt values to pass to the program. If the program to be executed is expecting multiple prompt values, the prompt values must be separated by commas. A literal list of comma separated values must be enclosed in quotation marks. The Build(), Build2(), or Concat() function can be used to concatenate literal values and or values contained in variables into a single parameter value.

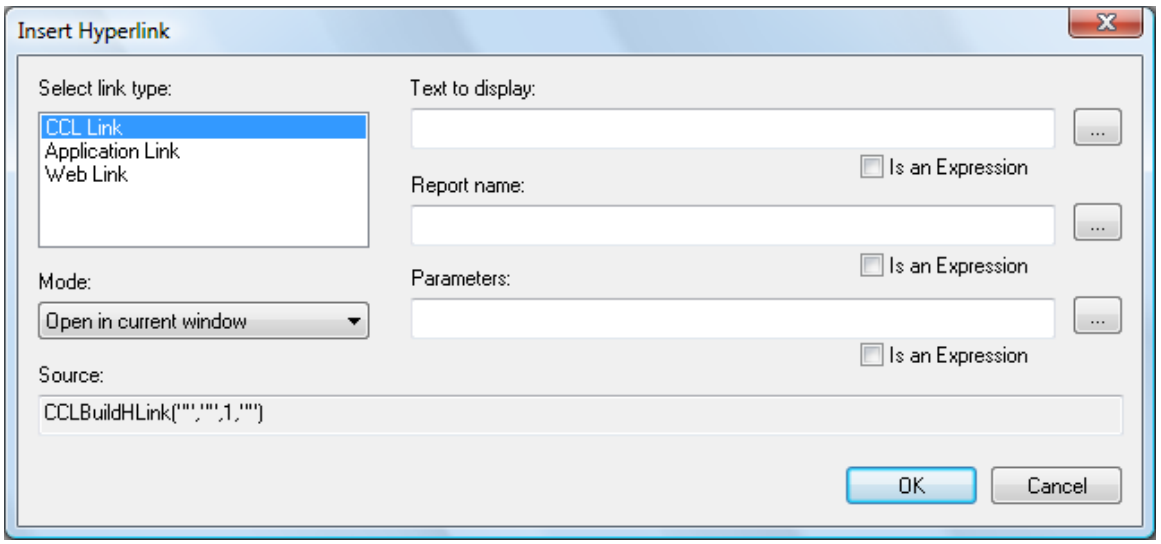
Display_Mode – 0 = open new window, 1 = open in current window.

Link_Text – Text to display in the link. The text can be a literal, a field with a character data type, a character data type variable, or a character data type value that is returned by a function.

64. Click in the cell on the right side of the DetailRow on your table. The Properties dialog box displays the information for the cell you selected.

65. In the Properties dialog box, modify the Name property to **OrderLink**.

66. Change the Is Hyperlink property to **Yes**. The Insert Hyperlink dialog box is displayed.



The Insert Hyperlink dialog box is used to populate the CCLBuildHLink() subroutine. When all of the parameters are filled out, the subroutine will be placed as the value for the Source property.

Note: When creating a hyperlink, if there is text already associated to the source property, you will receive a warning of its deletion.

67. Verify the CCL Link is selected for the Select link type.

68. Click the menu on the Mode option and select Launch DiscernReportViewer.exe.

Note: As the parameters are populated, the CCLBuildHLink() subroutine is filled out and displayed in the Source box.

69. In the Text to display box, enter the following command:

```
concat("Get Orders for Encounter ID: ", cnvtstring(e.encntr_id))
```

To create this command, click the ellipsis  button that is to the right of theText to display box. The Layout Source dialog opens. The Concat() and Cnvtstring() functions can be selected from the CCL Function list in the Functions column. The E.Encntr_id can be selected from the Select Fields list in the Variables column..

70. Check the box for the Is an Expression option. Selecting the option will enable the system to treat the command as an expression instead of a literal value.

Filling out the Text to display box populates the Link_Text parameter of the CCLBuildHLink() subroutine.

71. In the Report Name box, enter the name of the program that you created to find the orders for an encounter:

```
1_ your_initials Linked_Orders
```

The above command assumes that the 1_*your_initials*_linked_orders program was created as a cclgroup0 (DBA) program. If the username that was used to log in to DVDev when the 1_*your_initials*_linked_orders program was created is not in cclgroup0, then by default the program would have been created as a cclgroup1 object. If the 1_*your_initials*_linked_orders program is a cclgroup1 object, you will need to append :group1 to the end of the program name as shown below:

```
1_ your_initials Linked_Orders:group1
```

Use CCLPROT to determine if the 1_*your_initials*_linked_orders program is a cclgroup1 or cclgroup0 program.

72. Verify that the Is an Expression is not checked. This will ensure that the program name is treated as a literal value and quotes will be placed around the value in the subroutine.

Filling out the Report Name box populates the Program_Name parameter of the CCLBuildHLink() subroutine.

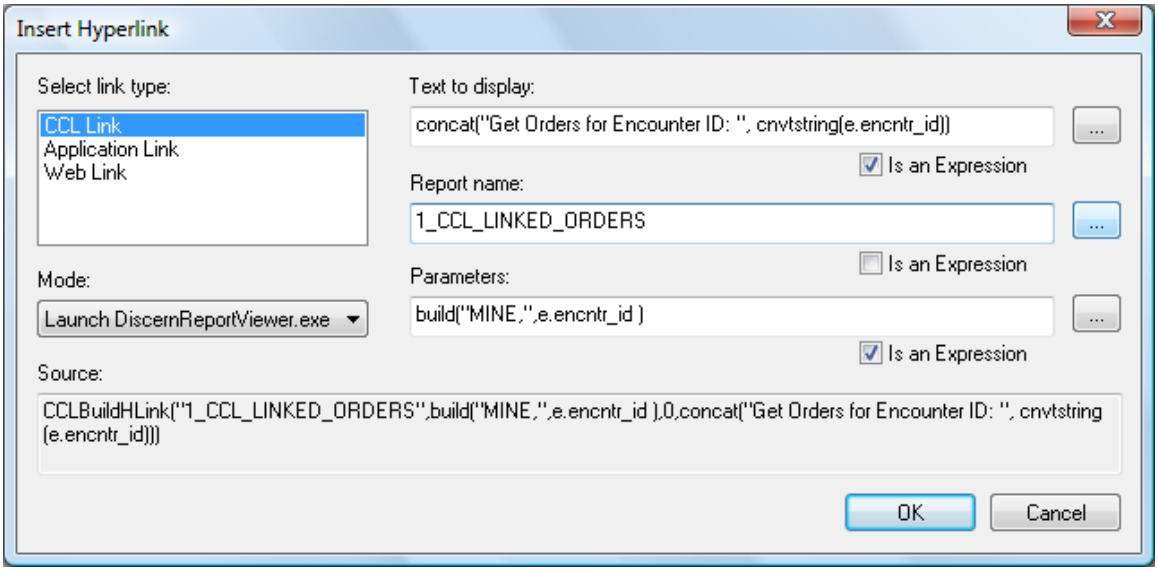
73. In the Parameters box, enter the command as shown below:

```
build("MINE,", e.encntr_id)
```

To create this command, click the ellipsis  button that is to the right of the Parameters: box. The Layout Source dialog opens. The Build( ) function can be selected from the CCL Function list in the Functions column. The E.Encntr_id can be selected from the Select Fields list in the Variables column.

74. Check the box for the Is an Expression option. Selecting the option will enable the system to treat the command as an expression instead of as a literal value.

The Insert Hyperlink dialog box will look similar to the following:



Filling out the Parameters box populates Prompt_Values parameter of the CCLBuildDHLink() subroutine.

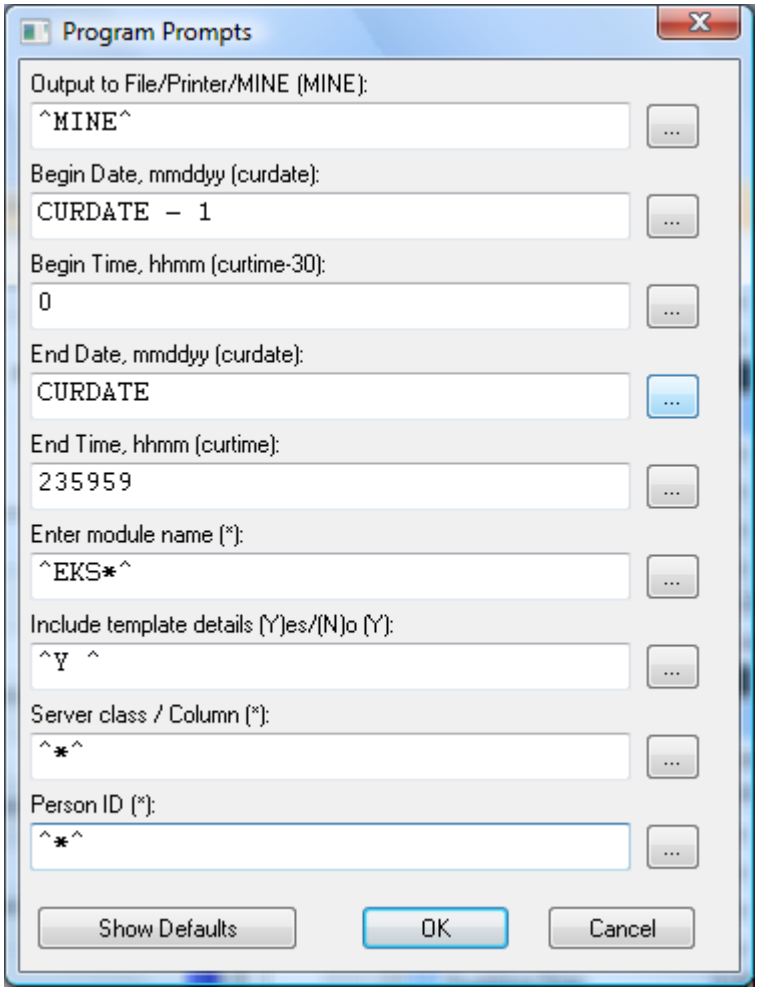
In the above example, the e.encntr_id is referencing a field from the query. In order to render the ENCNTR_ID and to have the actual ENCNTR_ID available to show in the link, the *Is an Expression* option is selected so that the value will be treated as a expression and quotes will not be wrapped around the value. In this example, to render the value, we used the Build() function to manually create the parameter string of comma separated values consisting of the parameter of “MINE” and the actual ENCNTR_ID brought back from the query.

Any time you are referencing a field or expression from the query in the Parameters string, select the *Is an Expression* option and then use the Build(),Build2() or Concat() functions to manually build the parameter string. Using this method tells the system that you have a command that needs to be executed and rendered for the display of the link. Otherwise, the parameter string is not evaluated until the link is selected. When the *Is an Expression* option is selected, you need to know the parameters required to pass to the program.

You do not need to select *the Is an Expression* option if the parameters being passed to the program are global variables, prompts, reserved system variables, or literal values. These expressions do not directly reference an item in the query. When parameters do not contain any reference to an item in the query, the *Is an Expression* option is not needed.

Click the ellipsis button  . A different dialog box, Program Prompts, is displayed and shows the prompts for the program. You do not need to remember all of the prompts required for the program as the Program Prompts dialog box displays them for you. However, the Program Prompts dialog box is only available is when the *Is an Expression* is not selected and the ellipsis button is clicked.

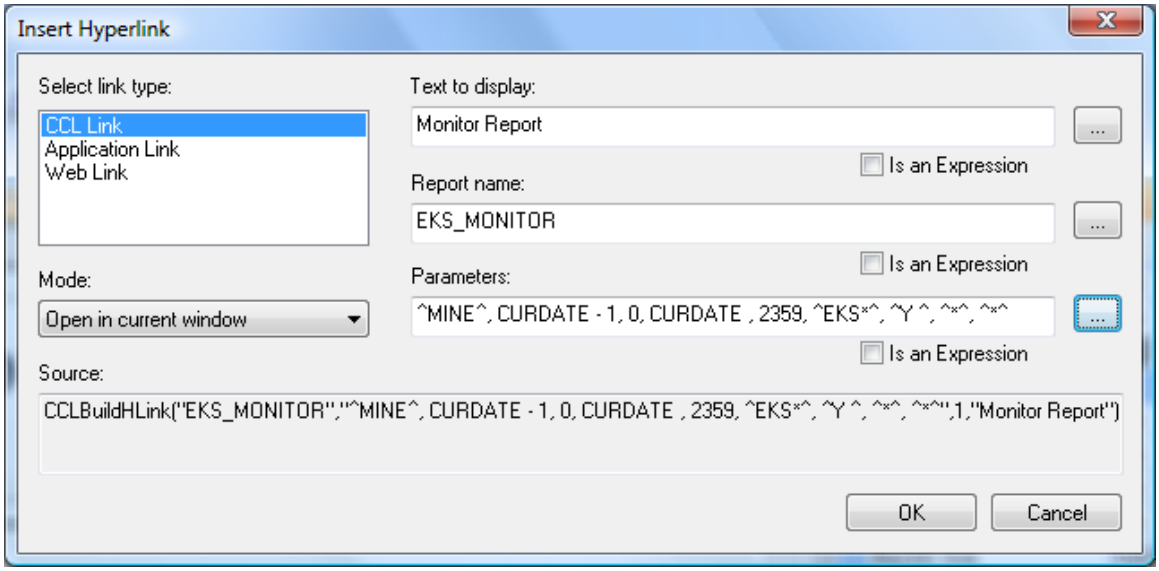
The following is an example of a Program Prompts dialog box when the *Is an Expression* option is not selected and the ellipsis button is clicked.



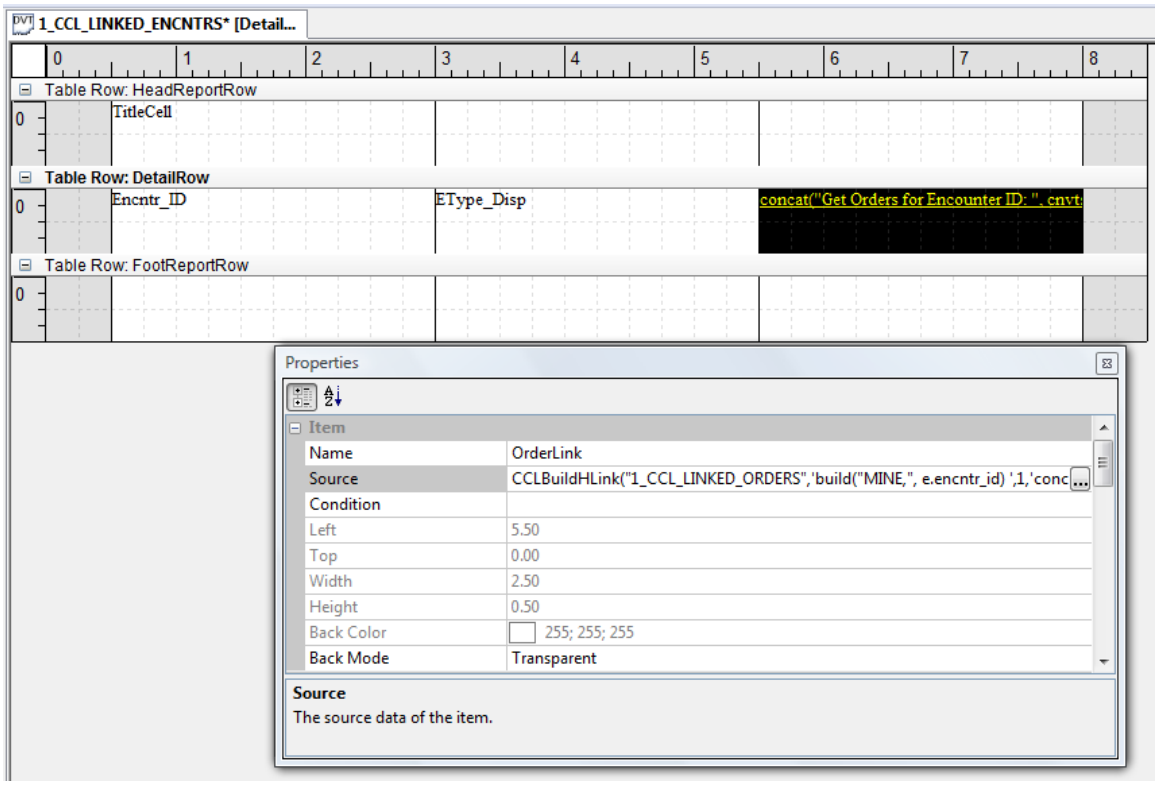
The user can enter any expression in the text boxes or click any of the ellipsis buttons to display the Layout Source dialog box, where the values can be specified. Clicking OK assembles a parameter string of comma separated values and returns you to the Insert Hyperlink dialog.

The following is an example of the Hyperlink dialog box that has the Source populated after closing the Program Prompt dialog box.

Doc ID: A1001001A08G07B34433A94851



75. Click OK to close the Insert Hyperlink dialog box. The following layout is displayed:



The CCLBuildHLink() subroutine is populated and placed as the value for the Source property and should look similar to the following:

```
CCLBuildHLink("1_your_initials_linked_orders",
build("MINE,", e.encntr_id),
0,
concat("Get Orders for Encounter ID: ", cnvtstring(e.encntr_id)))
```

76. From the Build menu, select Run "1_*Your_Initials*_Linked_Encntrs", or press CTRL+F5 to execute your layout program. The Discern Prompt dialog box opens.

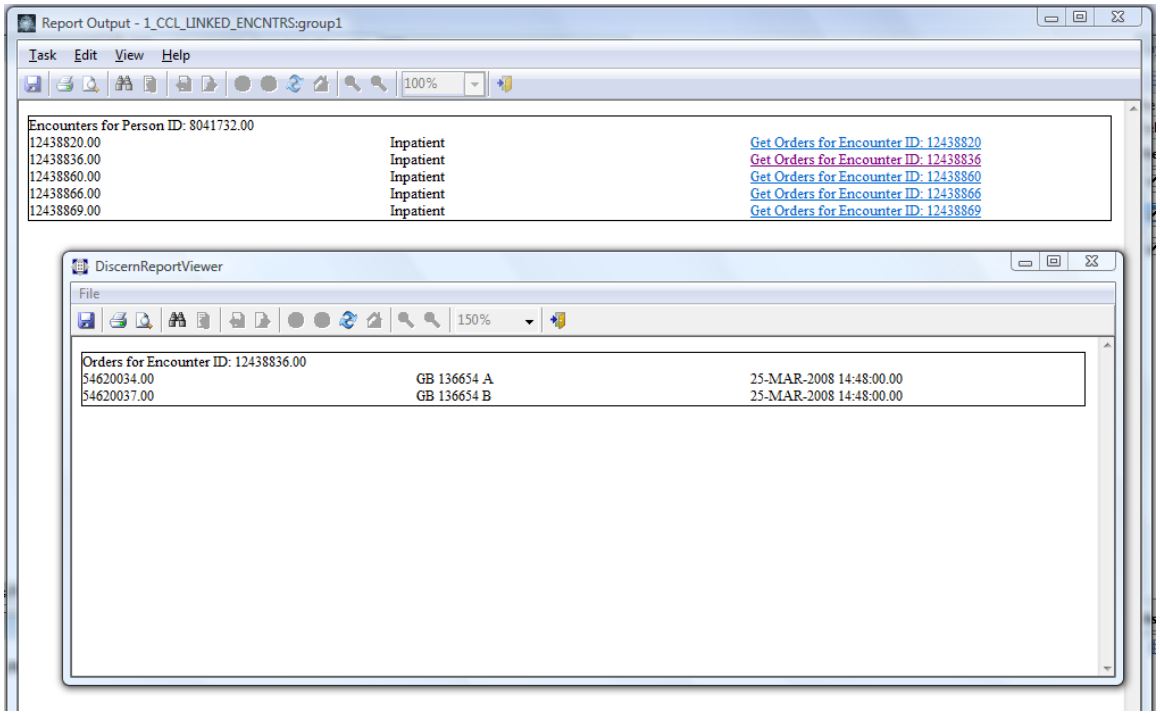
77. Select MINE as the output device.

78. At the Person_ID prompt, enter one of the person_ids you recorded earlier and click Execute. The Report Output window opens and the output of your 1_*Your_Initials*_Linked_Encntrs layout should resemble the following example:

Encounters for Person ID: 589983		
589884.00	Inpatient	Get Orders for Encounter ID: 589884
590069.00	Inbox Message	Get Orders for Encounter ID: 590069
590072.00	Inbox Message	Get Orders for Encounter ID: 590072
590102.00	Outpatient	Get Orders for Encounter ID: 590102

79. Click one of the Get Orders for Encounter ID links on the right side of the output.

    Because the Launch DiscernReportViewer.exe was selected as the Mode, a 0 (zero) was passed in to the Display_Mode parameter of the CCLBuildHLink() subroutine. This causes a new DiscernReportViewer window to be opened and the output of your 1_*your_initials*_Linked_Orders report is displayed. If the DiscernReportViewer window is maximized, right-click the window title bar, and from the context menu, select Restore, and then resize the window so you can see the output from both reports, similar to the following example:



80. Close the DiscernReportViewer and Report Output windows.

We can now create the top-level program of our linked reports. For this level we will create a report that displays information about people with a specific name and provide a link that will pass the person_id and execute the 1_*your_initials*_Linked_Encntrs report you created above. We will create this report as a Table View that prompts for an output device, a first and last name, and displays the information about people with that name in a table.

81. Using DVDev, from the File menu, select New. The New dialog box opens.
82. From the File Type list, select Layout Program.
83. In the Program Name box, enter **1_*your_initials*_Linked_People**, and click OK. The New Layout Program dialog box opens.
84. Select Table View from the Report Layout option. Enter 3 for the Number of Columns option.
85. Select the HTML option for the Output Type. After clicking the Finish button, three rows will be created: HeadReportRow, DetailRow and FootReportRow. Each row will contain three equally spaced cells,
86. From the Tools menu, select Prompt Builder. The Prompt Builder dialog box is displayed with a single control for an output device.
87. Click Add to add a second prompt to get a last name from the user.
88. In the General Tab, set the following:

Prompt Display:	Enter a Last Name
Prompt Name:	LName
Control Type:	Text Edit
Prompt Type:	String

89. On the Text Properties Tab set the Character Case to Upper.
90. Leave the default values for all other properties.
91. Click Add to add a third prompt to get a first name from the user.
92. In the General Tab, set the following:

Prompt Display:	Enter a First Name
Prompt Name:	FName
Control Type:	Text Edit
Prompt Type:	String

93. On the Text Properties Tab set the Character Case to Upper.

94. Leave the defaults for all other properties.

95. Click Save to save the prompt form and close the Prompt Builder.

    The following query selects information about people that have a specific first and last name that is entered at prompts named FName and LName. You created these prompts above.

```
SELECT INTO "NL:"
  P.PERSON_ID,
  P.NAME_FULL_FORMATTED
FROM
  PERSON  P
WHERE  P.NAME_LAST_KEY = $LName and
  P.NAME_FIRST_KEY = $FName
WITH MAXREC = 100, NOCOUNTER, SEPARATOR=" ", FORMAT
```

96. Copy this query to the clipboard.

97. From the Tools menu, select Query Builder.

98. To add a new query, click Add in the Add Queries dialog box.

99. In the Add Query dialog box, enter **Get_Data** in the Query Name box.

100. Select the Associate Layout and Insert Query From Clipboard options and click OK to add the query. The Discern Query Builder dialog box opens.

101. Click Close to close the Discern Query Builder.

102. Click OK to close the Add Queries dialog box.

103. Click in the center cell of the HeadReportRow on your table. The properties for that cell are displayed in the Properties dialog box.

104. Modify the Name property to **TitleCell**.

105. In the Source property, enter **"Linked Person Encounters Orders Report"** for this cell.

106. In the Align Horizontal property, select Center.

107. Click in the first cell of the DetailRow on your table, and in the Properties dialog box, modify the Name property to **Person_ID**.

108. In the Source box, enter P.PERSON_ID, or click the ellipsis  button to open the Layout Source [Person_ID] dialog box and select P.PERSON_ID from the Select Fields list in the Variables column.

109. Click in the middle cell in the second row of your table, and in the Properties dialog box, modify the Name property to **Formatted_Name**.

110. In the Source property, enter **P.NAME_FULL_FORMATTED**, or click the ellipsis  button to open the Layout Source [Formatted_Name] dialog box and select P.NAME_FULL_FORMATTED from the Select Fields list in the Variables column.

    We will use the cell on the right side of the DetailRow on the table to create the link to execute the 1_*your_initials*_Linked_Encntrs program you previously created using the special CCLBuildHLink() subroutine.

111. Click in the cell on the right side of the DetailRow on your table. The Properties for the cell are displayed.

112. In the Properties dialog box, modify the Name property to **EncntrLink**.

113. Change the Is Hyperlink property to Yes. The Insert Hyperlink dialog box is displayed.

114. Verify the CCL Link is selected for the Select link type.

115. Click the menu on the Mode option and select Launch DiscernReportViewer.exe.

116. In the Text to display box, enter the following command:

```
concat("Get Encounters for ", p.name_full_formatted)
```

    To create this command, click the ellipsis  button that is to the right of theText to display box. The Layout Source dialog opens. The Concat( ) function can be selected from the CCL Function list in the Functions column. The P.Name_full_formatted can be selected from the Select Fields list in the Variables column.

117. Check the box for the Is Expression option. Selecting the option will enable the system to treat the command as an expression instead of a literal value.

118. In the Report Name box, enter the name of the program that you created to find the encounters for a person as shown below:

```
1_ your_initials Linked_Encntrs
```

Note: The above command assumes that the 1_*your_initials*_linked_encntrs program was created as a cclgroup0 (DBA) program. If the username that was used to log in to DVDev when the 1_*your_initials*_linked_encntrs program was created is not in cclgroup0, then by default the program would have been created as a cclgroup1 object. If the 1_*your_initials*_linked_encntrs program is a cclgroup1 object, you will need to append ":group1" to the end of the program name as shown below:

```
1_ your_initials Linked_Encntrs:group1
```

    Use CCLPROT to determine if the 1_*your_initials*_linked_encntrs program is a cclgroup1 or cclgroup0 program.

119. Verify that the Is an Expression is not checked. This will ensure that the program name is treated as a literal value and quotes will be placed around the value in the subroutine.

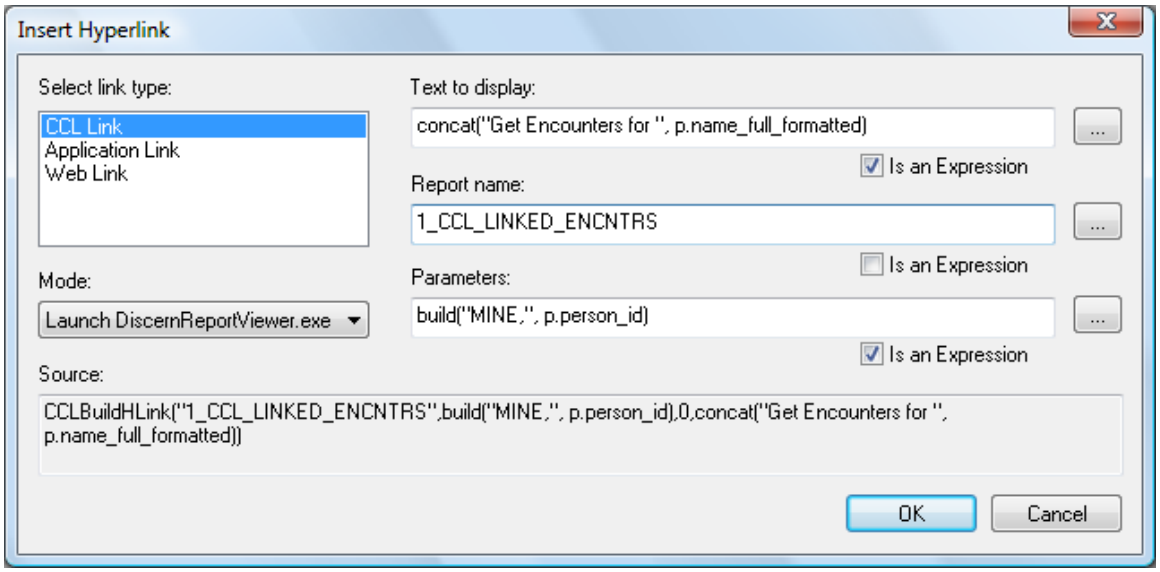
120. In the Parameters box, enter the command as shown below:

```
build("MINE,", p.person_id)
```

To create this command, click the ellipsis ... button that is to the right of the Parameters box. The Layout Source dialog opens. The Build( ) functions can be selected from the CCL Function list in the Functions column. The P.Person_id can be select from the Select Fields list in the Variables column.

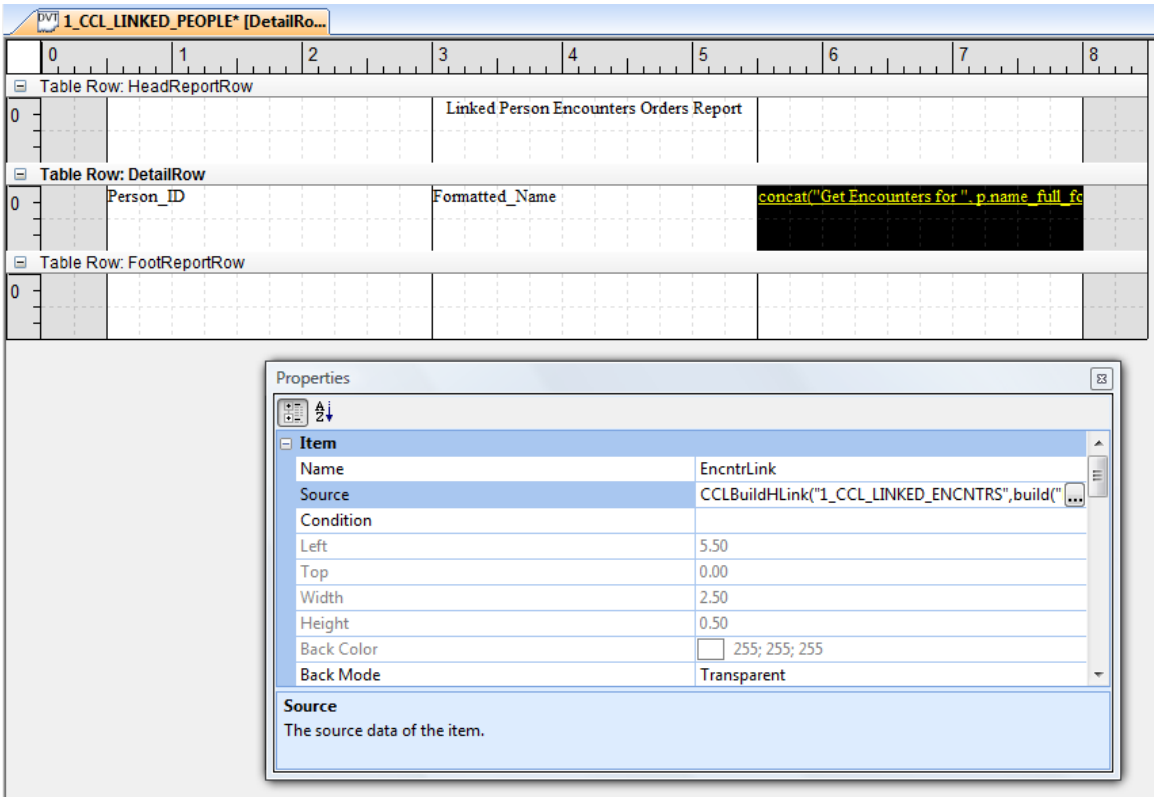
121. Check the box for the Is an Expression option. Selecting the option will enable the system to treat the command as an expression instead of a literal value.

The Insert Hyperlink dialog box should look similar to the following:



122. Click OK to close the Insert Hyperlink dialog box. The following layout is displayed:

Doc ID: A1001001A08G07B34433A94851



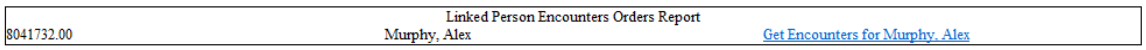
The CCLBuildHLink() subroutine is populated and placed as the value for the Source property and should look similar to the following:

```
CCLBuildHLink("1_your_initials_linked_encntrs",
build("MINE,", p.person_id),
0,
concat("Get Encounters for ", p.name_full_formatted))
```

123. From the Build menu, select Run "1_ Your_Initials_Linked_People" or press CTRL+F5 to execute your layout program. The Discern Prompt dialog box opens.

124. Select MINE as the output device.

125. Enter one of the last names that you recorded earlier at the last name prompt. Enter one of the first names that you recorded earlier at the first name prompt. The Report Output window opens and the output of your 1_your_initials_Linked_People layout should resemble the following example:

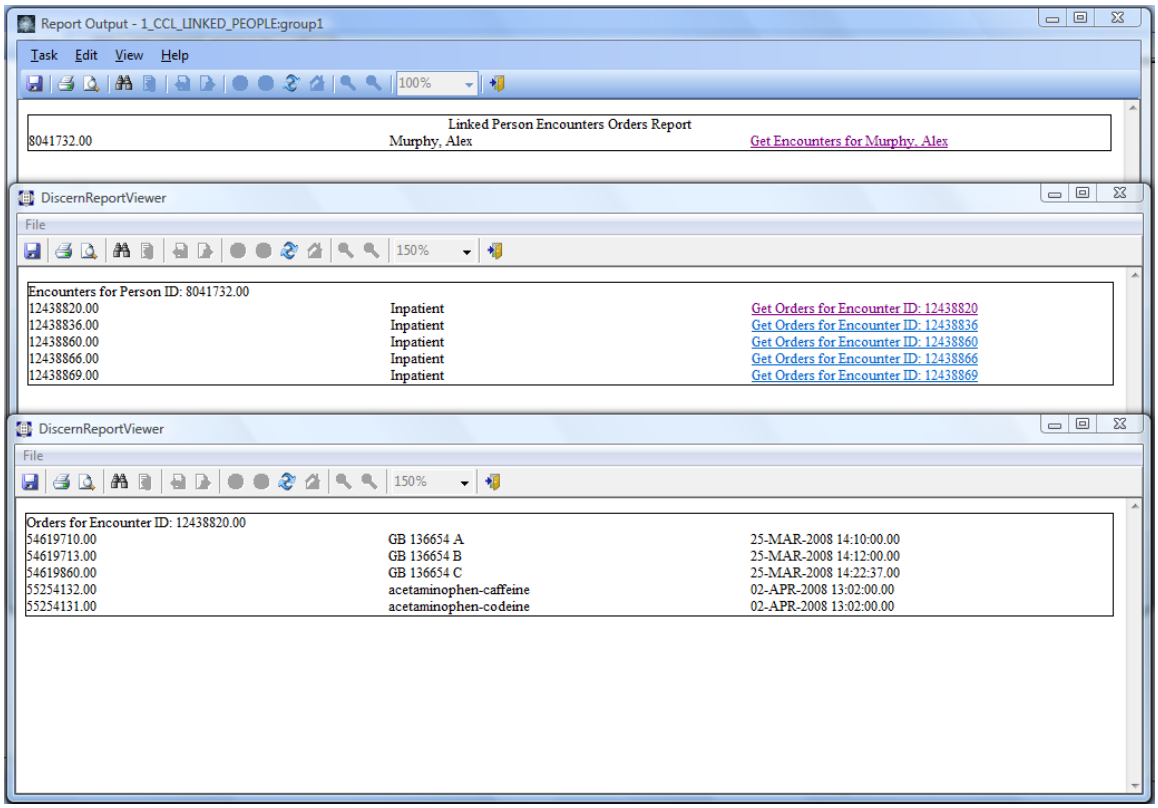


126. Click the Get Encounters for *Name_Full_Formatted* link on the right side of the output.

Because the Launch DiscernReportViewer.exe was selected as the Mode, a 0 (zero) was passed in to the Display_Mode parameter of the CCLBuildHLink() subroutine. This causes a new DiscernReportViewer window to be opened and the output of your 1_*your_initials*_Linked_Encntrs report is displayed. If the DiscernReportViewer window is maximized, resize it so you can see the output from both reports.

127. Click one of the Get Orders for Encounter ID: Encntr_ID links on the right side of the output.

    Because the Launch DiscernReportViewer.exe was selected as the Mode, a 0 (zero) was passed in to the Display_Mode parameter of the CCLBuildHLink() subroutine. This causes a new DiscernReportViewer window to be opened and the output of your 1_*your_initials*_Linked_Orders report is displayed. If the DiscernReportViewer window is maximized, resize it so you can see the output from all three reports. The following Report Output windows are displayed:



128. Close the DiscernReportViewer and the Report Output windows.

Converting Existing Programs to Layout Programs Using a Driver Program

A driver program collects data and loads it into a record structure. A layout program can then reference that record structure. There are several common uses of driver programs. For example, you may have a situation where the same query is used to retrieve data that is displayed in several different formats. While you could duplicate the query in a layout

program for each output format, doing so would require you to modify each program if you needed to modify the query. To prevent this type of rework, you could place the query in a driver program and create a layout program for each format. Each layout program would use the driver program to retrieve the data. Then, if you needed to modify the query, you would only have to modify the driver program. Another common use of a driver program is to convert an existing program to use the Discern Layout Builder functionality. If the program is a simple one, it is most easily re-created as a layout program. The Insert From Clipboard option on the Add Query dialog box makes it easy to import an existing query, and the Edit Code button on the Add Queries dialog box can be used to re-create the code in the Reportwriter sections. However, if the existing program is large and complicated, it might be easier to convert it to a driver program that loads the data into a record structure, and then create a layout program that references that record structure. The following source code example creates a program that displays the number of orders placed each day for a date range that is entered by the user at run time.

```
drop program ccl_get_order_count go
create program ccl_get_order_count

prompt
"Output to File/Printer/MINE" = "MINE"
, "Enter the Start Date" = " "
, "Enter the End Date" = " "

with OUTDEV, Sdate, Edate
SELECT INTO $outdev
  odate = format(O.ORIG_ORDER_DT_TM, "yyyy-mm-dd")
FROM ORDERS  O
WHERE  O.ORIG_ORDER_DT_TM between
   cnvtdatetime($sdate) and cnvtdatetime($edate)
ORDER BY odate
head report
  col 0 "Total Orders for:"
  row +1
head odate
  col 5 odate
foot odate
  col +1 count(odate)
  row +1
WITH  NOCOUNTER, SEPARATOR=" ", FORMAT
end
go
```

The output of this simple program would look something like the following example:

```
Total Orders for:
     2007-05-30        400
     2007-05-31        279
     2007-06-01        111
     2007-06-03         21
     2007-06-04        323
     2007-06-05        298
     2007-06-06        272
```

If you want the data displayed in a graph instead of listed as numerical values, you can use the graphing functionality in Discern Layout Builder. Considering the simplicity of this example program, it would be very easy to create a layout program that selects the data, generates the counts, and displays the data as a graph. However, re-creating larger and more complex programs is not as easy. In those cases, it would be easier to convert the

existing program into a driver program and then call the driver program from a layout program. To demonstrate the process, we will convert the example program above to a driver program and then call it from a layout program.

Creating a Driver Program

A driver program takes the selected data and loads it into a record structure. Any tool can be used to create a driver program. Before we can go any further, the program needs to be created. Use the following steps to create the program in your environment.

1. Using DVDev, from the File menu, select New. The New dialog box opens.
2. From the File Type List, select Prompt Program.
3. In the Program Name box, enter **1_*your_initials*_Example_Driver** and click OK. The Discern Prompt Builder dialog box opens, with a single control for an output device.
4. Click Add to add the second prompt that asks the user for a start date.
5. On the General Tab, set the following:

Prompt Display:	Select the Beginning Date
Prompt Name:	Sdate
Control Type:	Date Time
Prompt Type:	String

6. On the Date/Time tab, verify the Date Only option is selected and the Command Line Format is DD-MMM-YYYY.
7. In the Anchor date & time area on the Calculate Default tab, uncheck the Current date & time option. Enter a negative value in the Day or Month control, to make the default start date far enough in the past to allow the query to get orders that span several different days.

The values you enter will depend on the data that exists in your environment. If you have a lot of data you can set the Day field to -7 to make the default start date one week ago. If you do not have a lot of orders, you could set the Month field to -3 to make the default start date three months ago.

8. Click Add to add a third prompt that asks the user for an end date.
9. On the General Tab, set the following:

Prompt Display:	Select the Ending Date
Prompt Name:	Edate
Control Type:	Date Time

Prompt Type: String

10. On the Date/Time tab, select the Date and Time option and set the Command Line Format to DD-MMM-YYYY HH:MM:SS.

11. Click Save to save the prompt form and close Discern Prompt Builder.

12. Copy the following select statement into your DVDev file. Place this code below the prompts and before the key words End Go.

```
SELECT INTO $outdev
 odate = format(O.ORIG_ORDER_DT_TM, "yyyy-mm-dd")
FROM ORDERS  O
WHERE  O.ORIG_ORDER_DT_TM between
 cnvtdatetime($sdate) and cnvtdatetime($edate)
ORDER BY odate
head report
 col 0 "Total Orders for:"
 row +1
head odate
 col 5 odate
foot odate
 col +1 count(odate)
 row +1
WITH  NOCOUNTER, SEPARATOR=" ", FORMAT
```

13. Save and compile the source code file.

14. Execute the prompt program and verify it returns a count of the orders placed over several days.

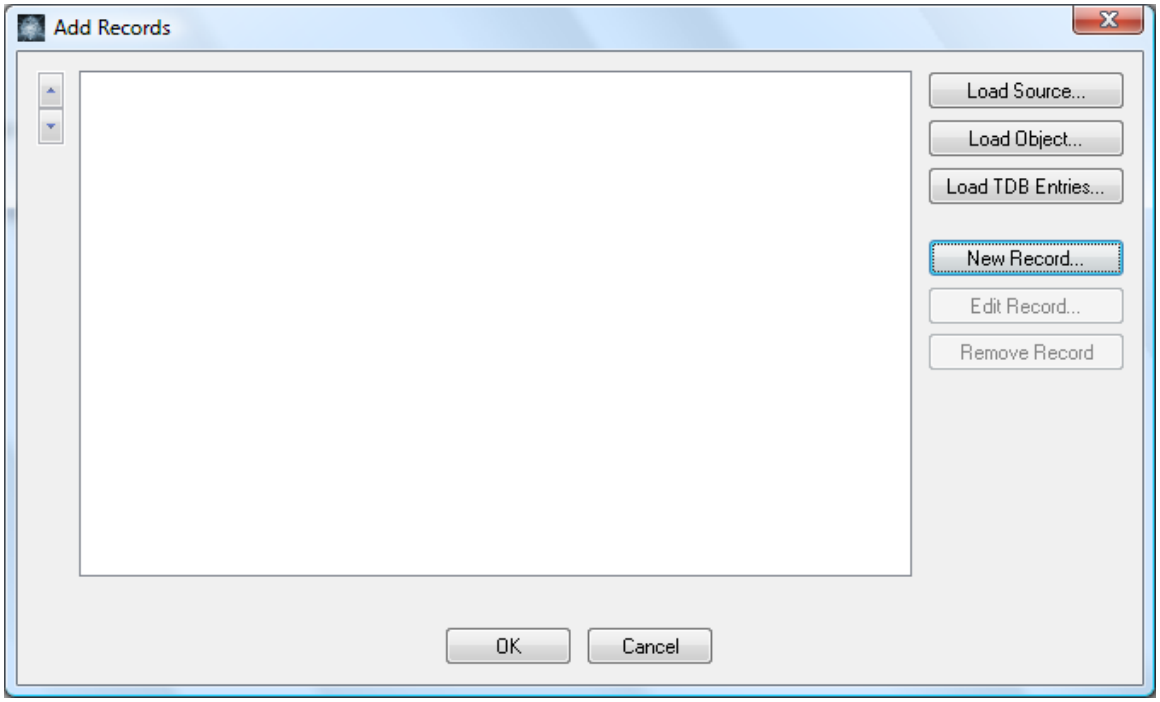
You have now completed the steps necessary to create the prompt program to display a count of orders placed on a given day. We will assume this program was written in the past and we now want to use the functionality of the layout builder to add a graphical view of the data. To accomplish that, we will convert the program you just created to a driver program. Again, if the program does not already exist, it will be easier to create it as a layout program than to create a driver program and then call it from a layout program. The existing program displays the results of the select command using Reportwriter sections. We want to convert the existing program to a driver program that will load the results into a record structure instead of displaying them as output.

Create the Record Structure

1. In your 1_*your_initials*_Example_Driver.prg file, place the cursor after the Prompt clause and before the Select command.

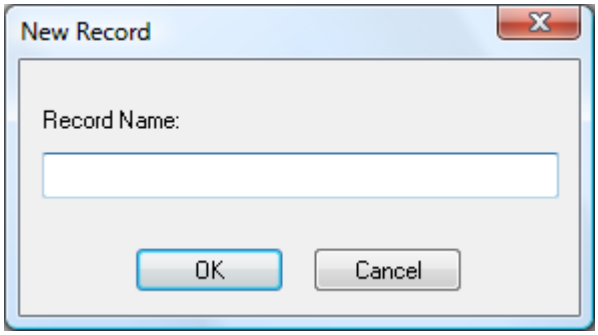
2. From the Tools menu, select Record Builder.

3. Select Yes at the prompt to add a new record. The Add Records dialog box opens.



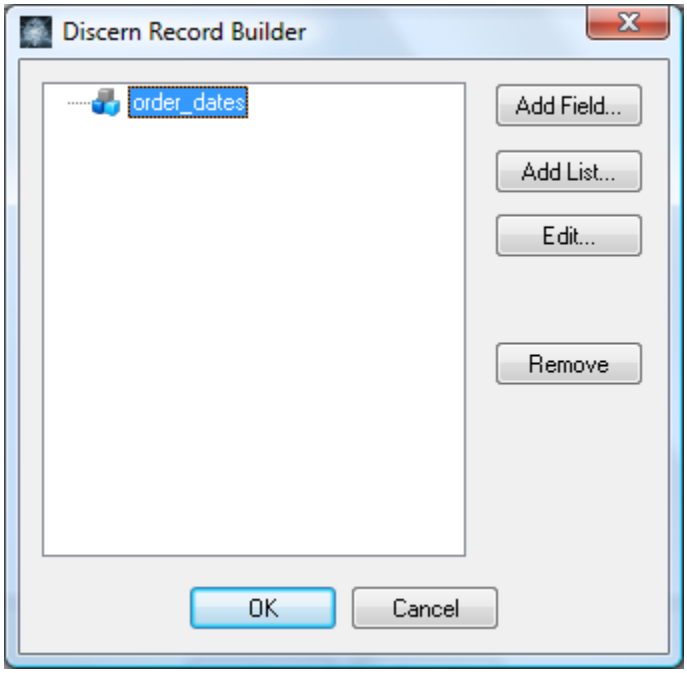
The Add Records dialog box can be used to create new record structure definitions or to pull a record structure definition from an existing source code file, program object, or TDB entry into your source code file.

4. Click New Record. The New Record dialog box opens.

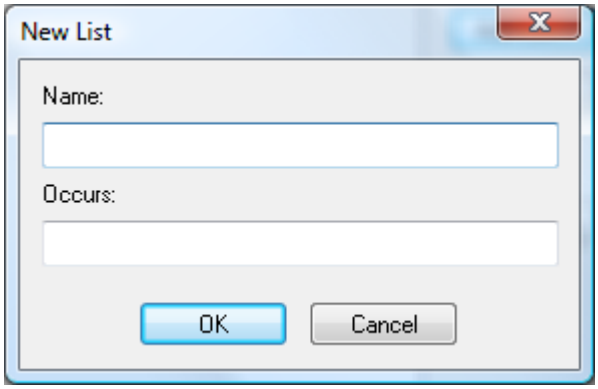


5. Enter **order_dates** as the Record Name and click OK. The Discern Record Builder dialog box is displayed.

Doc ID: A1001001A08G07B34433A94851



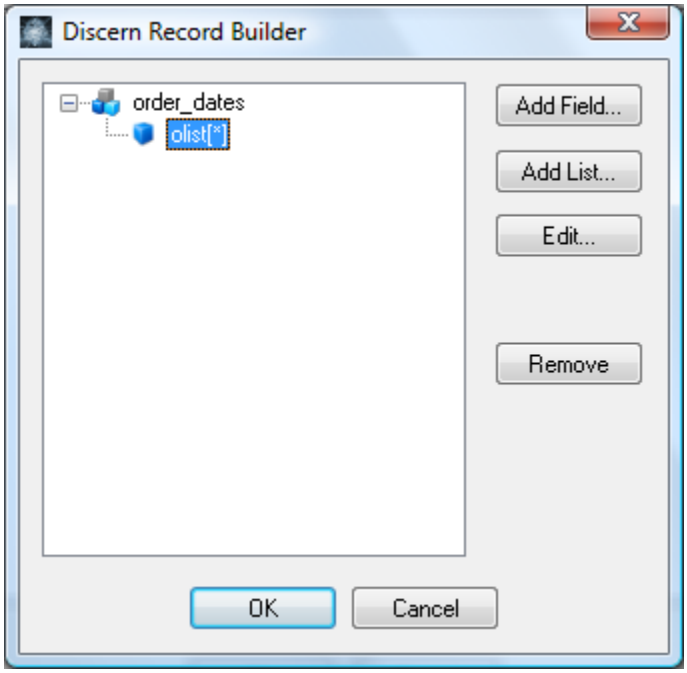
6. Click Add List. The New List dialog box opens.



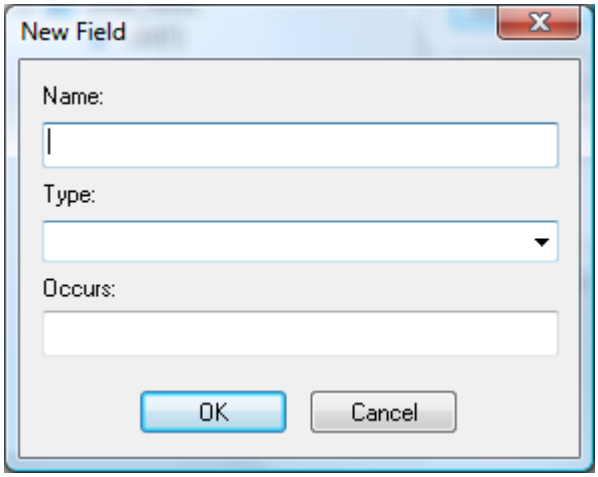
7. In the Name box, enter **olist** and in the Occurs box, enter *****.

8. Click OK to close the Add List dialog box.

9. From the tree, select olist[*]. Your screen in Discern Record Builder should resemble the following:



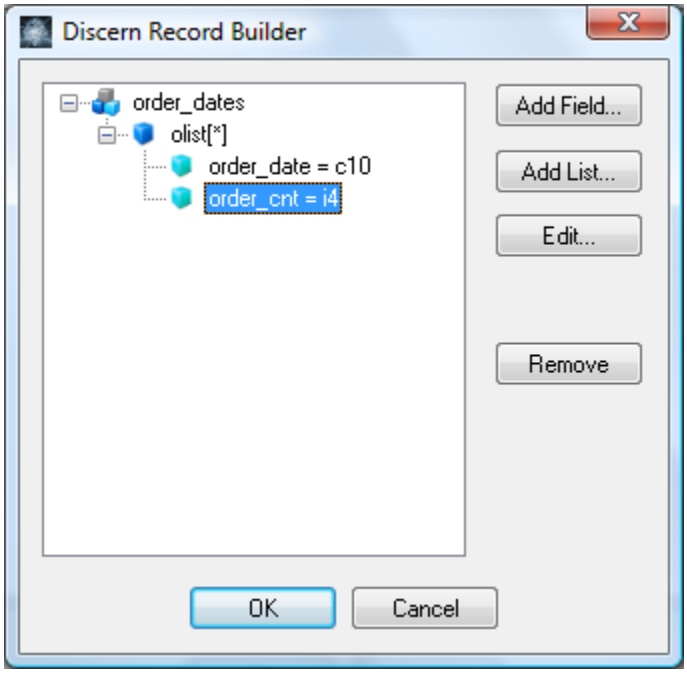
10. Click Add Field. The New Field dialog box opens.



11. In the Name box, enter **order_date**.
12. In the Type box, enter **C10**.
13. Click OK to close the New Field dialog box.
14. Click Add Field to add a second field to the olist list.
15. In the Name box, enter **order_cnt**.
16. In the Type box, enter **i4**.

Doc ID: A1001001A08G07B34433A94851

17. Click OK to close the New Field dialog box. Your Discern Record Builder dialog box is displayed similar to the following example:



18. Click OK to close the Discern Record Builder dialog box.

19. Click OK to close the Add Records dialog box. The Record Builder added code similar to the following to your source code file:

```
record order_dates (
 1 olist [*]
  2 order_date = c10
  2 order_cnt = i4
)
```

Create Needed Variables

In addition to the record structure you also need a counter variable to use for referencing the position of the olist in your order_dates record structure.

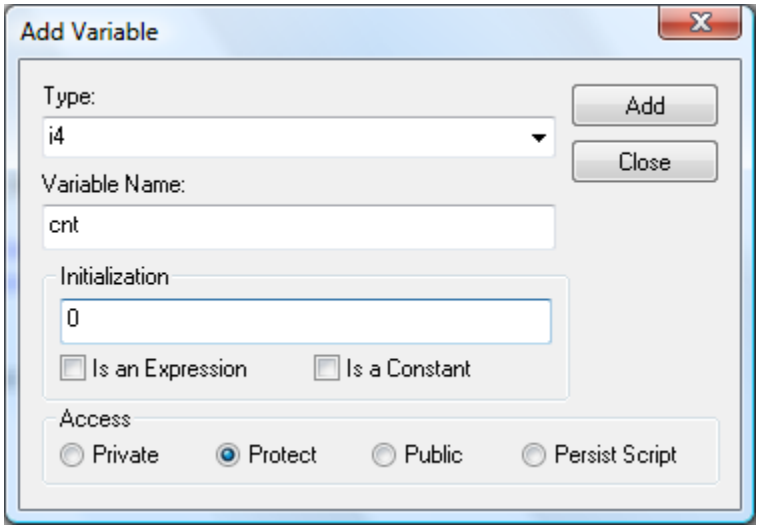
1. From the Tools menu, select Add Variable. The Add Variable dialog box opens.

2. From the Type list, select i4.

3. In the Variable Name box, enter **cnt**.

4. In the Initialization box, enter **0** (zero). Your Add Variable dialog is displayed similar to the following example:

Doc ID: A1001001A08G07B34433A94851



5. Click Add to add the cnt variable.

6. Click Close to close the Add Variable dialog box. The following declare command is added to your source code:

```
declare cnt = i4 with Protect
```

Note: You have the option of typing Declare and Record commands directly in your source code file instead of using the Record Builder and Add Variable tools. However, using the tools eliminates the need to remember or look up the exact syntax for these commands.

Store the Data in the Record Structure

We now need to modify the Reportwriter sections of the existing Select command to load the data into the record structure instead of displaying it.

1. Select the following code in your 1_*your_initials*_Example_Driver.prg file:

```
head report
 col 0 "Total Orders for:"
 row +1
head odate
 col 5 odate
foot odate
 col +1 count(odate)
 row +1
```

2. From the Edit menu, select Extras > Comment Block to comment out the selected code.

3. Place the following code directly below the code you commented out in the preceding step.

```
foot odate
 ocnt = count(odate)
 cnt = cnt +1
 if(mod(cnt,10) = 1)
  stat = alterlist(order_dates->olist, cnt +9)
 endif
 order_dates->olist[cnt].order_date = odate
 order_dates->olist[cnt].order_cnt  = ocnt
foot report
 stat = alterlist(order_dates->olist, cnt)
```

Your complete 1_*your_initials*_Example_Driver.prg file now contains source code that is very similar to the following example:

```
drop program 1_CCL_Example_Driver go
create program 1_CCL_Example_Driver
prompt
"Output to File/Printer/MINE" = "MINE"
, "Select the Beginning Date" = " "
, "Select the Ending Date" = " "

with OUTDEV, Sdate, Edate
record order_dates (
 1 olist [*]
  2 order_date = c10
  2 order_cnt = i4
)

declare cnt = i4 with Protect
SELECT INTO $outdev
 odate = format(O.ORIG_ORDER_DT_TM, "yyyy-mm-dd")
FROM ORDERS  O
WHERE  O.ORIG_ORDER_DT_TM between
   cnvtdatetime($sdate) and cnvtdatetime($edate)
ORDER BY odate
;head report
;  col 0 "Total Orders for:"
;  row +1
;head odate
;  col 5 odate
;foot odate
;  col +1 count(odate)
;  row +1
foot odate
 ocnt = count(odate)
 cnt = cnt +1
 if(mod(cnt,10) = 1)
  stat = alterlist(order_dates->olist, cnt +9)
 endif
 order_dates->olist[cnt].order_date = odate
 order_dates->olist[cnt].order_cnt  = ocnt
foot report
 stat = alterlist(order_dates->olist, cnt)
WITH  NOCOUNTER, SEPARATOR=" ", FORMAT
end
go
```

Since the driver program is loading data into a record structure instead of returning data for display, the prompt for an output device and selecting into that device is no longer needed. Therefore the output device prompt could be deleted and the Select Into $outdev could be modified to Select Into "NL:". Another option is to leave the prompt for the output device and the Select Into $outdev as is, and pass "NL:" as the first parameter when the driver program is called. For now we will use the second option. A start date in

the format DD-MMM-YYYY and an end date in the format DD-MMM-YYY HH:MM:SS also need to be passed as parameters when the driver program is called.

4. Save and compile the source code file.

Calling a Driver Program from a Layout Program

Once a driver program that loads data into a record structure has been created, it can be called from a layout program using Set Layout Driver from the Tools menu in the Layout Builder. In addition to setting the layout driver, the layout program needs to; prompt the user for any input that is needed at run-time, define the record structure, access the data that is in the record, and display the data in the appropriate format.

Creating a New Layout Program To Use the Driver Program

1. Using DVDev, from the File menu, select New.

2. From the File Type list, select Layout Program.

3. In the Program Name box, enter **1_*your_initials*_Layout_Driver** and click OK. The New Layout Program dialog box opens.

4. Verify the Standard Layout option for the Report Layout and the PostScript option for the Output Type are selected and click Next.

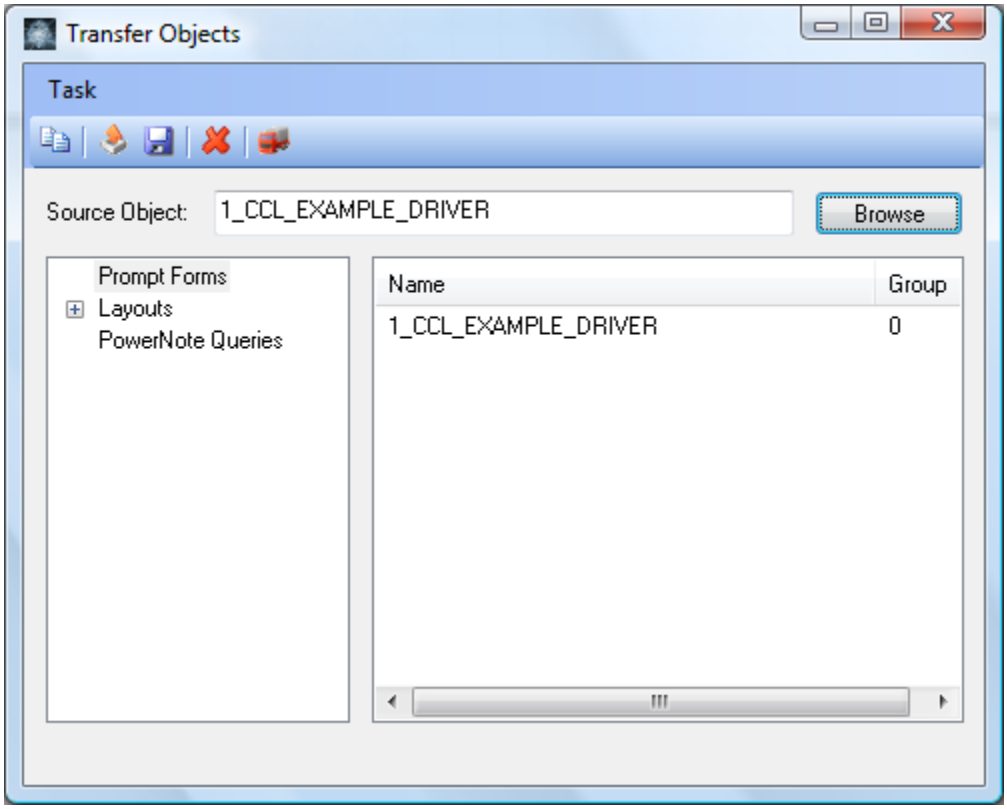
5. Keep the defaulted values for the Paper Size and click Finish. A layout with a single section named DetailSection is created.

Prompt for User Input

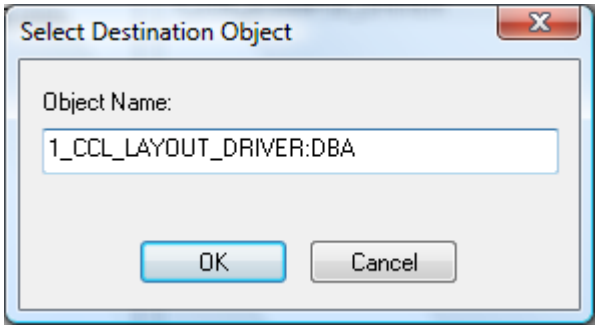
The layout program needs to prompt the user for any input that is required at run time. Since the driver program we are planning to use is expecting a start date and an end date, our layout program needs to prompt for these dates. The dates the user selects at the prompts will be passed to the driver program when it is called. A prompt form can be created for the layout program just like it can be created for any other program using the prompt builder. However, because the program we converted to a driver program already has a prompt form, we can copy that prompt form for use by the layout program.

1. From the Tools menu, select Transfer Objects. The Transfer Objects dialog box opens.

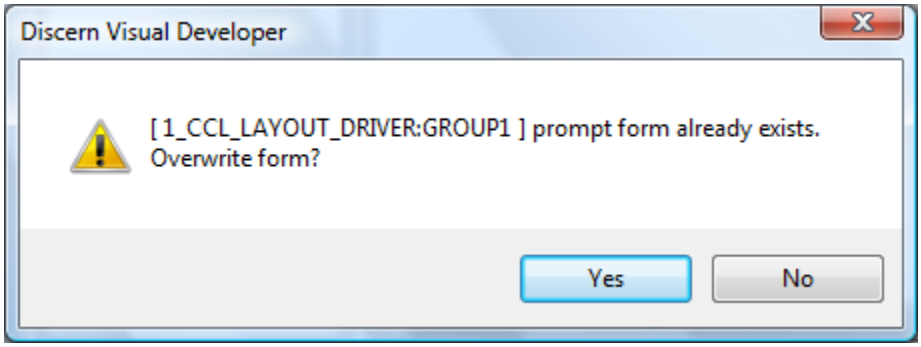
2. In the Source Object box, enter **1_*your_initials*_Example_Driver** and click Browse. Your prompt form opens in the search result box.



3. Right-click your prompt form and select Copy Object. The Select Destination Object dialog box opens.

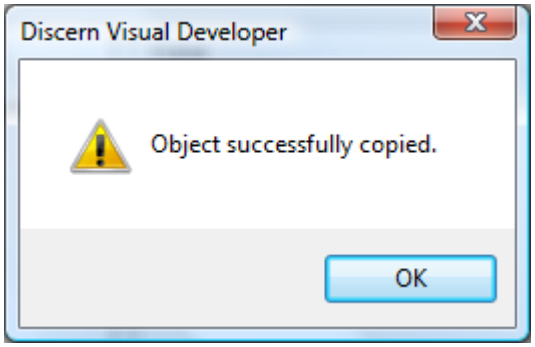


4. In the Object Name box, enter **1_*your_initials*_Layout_Driver** and click OK. A warning message similar to the following example is displayed:



When you created the layout program earlier in this exercise, a default prompt form with an output device control was created for your program.

5. Click Yes to overwrite the default form with the new copy. A message similar to the following example is displayed:



6. Click OK and close the Transfer Objects dialog box.

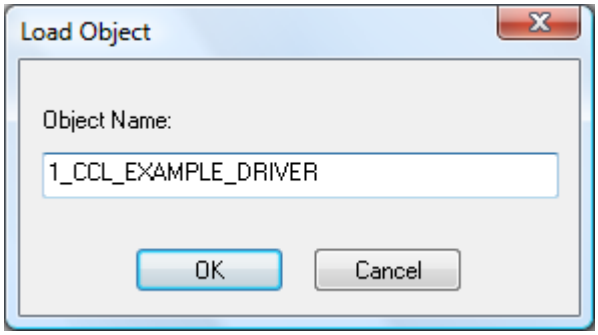
Define the Record Structure

The layout program executes the driver program to load data into a record structure. In order for the layout program to format the data from the record structure, the record structure needs to be known in the layout program and in the driver program. Since the layout program will call the driver program, the easiest way to have the record structure known in both the layout and driver programs is to have the layout program create the record structure.

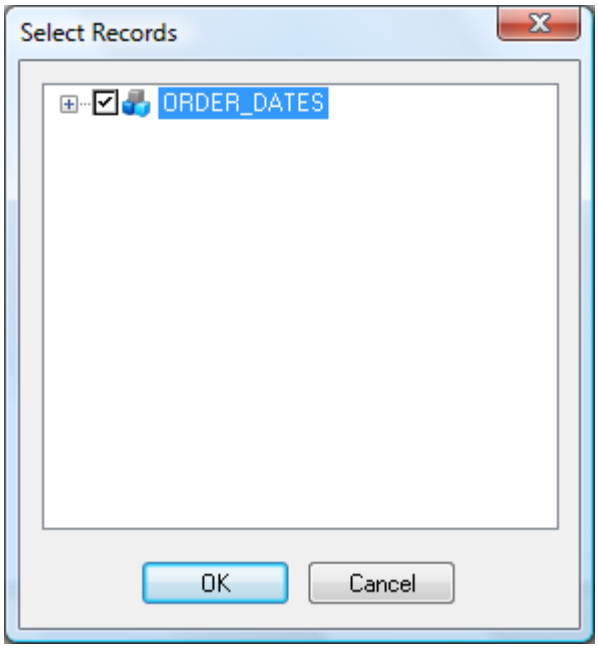
1. From the Tools menu, select Record Builder. The Add Records dialog box opens. You used the Add Records dialog box previously to create the record structure in the driver program. Since the record structure definition already exists in the driver program, you can use Load Object to have the Record Builder pull the definition from your 1_*your_initials*_Example_Driver program into your 1_*your_initials*_Layout_Driver program.

2. Click Load Object. The Load Object dialog box opens.

Doc ID: A1001001A08G07B34433A94851



3. In the Object Name box, enter **1_*your_initials*_Example_Driver** and click OK. The Select Records dialog box opens. Each record structure defined in the program object will be displayed in the Select Records dialog box. Since your 1_*your_initials*_Example_Driver program only has one record structure definition, your Select Records dialog box will resemble the following example:



4. Select Order_Dates and click OK. Your selection is displayed in the Add Records dialog box.

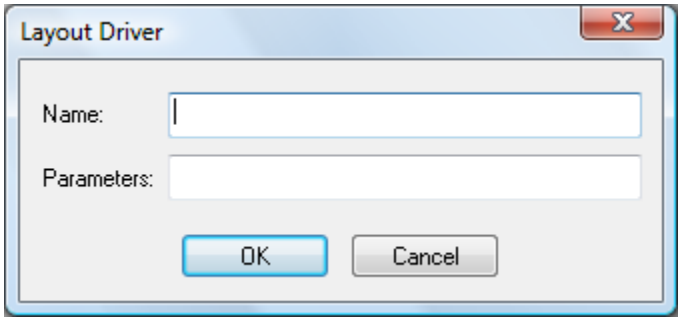
5. Click OK to close the Add Records dialog box.

Using the Add Records dialog box to load the Order_Dates record from your 1_*your_initials*_Example_Driver program creates a record command in the layout code. At this point you have the record command to create the Order_Dates record in both the 1_*your_initials*_Example_Driver and the 1_*your_initials*_Layout_Driver programs. Since the layout program calls the driver program, it is not necessary to define the record structure in the driver program; by default it will be known in the driver. Having the record structure definition in both the driver program and the layout program will result in a warning that the record already exists when the driver program is executed. If you want to prevent this warning you can delete the record definition from the driver program.

Calling the Driver Program

Once a driver program has been created, it can be called or executed from a layout program by selecting Set Layout Driver from the Tools menu. Setting a layout driver will cause the layout program to execute the driver program.

1. From the Tools menu, select Set Layout Driver. The Layout Driver dialog box opens.

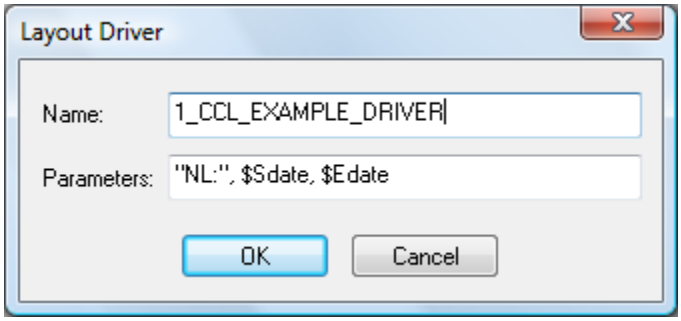


The Layout Driver dialog box enables you to enter a program name in the Name box and any values that need to be passed to the program as parameters in the Parameters box. The example driver program you created previously used three prompts; one for output device, one for a starting date, and one for an ending date. When your layout program executes the driver program, it needs to pass parameters that will be assigned to each of the prompts in the driver program. Earlier, you created prompts for the layout program by copying the prompts from the driver program. The values that the user enters at the prompt for the start date and end date can be passed to the driver program. Since you modified the driver program to load data into a record structure, it needs to select into the null device ("NL:") instead of the output device that the user enters at the prompt. This can be accomplished two different ways. First, you could remove the prompt for the output device from the driver program, and modify the select into clause from Select INTO $OutDev to Select INTO "NL:". The second option is to simply pass "NL:" as the first parameter when you set the layout driver.

2. In the Name box, enter **1_*your_initials*_Example_Driver**.

If your 1_*your_initials*_Example_Driver program, is a cclgroup1 object, you will need to append :group1 to the program name when you enter it in the Name: box. Use CCLPROT to determine if your program is a cclgroup1 or cclgroup0 (DBA) object.

3. In the Parameters box, enter **"NL:", $Sdate, $Edate**. Your Layout Driver dialog box should look similar to the following example:



4. Click OK to close the Layout Driver dialog box.

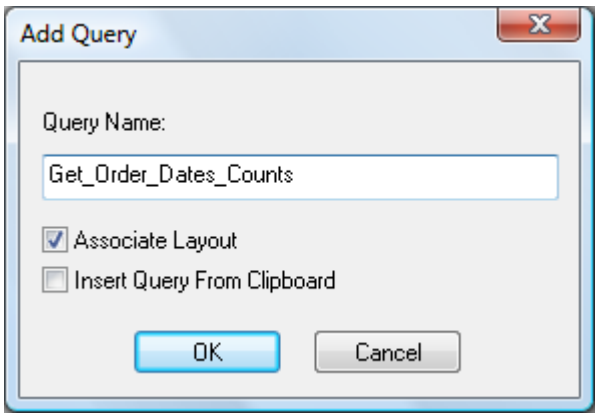
Accessing and Displaying the Data From the Record Structure

Since the layout program defines the record structure and calls the driver program to load it, the record structure is known to the layout program and items in the record structure can be used as the source for items that are placed on the layout. The driver program is loading both a list of dates and a count of the number of orders placed on that date into the Order_Dates record structure. We want our layout program to display the date and count in textual format but also create a graph that displays the number of orders per day. We therefore need to access each of the values in the record structure list. A simple way to access all the values in the record structure list is to use a query to assign the values in the list to select expressions.

1. From the Tools menu, select Query Builder. The Add Queries dialog box opens.

2. Click Add. The Add Query dialog box opens.

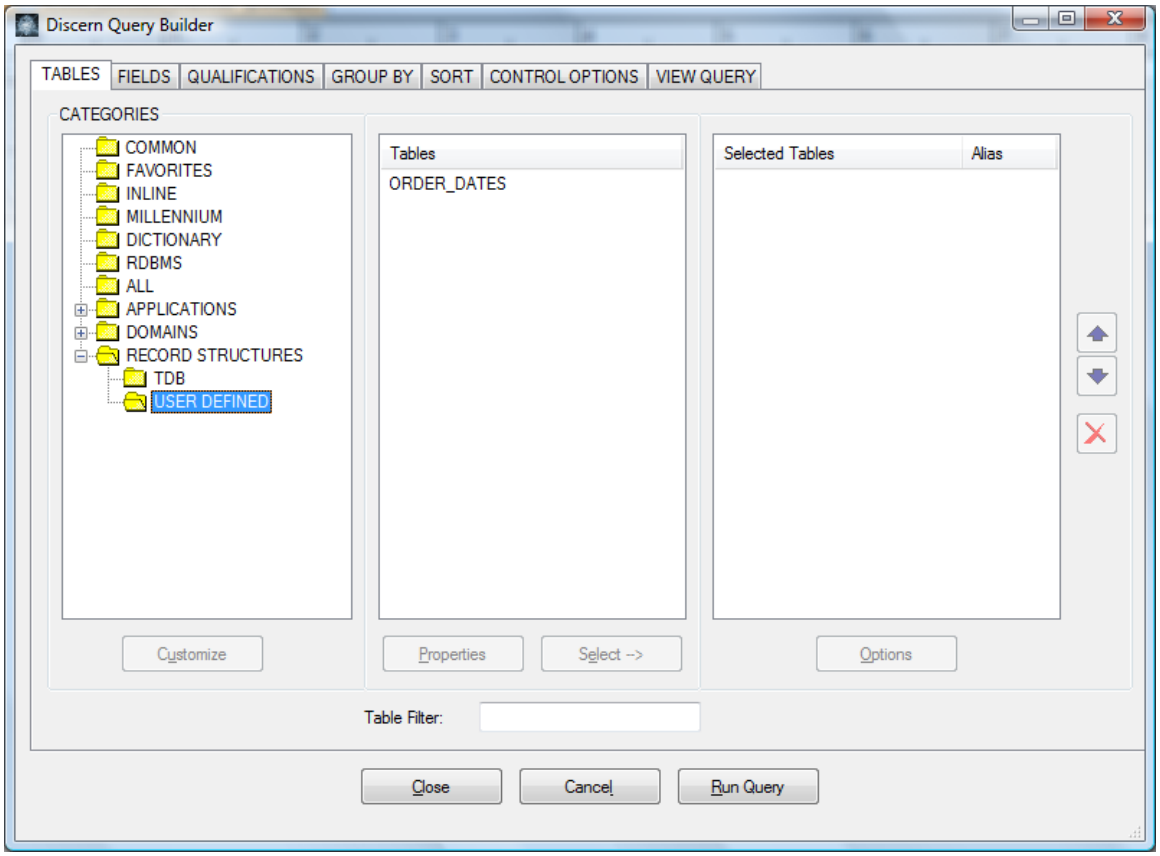
3. In the Query Name box, enter **Get_Order_Dates_Counts** and select the Associate Layout option. Your Add Query dialog box should look similar to the following example:



4. Click OK. The Discern Query Builder window opens.

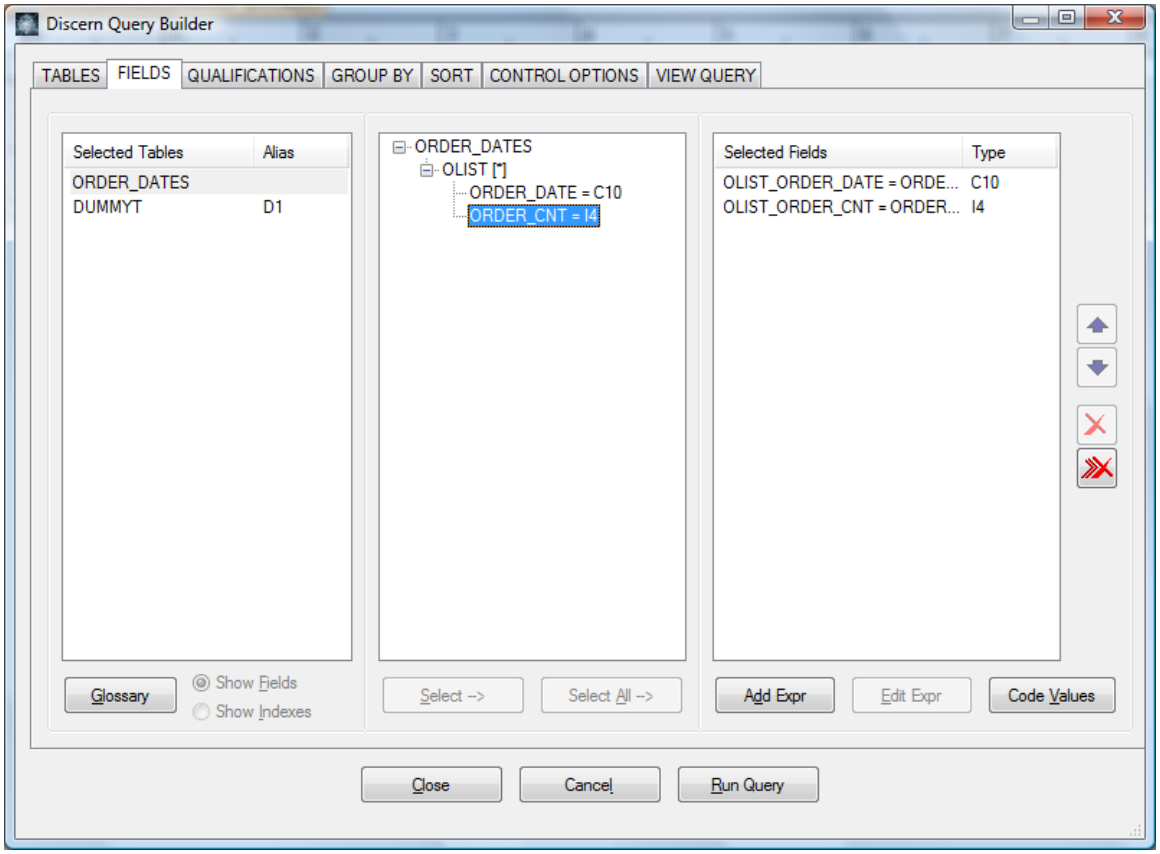
Doc ID: A1001001A08G07B34433A94851

5. On the Tables tab, expand the Record Structures tree and select the User Defined category. The ORDER_DATES record structure is displayed in the Tables column. Your Discern Query Builder window should look similar to the following example:

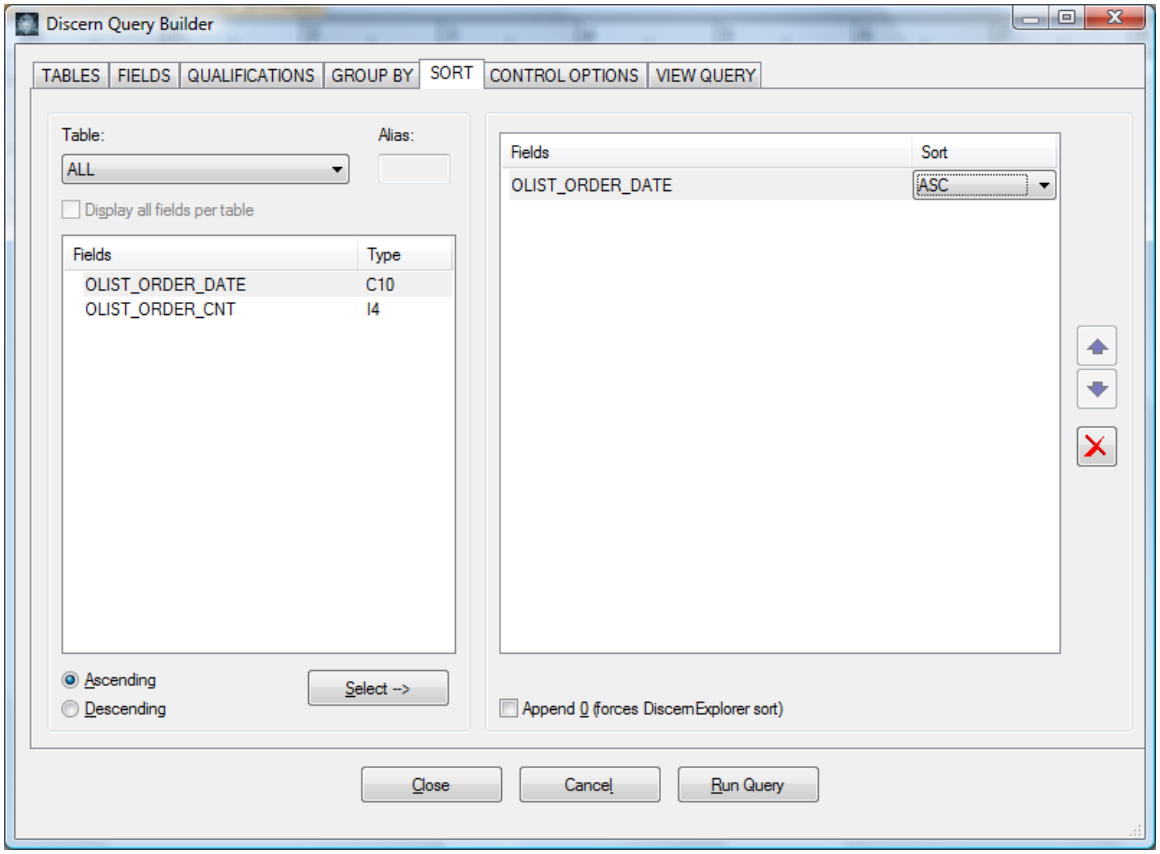


6. Double-click the ORDER_DATES record structure to move it to the Selected Tables column.

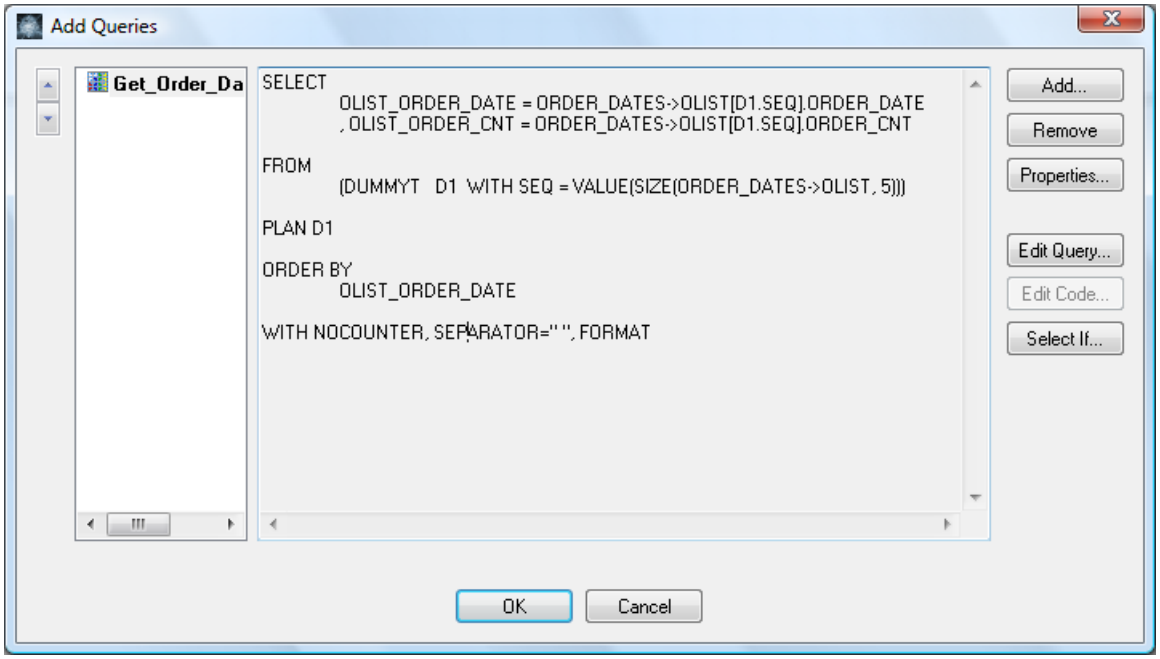
7. Click the Fields tab, expand the Olist tree, and double-click the ORDER_DATE and ORDER_CNT items to add them to the Selected Fields list. Double-clicking a record structure list item adds the DUMMYT table to the Tables tab and creates expressions in the Selected Fields column using *dummyt_alias*.seq to reference each position of the record structure list. Creating an expression for each item in a record structure makes it easier to reference the items in your layout. Your Discern Query Builder dialog box should look similar to the following example:



8. Click the Sort tab and sort by the OLIST_ORDER_DATE expression. Your Discern Query Builder dialog box should look similar to the following example:

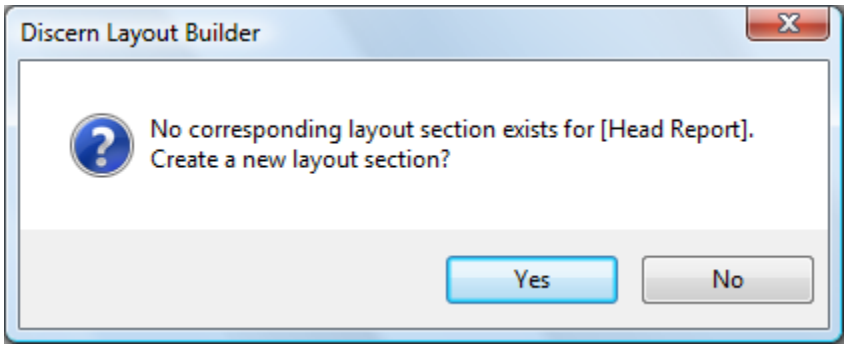


9. Click Close. The Add Queries dialog box opens.



10. Click OK to close the Add Queries dialog box. The Layout Workspace dialog box opens.

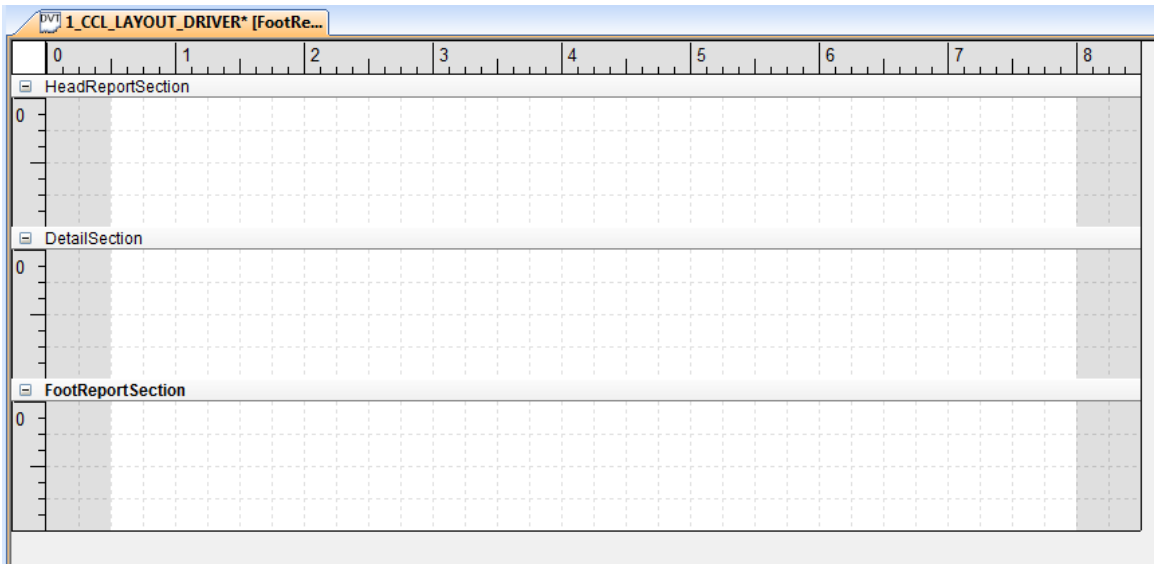
11. Check the Head Report Reportwriter Sections check box. The No corresponding layout section exists for [Head report]. Create a new layout section? message shown below opens.



12. Click Yes to add a HeadReportSection Layout Section.

13. Check the Detail and Foot Report Reportwriter Sections check boxes.

14. Click Yes at the message to create a new FootReportSection Layout Section. Your layout should resemble the following example:



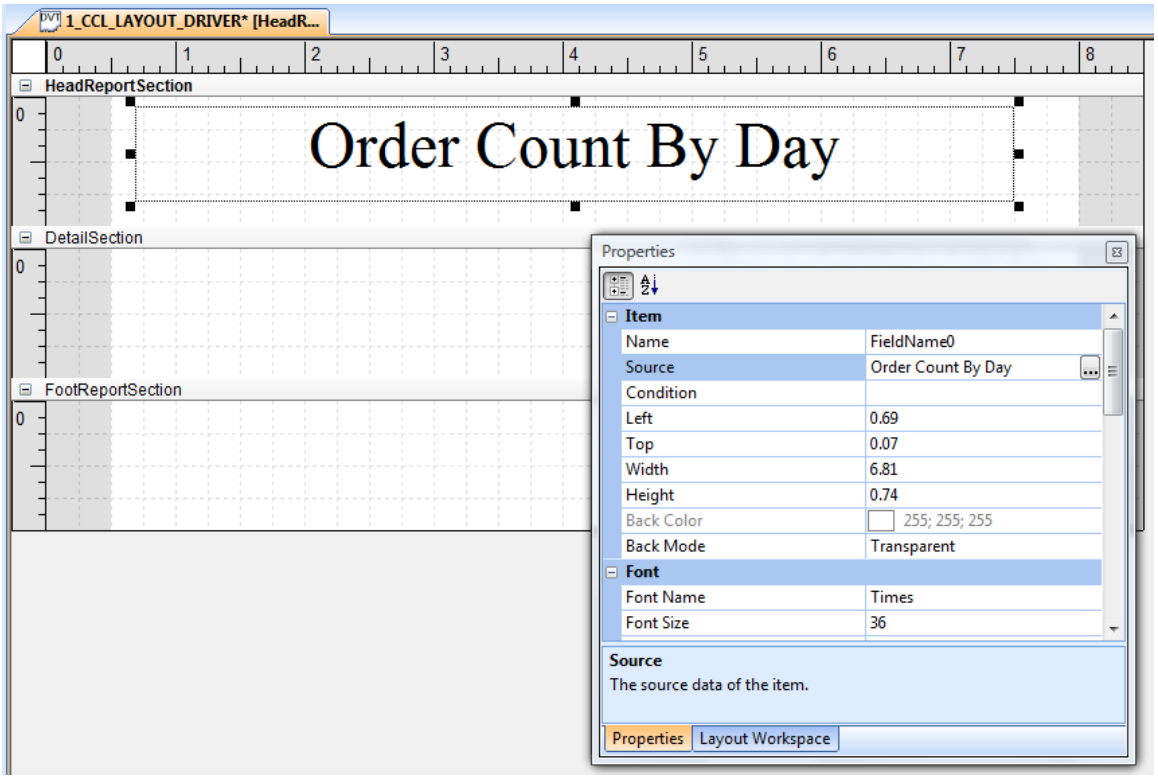
If your layout is displayed differently, you may want to modify the display using the View menu. For the example above, the following items are selected on the View menu: Properties, Horizontal Ruler, Vertical Ruler, Grid, Margins, and Section Title Bar.

15. Use the Label Tool  to add a report title Order Count By Day to the HeadReportSection.

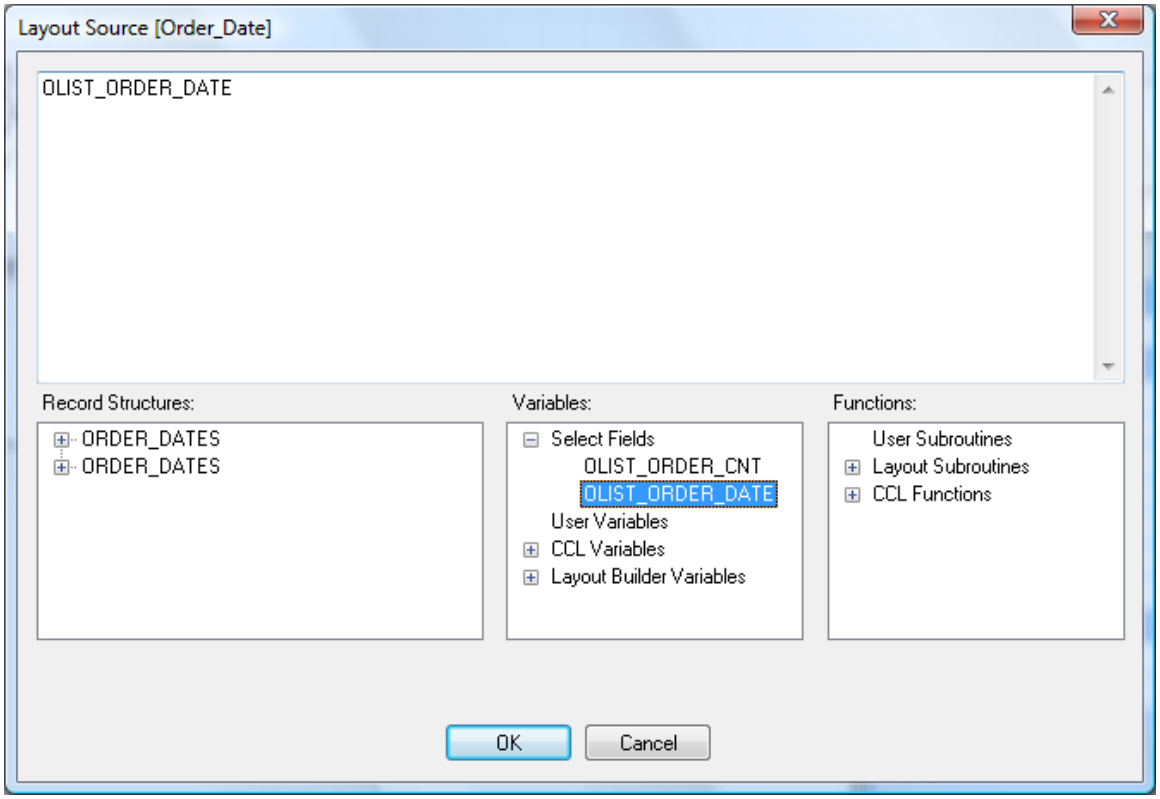
16. Use the Formatting toolbar to set the following values:

- The font to Times
- The font size to 36
- Center the text

If the Formatting toolbar is not displayed, from the View menu, select Toolbars > Formatting. Your layout should resemble the following example:



17. Use the Text Tool  to place a text item in the DetailSection.
18. In the Properties dialog box, modify the Name property to **Order_Date**.
19. Click in the Source box to activate the ellipsis  button. Use the ellipsis  button to open the Layout Source [Order_Date] dialog box.
20. Expand the Select Fields tree in the Variables column and double-click the OLIST_ORDER_DATE expression to add it as the Layout Source. Your Layout Source [Order_Date] dialog box should look similar to the following example:



21. Click OK to close the Layout Source [Order_Date] dialog box.

22. Use the Text Tool  to place a text item to the right of the Order_Date in the DetailSection.

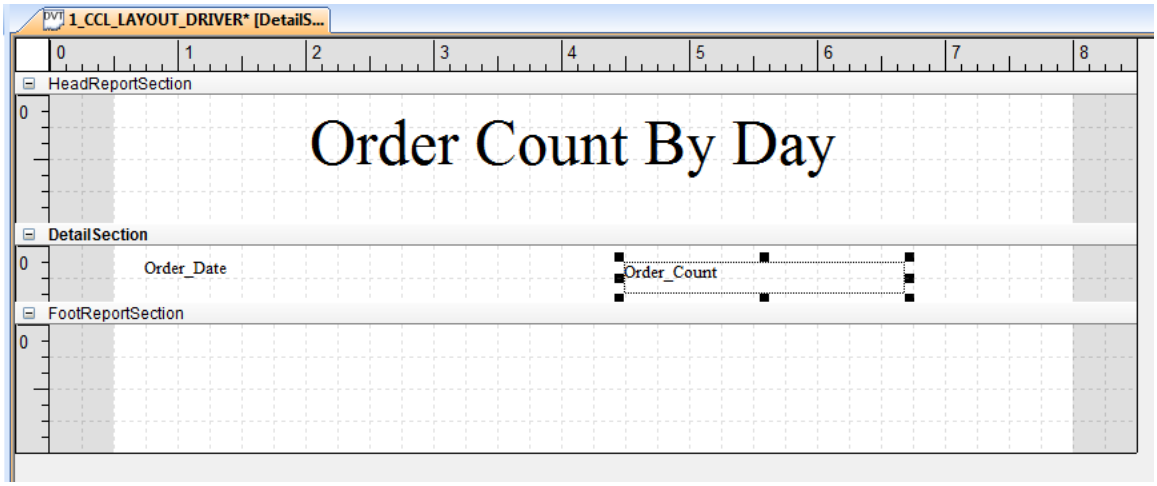
23. In the Properties dialog box, modify the Name property to **Order_Count**.

24. Click in the Source box to activate the ellipsis  button. Use the ellipsis  button to open the Layout Source [Order_Date] dialog box.

25. Expand the Select Fields tree in the Variables column and double-click the OLIST_ORDER_CNT expression to add it as the Layout Source.

26. Click OK to close the Layout Source [Order_Date] dialog box.

27. Slowly move your pointer over the FootReportSection title bar. At the top of the title bar the pointer will change to the vertical resize pointer. Use the vertical resize to move the bottom of the DetailSection up to just underneath the items you just added. Your layout should look similar to the following example:



28. Use the vertical resize to move the bottom of the FootReportSection down to make the total height of the section three inches.

29. Use the Graph Tool  to create a graph item that covers most of the FootReportSection.

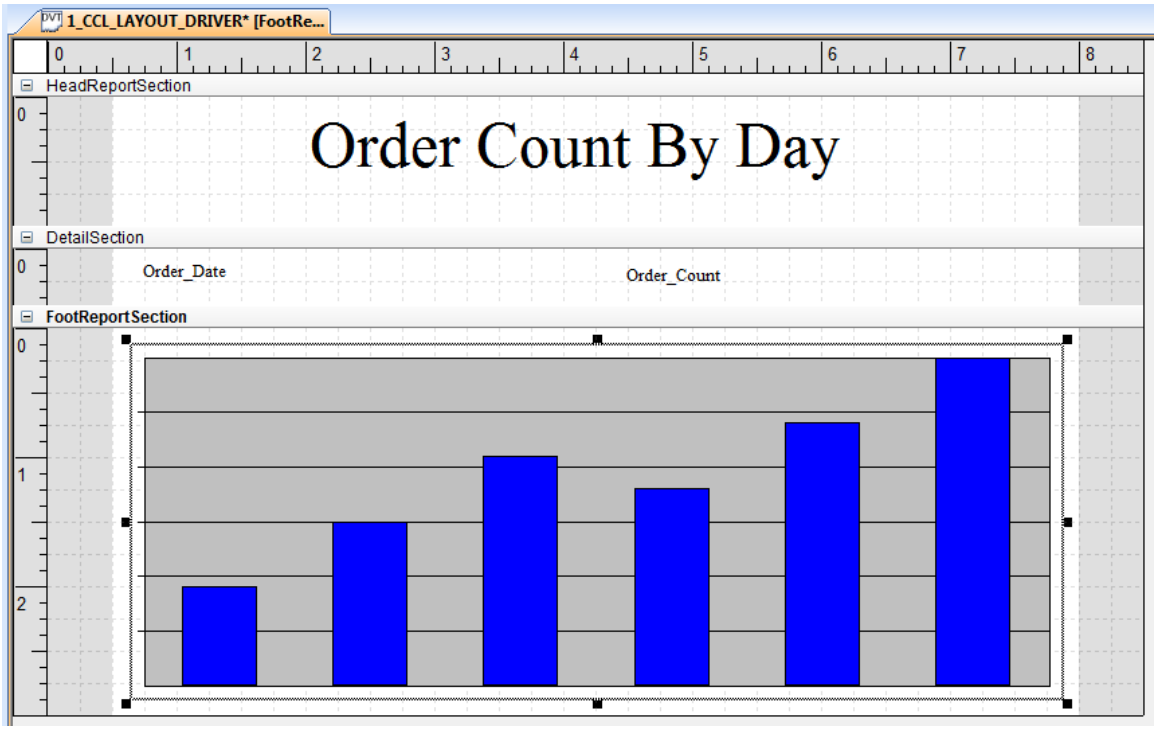
30. On the Data Source tab of the Graph Properties dialog box, perform the following actions:

- Select Get_Order_Dates_Counts as the query
- Use OLIST_ORDER_DATE as the Axis Field and Axis Label

31. On the Series tab of the Graph Properties dialog box, perform the following actions:

- Modify the Name property of Series 1 to **Order Count**.
- Enter **OLIST_ORDER_CNT** in the Data Values field, or click the ellipsis  button and select OLIST_ORDER_CNT from the Select Fields list.

32. Click OK to close the Graph Properties dialog box. Your layout should look similar to the following example:

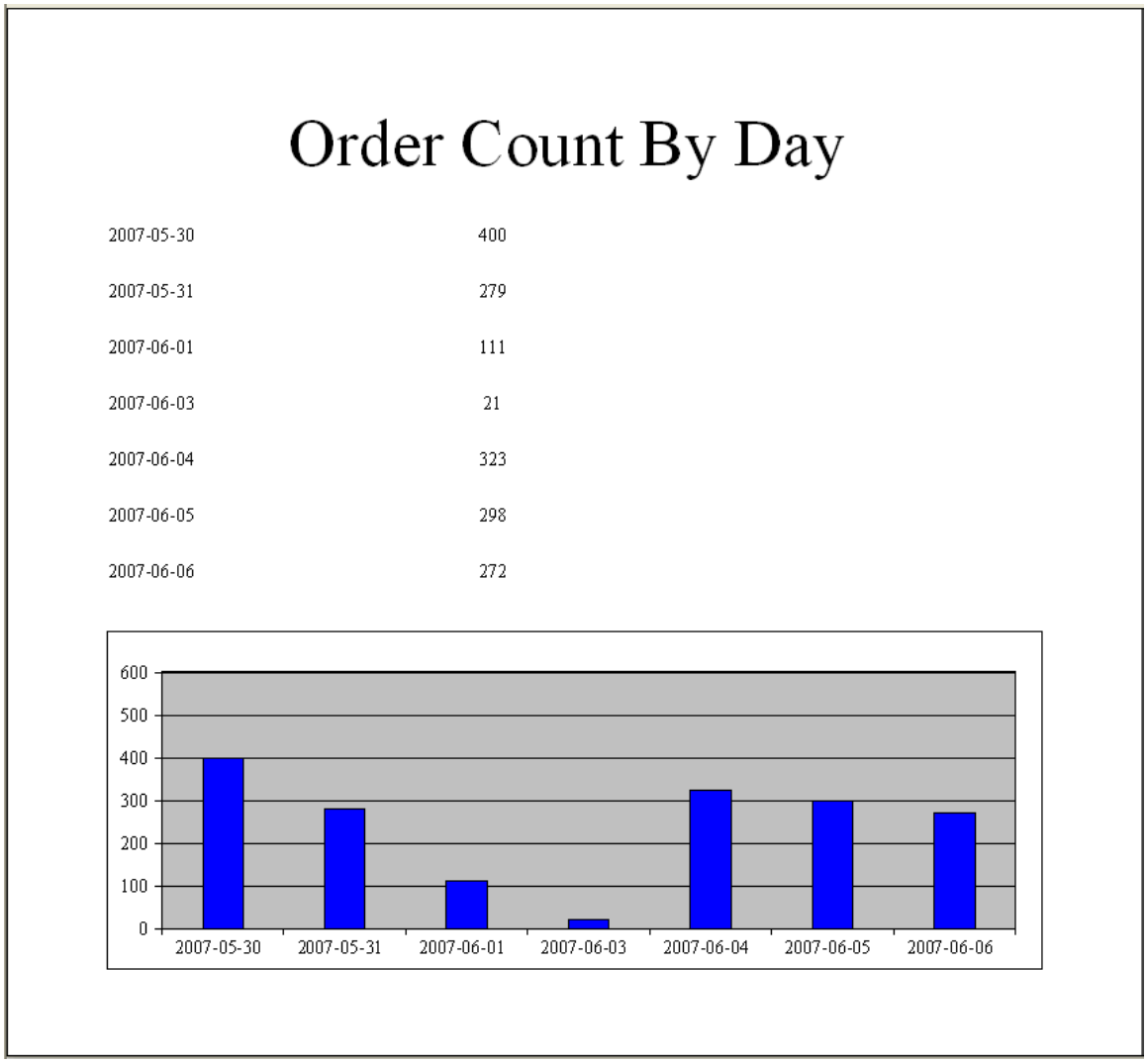


33. Save the layout.

34. Execute the layout by selecting Run "1_*your_initials*_Layout_Using_Driver" from the Build menu or clicking the Run Prompt Program button on the toolbar.

35. Use MINE as the output device.

36. Select starting and ending dates that allow you to qualify orders placed over several different days. Your output should look similar to the following example:



Displaying Data From Multi-Level Record Structures

Multi-level record structures are often used to temporarily store data gathered from several select statements. Once the data has been gathered, it is formatted for display. The following example demonstrates how to use a driver program to load data into a multi-level record structure and how to use a layout program to format the data for display.

1. Using DVDev, from the File menu select New and create a new program named 1_*your_intials*_multi_drv.prg.

The following code creates a multi-level record structure named Person_Enc_Alias, which stores information about people, their encounters, and their aliases. Two selects are used to get this information and load it into the record structure.

```
declare num  = i4
declare pcnt = i4
declare ecnt = i4
declare acnt = i4
declare pos  = i4

record person_enc_alias (
1 plist [*]
 2 person_id = f8
 2 name = c40
 2 elist [*]
  3 reg_dt_tm = c20
  3 type = c40
 2 alist [*]
  3 alias = c40
  3 type = c40
)


;get person and encounter information for person records that
;have been updated today
select into "nl:"
 p.person_id,
 p.name_full_formatted,
 e.encntr_id,
 e.reg_dt_tm,
 e_encntr_type_disp = uar_get_code_display( e.encntr_type_cd )
from
 person  p,
 encounter  e

plan p  where  p.updt_dt_tm between
   cnvtdatetime(curdate, 0) and cnvtdatetime(curdate,curtime3)
join e  where  p.person_id = e.person_id
order by
 p.name_full_formatted,
 p.person_id,
 e.reg_dt_tm,
 0
head report
 pcnt = 0
 ecnt = 0
head p.person_id
 pcnt = pcnt +1
 if(mod(pcnt,10) = 1)
  stat = alterlist(person_enc_alias->plist, pcnt +9)
 endif
 person_enc_alias->plist[pcnt].person_id  = p.person_id
 person_enc_alias->plist[pcnt].name = p.name_full_formatted
 ecnt = 0
detail
 ecnt = ecnt +1
 if(mod(ecnt, 10) = 1)
  stat = alterlist(person_enc_alias->plist[pcnt].elist, ecnt +9)
 endif
 person_enc_alias->plist[pcnt].elist[ecnt].reg_dt_tm =
    format(e.reg_dt_tm, "dd-mmm-yyyy hh:mm:ss")
 person_enc_alias->plist[pcnt].elist[ecnt].type  = e_encntr_type_disp
foot p.person_id
 stat = alterlist(person_enc_alias->plist[pcnt].elist, ecnt )
foot report
 stat = alterlist(person_enc_alias->plist, pcnt )
with maxrec = 500 , nocounter, separator=" ", format
```

```
;get alias information for person_ids that are currently in the record
SELECT INTO "NL:"
 P.ALIAS,
 P_PERSON_ALIAS_TYPE_DISP = UAR_GET_CODE_DISPLAY(P.PERSON_ALIAS_TYPE_CD)
FROM
 PERSON_ALIAS  P
where
 expand(num,1,pcnt,p.person_id,person_enc_alias->plist[num].person_id)
head p.person_id
 pos =
 locateval(num,1,pcnt,p.person_id,
    person_enc_alias->plist[num].person_id)
 acnt = 0
detail
 acnt = acnt +1
 if(mod(acnt, 10) = 1)
  stat = alterlist(person_enc_alias->plist[pos].alist, acnt +9)
 endif
 person_enc_alias->plist[pos].alist[acnt].alias = p.alias
 person_enc_alias->plist[pos].alist[acnt].type = P_PERSON_ALIAS_TYPE_DISP
foot p.person_id
 stat = alterlist(person_enc_alias->plist[pos].alist, acnt)
WITH  NOCOUNTER, SEPARATOR=" ", FORMAT
call echorecord(person_enc_alias)
```

2. Copy and paste the above code above into the 1_*your_intials*_multi_drv.prg file you created in Step 1.

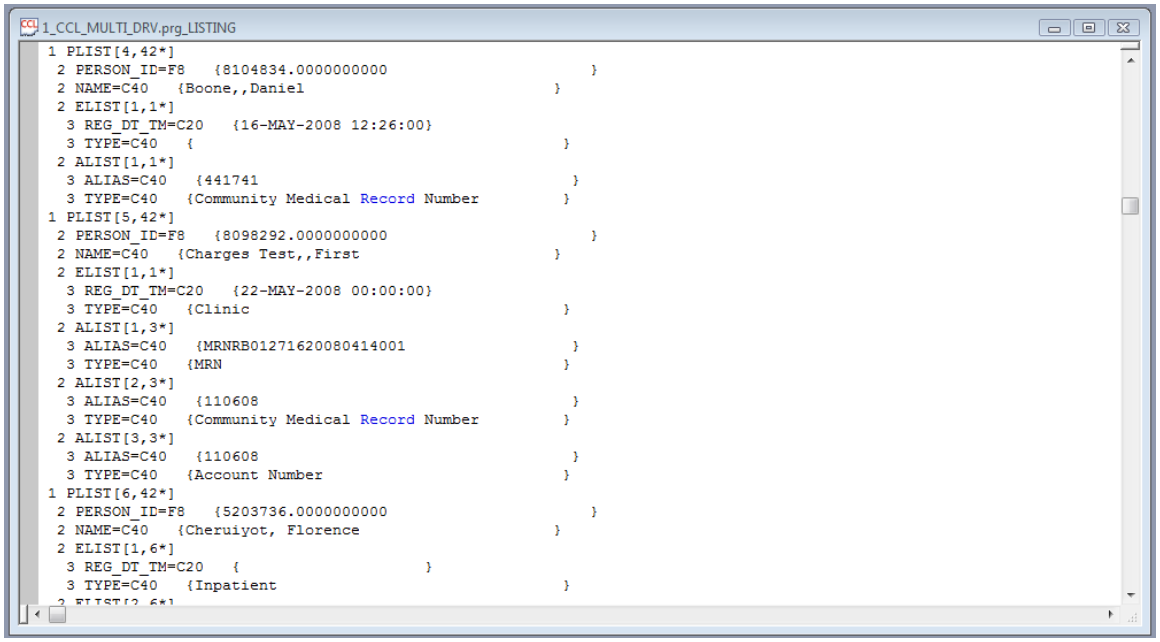
3. Enter **1_*your_intials*_multi_drv GO** at the end of your 1_*your_intials*_multi_drv.prg file. Ensure this command is placed after the End Go.

4. Save and Include/Compile your 1_*your_intials*_multi_drv.prg file to create and execute your 1_*your_intials*_multi_drv program.

5. Press CTRL+L, click the Listing button, or from the View menu select Listing to open the listing file.

The listing file is created automatically by DVDev when you Include/Compile a file. The Call Echorecord(person_enc_alias) command at the end of your 1_*your_intials*_multi_drv program echoes the contents of the person_enc_alias record structure to the listing file. As you scroll down through the listing file, you will see first the commands that were compiled and then the output of the Call Echorecord command.

6. Review the output of the Call Echorecord command in the listing file and verify your 1_*your_intials*_multi_drv program is creating the record structure and populating it with the names and person IDs of people, the encounter type and registration date and time of their encounters, and their aliases and alias types. Your listing file should look similar to the following example:



If your Person_Enc_Alias record structure is empty you may need to modify the qualifications on the first select command to allow it to find some people with encounters and aliases.

Each person stored in the record structure can have multiple encounters and multiple aliases. We want to use Layout Builder to display this information in a user-friendly format. In order to do this, we will need to traverse the PLIST and display the person information. On each person we will need to traverse first the ELIST to display the encounter information and then the ALIST to display the alias information. Joins to the DUMMYT table can be used to traverse the level 1 PLIST and one of the level 2 lists (ELIST or ALIST). A For loop can then be used to traverse the other Level 2 list. The following source code example shows how these methods could be used to display the data from the Person_Enc_Alias record structure in a simple ASCII report.

```
SELECT INTO $outdev
;Create expressions for the person and encounter information
 plist_person_id = person_enc_alias->plist [D1.SEQ].person_id,
 plist_name = person_enc_alias->plist [D1.SEQ].name,
 elist_type = person_enc_alias->plist [D1.SEQ].elist [D2.SEQ].type,
 elist_reg_dt_tm =
  person_enc_alias->plist [D1.SEQ].elist [D2.SEQ].reg_dt_tm
FROM
;Use dummyt tables to traverse the plist and the elist.
;Setting SEQ to the size of plist will cause d1.seq
;to be set to 1 and incremented by 1 until it is equal
;to the size of plist.  D1.seq can then be used to
;look at each position of the plist.
( DUMMYT  D1 WITH SEQ = VALUE(SIZE(person_enc_alias->plist,5))),
( DUMMYT  D2 WITH SEQ = 1 )
PLAN D1 WHERE
 MAXREC(D2, SIZE(person_enc_alias->plist[D1.SEQ].elist,5))
;Using maxrec()causes Discern Explorer to get the size of the
;elist for each person.  D2.seq is then set to 1 and
;incremented by 1 until it is equal to the size of each
;elist.  D2.seq can then be used to look at each position
;of each elist.
JOIN D2
order by plist_person_id
head report
 acnt = 0
 totalacnt = 0
head plist_person_id
 ;Display the name and id of each person in the record using
 ;the expressions created in the select list.
 col 0 plist_name
 col +1 plist_person_id
 row +1
 ;Get the number of aliases for the person and use a for loop
 ;to display each alias and alias type.
 totalacnt = size(person_enc_alias->plist[D1.seq].alist,5)
 for(acnt = 1 to totalacnt)
  col 10 person_enc_alias->plist[D1.seq].alist[acnt].alias
  col +2 person_enc_alias->plist[D1.seq].alist[acnt].type
  row +1
 endfor
detail
 ;Display the encounter types and registration date/times for the
 ;person using the expressions created in the select list.
 col 10 elist_type
 col +2 elist_reg_dt_tm
 row +1
WITH  NOCOUNTER, SEPARATOR=" ", FORMAT
```

We will now begin the process of creating the same code to traverse the record structure lists; however, we will use the Layout Builder functionality to display the data instead of the simple row and col commands shown in the ASCII version above. Using the Layout Builder in this way allows us to create code using all of the Layout Builder functionality that would otherwise be difficult to create if we were writing the program in a text editor.

7. Using DVDev, from the File menu, select New.

8. From the File Type list, select Layout Program.

9. In the Program Name box, enter **1_*your_initials*_Multi_Layout** and click OK. The New Layout Program dialog box opens.

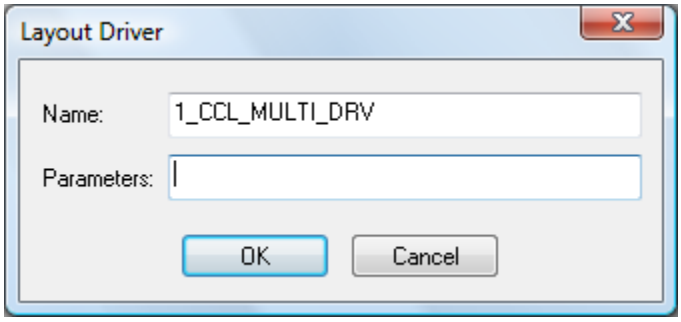
10. Verify the Standard Layout option for the Report Layout and PostScript option for the Output Type are selected and click Next. The Paper Size dialog box is displayed.

11. Keep the defaulted values for the Paper Size and click Finish. The previous two dialog boxes, New Layout Program and Paper Size, populate the basic properties of the layout. These properties can be reviewed at any time by selecting Report Properties dialog box from the Edit menu. A layout with a single section named DetailSection is created.

12. From the Tools menu, select Set Layout Driver. The Layout Driver dialog box opens.

13. In the Name box, enter **1_*your_initials*_Multi_Drv**.

Because your 1_*your_initials*_Multi_Drv program does not have prompts or expect any parameters to be passed to it, leave the Parameters box blank. Your Layout Driver dialog box should look similar to the following example:



If your 1_*your_initials*_Multi_Drv program, is a cclgroup1 object, you will need to append :group1 to the program name when you enter it in the Name: box. Use CCLPROT to determine if your program is a cclgroup1 or cclgroup0 (DBA) object.

14. Click OK to close the Layout Driver dialog box.

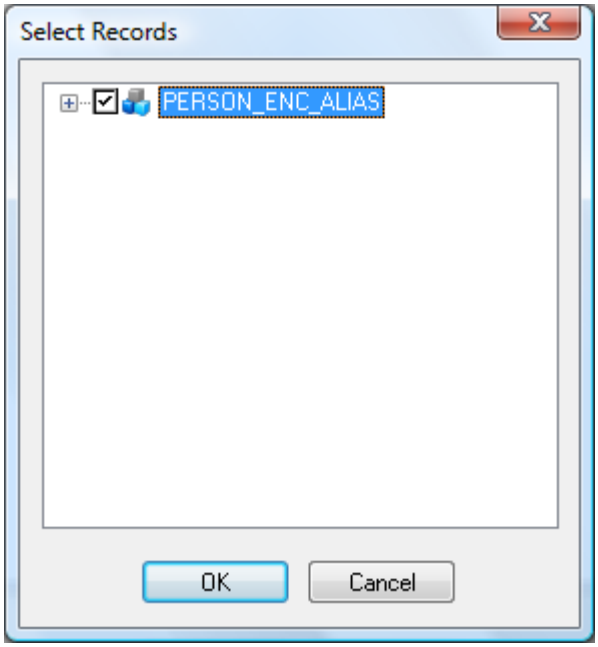
15. From the Tools menu, select Record Builder. The Add Records dialog box opens. Because the record structure definition already exists in the driver program, you can use Load Object to have the Record Builder pull the definition from your 1_*your_initials*_Multi_Drv program into your 1_*your_initials*_Multi_Layout program.

16. Click Load Object. The Load Object dialog box opens.

17. In the Object Name box, enter **1_*your_initials*_Multi_Drv** and click OK. The Select Records dialog box opens.

Each record structure defined in the program object is displayed in the Select Records dialog box. Because your 1_*your_initials*_Multi_Drv program only has one record structure definition, your Select Records dialog box should look similar to the following example:

Doc ID: A1001001A08G07B34433A94851



18. Select the Person_Enc_Alias record and click OK.

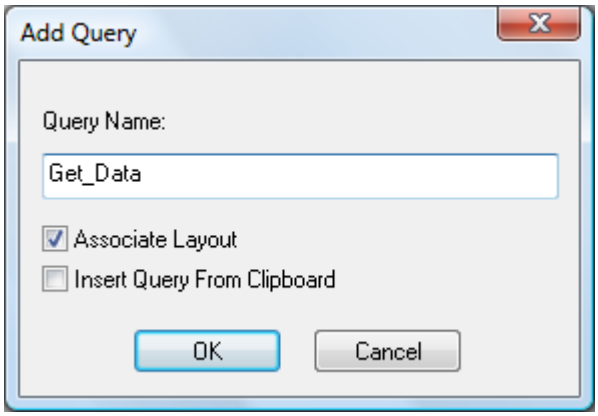
19. Click OK to close the Add Records dialog box.

At this point you have the record command in the layout code and in the program. Since the layout program calls the driver program, it is not necessary to define the record in both 1_*your_initials*_Multi_ Layout and the 1_*your_initials*_MULTI_DRV programs. For the subsequent steps defined below the record command in the driver program has been deleted. Having the record command in the driver will result in the record structure listed two times in the Layout Source [Code Segment] dialog box.

20. From the Tools menu, select Query Builder. The Ad Queries dialog box opens.

21. Click Add. The Add Query dialog box opens.

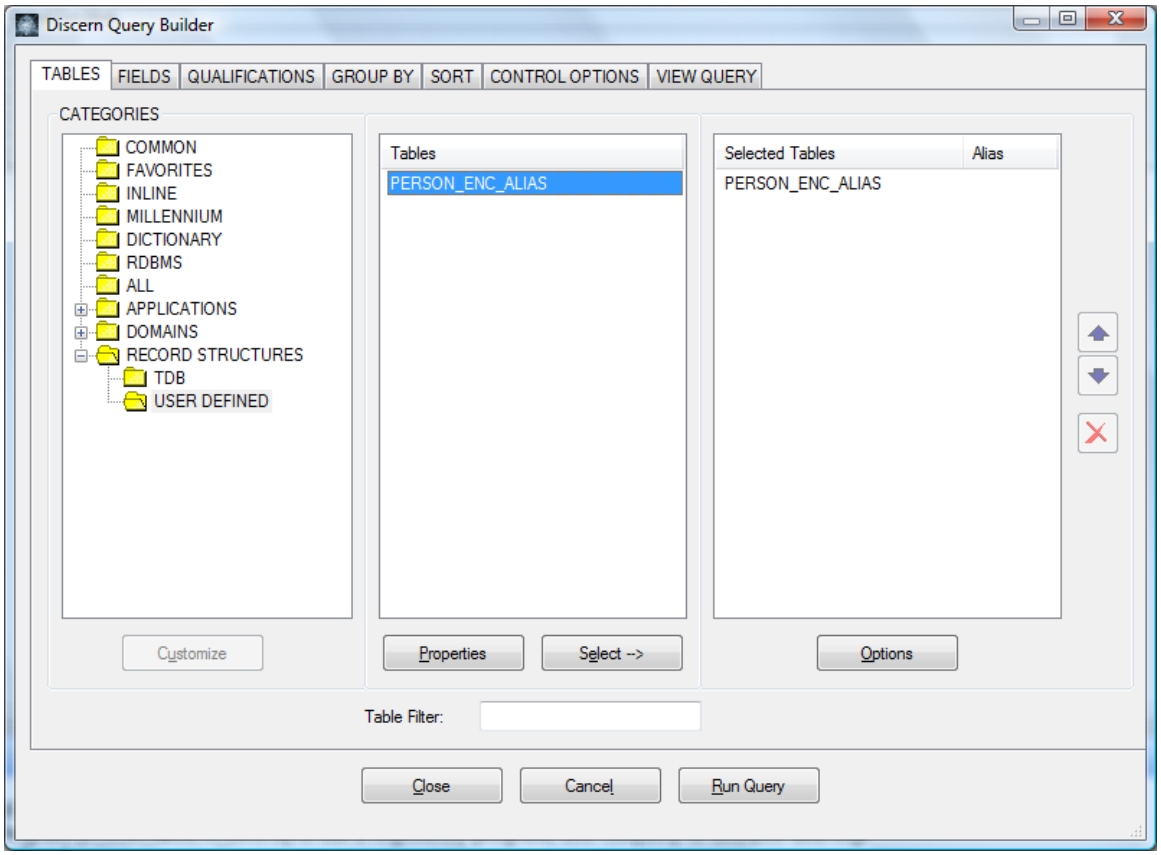
22. In the Query Name box, enter **Get_Data** and verify that the Associate Layout is selected. Your Add Query dialog box should look similar to the following example:



23. Click OK. The Discern Query Builder window opens.

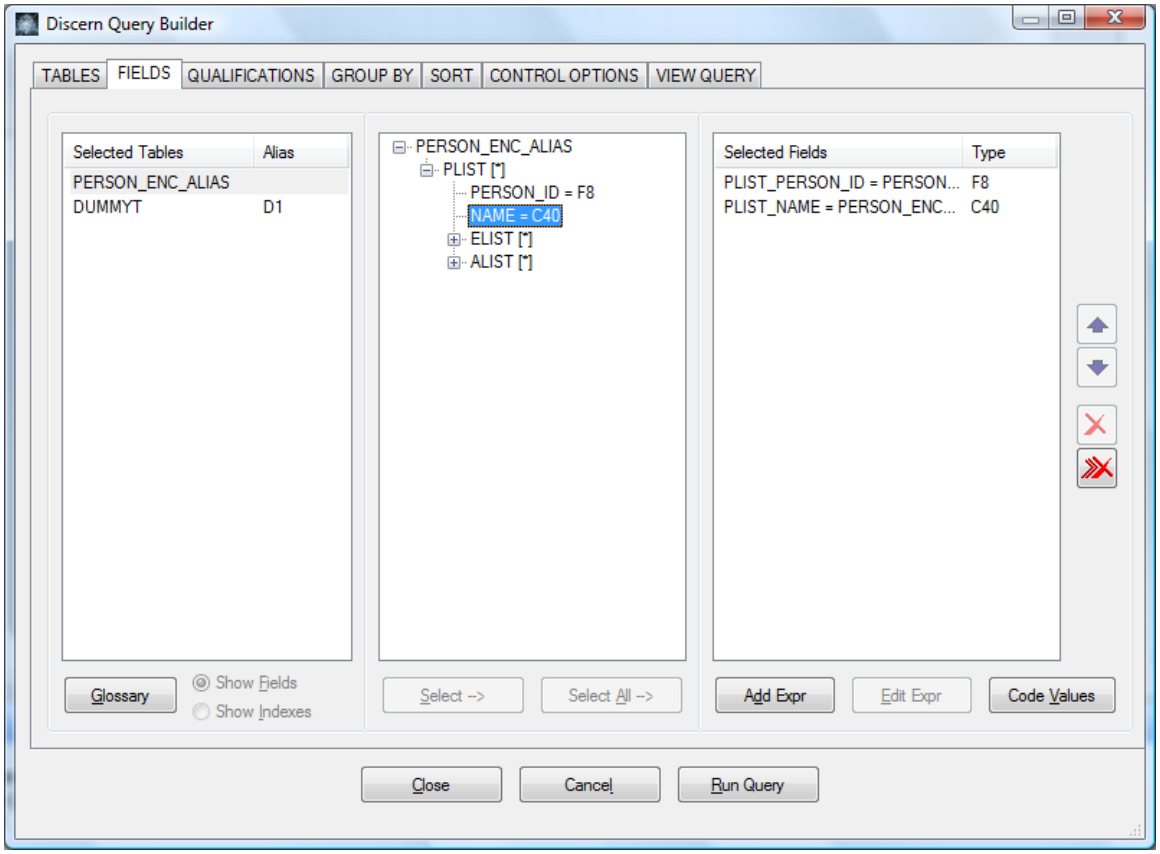
24. On the Tables tab expand the Record Structures list and select the User Defined category. The PERSON_ENC_ALIAS record structure is displayed in the Tables list.

25. Double-click the PERSON_ENC_ALIAS record structure to move it to the Selected Tables column. Your Discern Query Builder window should look similar to the following example:



26. On the Fields tab expand the PLIST and double-click the PERSON_ID and NAME items to add them to the Selected Fields list.

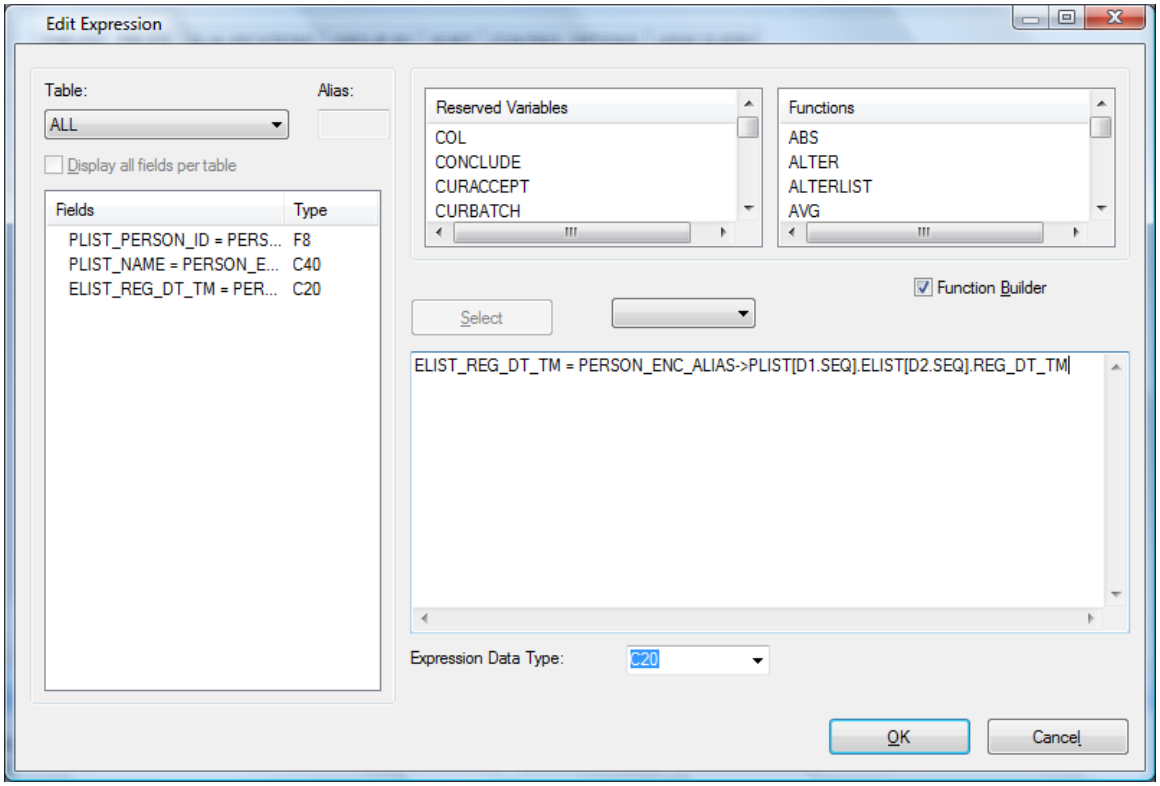
Double-clicking a record structure list item adds the DUMMYT table to the Tables tab and creates expressions in the Selected Fields column using *dummyt_alias*.seq to reference each position of the record structure list. Your Query Builder dialog box should look similar to the following example:



27. On the Fields tab expand the ELIST and double-click the Reg_DT_TM item to add it to the Selected fields list.

Since the REG_DT_TM item is under the Level 2 ELIST, the Query Builder will add a second reference to the DUMMYT table to the Tables tab and create an expression in the selected fields column using *dummyt_alias*.seq to reference each position of the list.

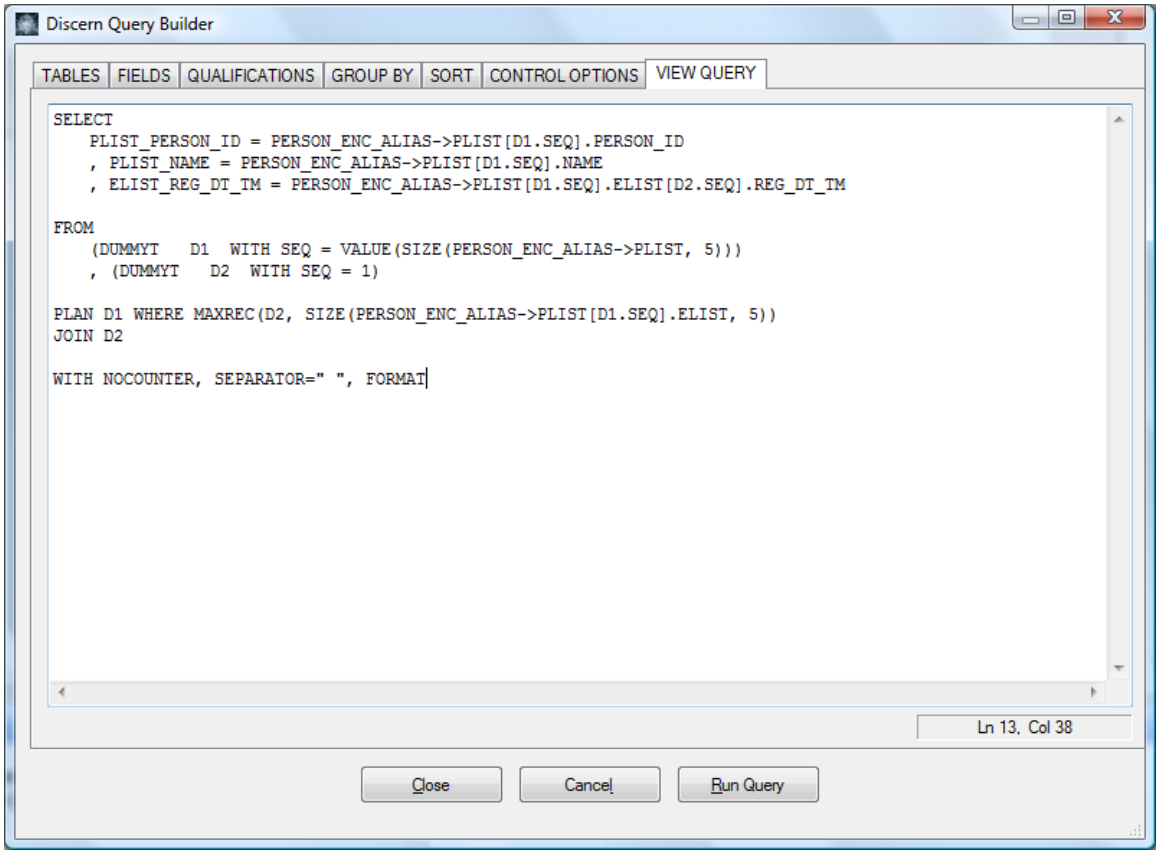
28. Right-click the ELIST_REG_DT_TM expression that was created when you double-clicked the REG_DT_TM item and select Edit Expression from the context menu to open the Edit Expression dialog shown below.



Notice in the ELIST_REG_DT_TM expression created above that the Query Builder is using D1.SEQ to reference the different positions of the PLIST and D2.SEQ to reference the different positions of the ELIST.

29. Click Cancel to close the Edit Expression dialog box.

30. Click the View Query tab. Your Query Builder is displayed similar to the following:

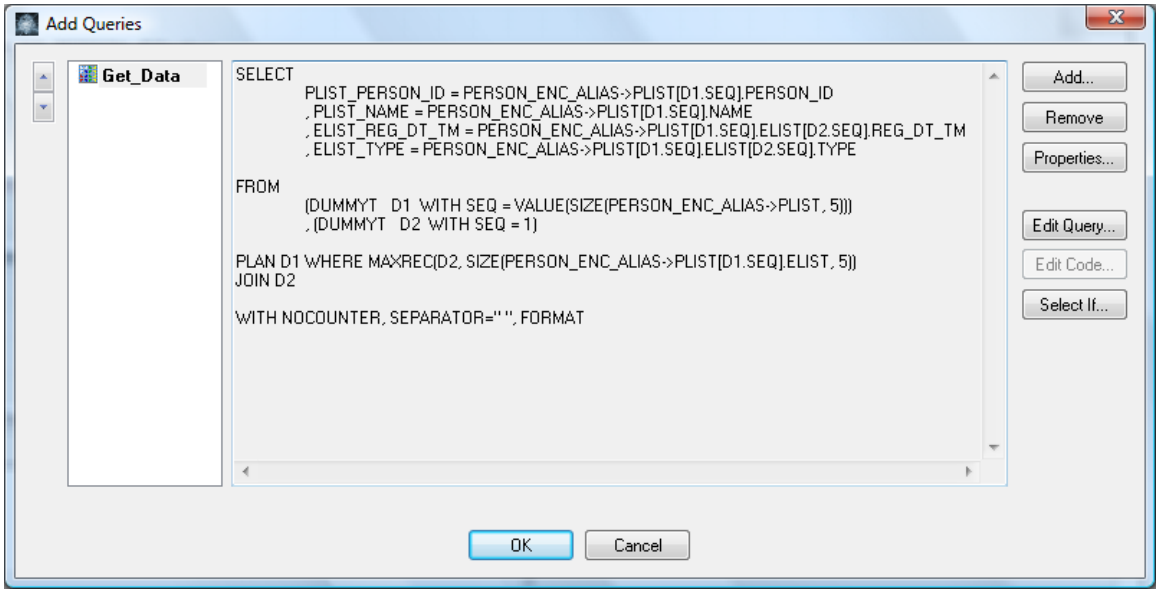


Notice that the Query Builder is defining the first reference to the DUMMYT table as D1 with Seq = the size of the PLIST. This will cause D1.Seq to be set to 1 and incremented by 1 until it reaches the total number of positions in PLIST. This allows D1.Seq to be used to look at each position of the PLIST. The second reference to the DUMMYT table sets Seq = 1. In the Plan clause, the MaxRec() function is used get the size of each ELIST. Using the MaxRec() function causes D2.Seq to be set to 1 and incremented by one until it reaches the total number of positions in each of the ELISTs. This allows Discern Explorer to use D1.seq and D2.Seq in an internal looping structure to traverse the Level 2 Elists contained within the Level 1 PLIST.

31. Click the Fields tab.

32. From the ELIST tree, select Type to add it to the Selected Fields list.

33. Click the View Query tab. Your Query Builder should look similar to the following example:



Creating expressions for the PERSON_ID and NAME items in the PLIST and for the REG_DT_TM and TYPE items in the ELIST makes it easier to display these items in your layout.

34. Click the Sort tab and sort by the PLIST_PERSON_ID expression.

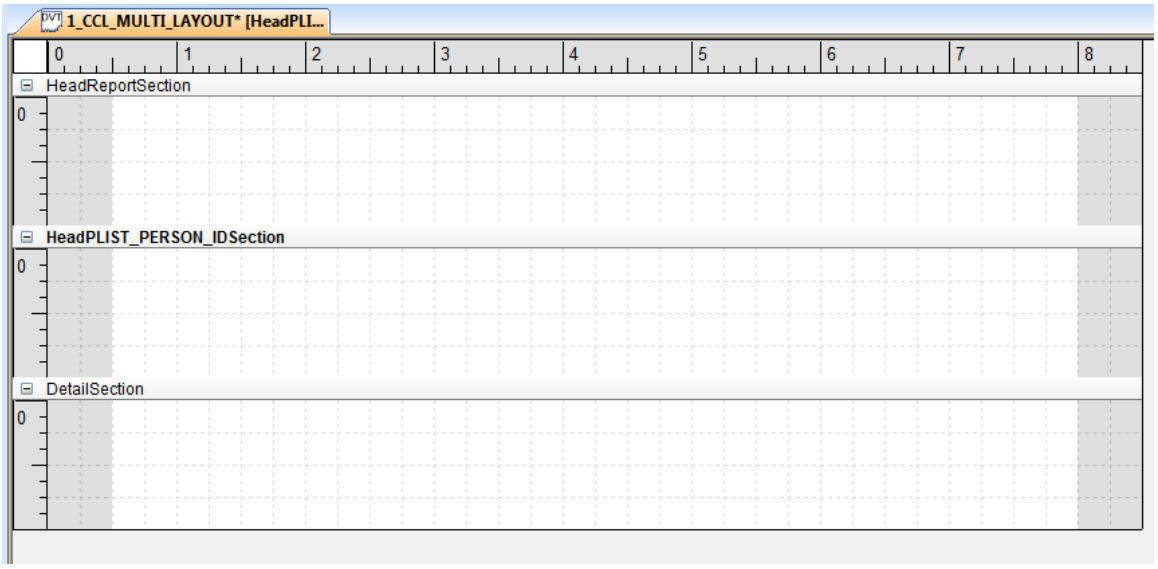
35. Click Close to close the Query Builder dialog box.

36. Click OK to close the Add Queries dialog box.

37. From the Select/Modify Section of the Workspace dialog box, select the Head Report Reportwriter check box. Click Yes to create a layout section associated to the Reportwriter section.

38. Check the Head PLIST_PERSON_ID Reportwriter Sections check box and create a new Layout Section.

39. Check the Detail Reportwriter Sections check box. Since the DetailSection Layout Section already exists you are not prompted to create it. You can expect your layout to be similar to the following:

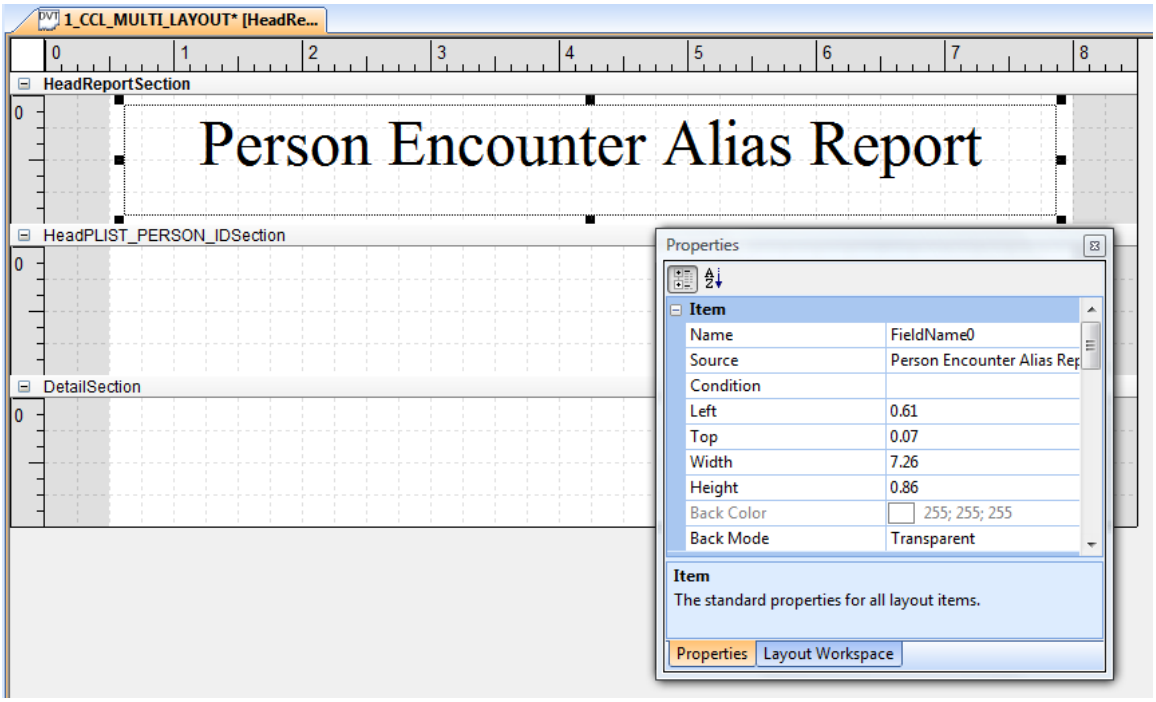


40. Use the Label Tool  to add a report title Person Encounter Alias Report to the HeadReportSection, and size the label box to take up most of the HeadReportSection grid.

41. Use the Formatting toolbar to set the following properties:

- The font to Times
- The font size to 36
- Center the text

Your layout should resemble the following example:



42. Use the Line  Tool to draw a line across the top of the HeadPLIST_PERSON_IDSection.

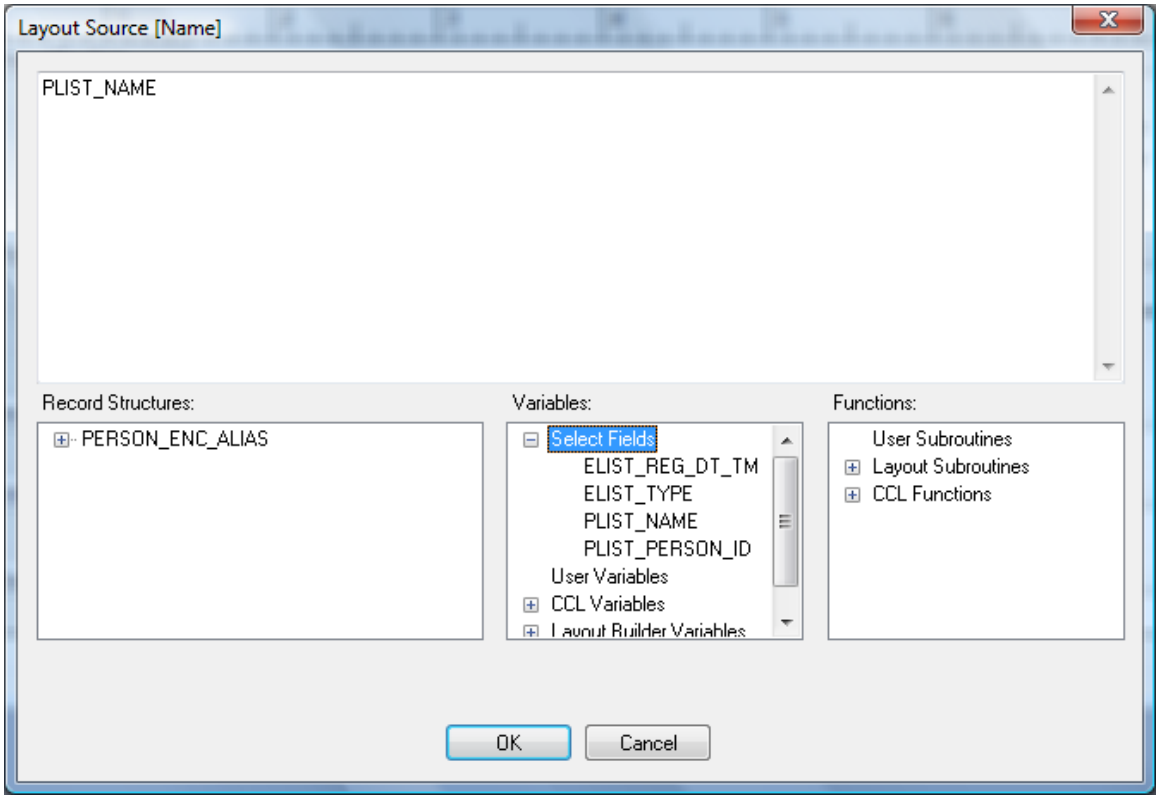
43. In the Properties dialog box, set the Pen Size to 0.028.

44. Use the Text Tool  to add a field to the left side of the HeadPLIST_PERSON_IDSection to display each name from the record structure.

45. In the Properties dialog box, modify the Name property to **Name**.

46. In the Source box click the ellipsis  button. The Layout Source [Name] dialog box opens.

47. Expand the Select Fields list in the Variables: column and double-click the PLIST_NAME expression to add it as the Layout Source. Your Layout Source [Name] dialog box should look similar to the following example:



48. Click OK to close the Layout Source [Name] dialog box.

49. Use the Text Tool  to add a field towards the middle of the HeadPLIST_PERSON_IDSection to display the Person_ID from the record structure.

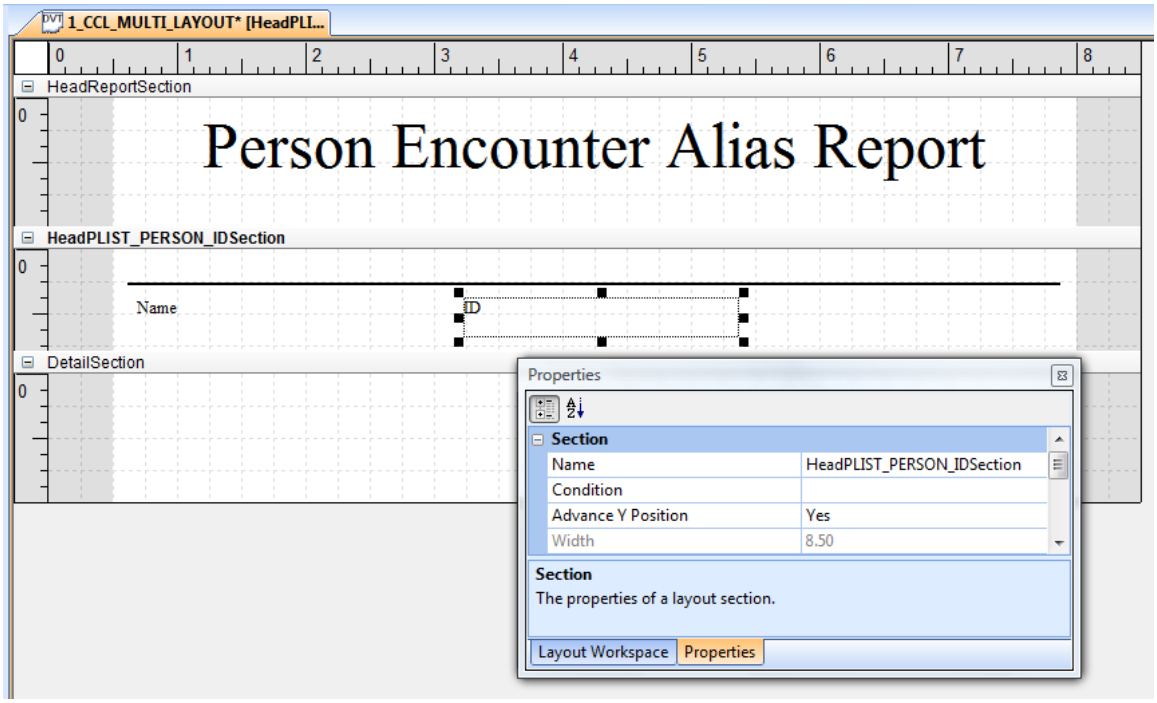
50. In the Properties dialog box, modify the Name property to **ID**.

51. In the Source box click the ellipsis  button. The Layout Source [ID] dialog box opens.

52. Expand the Select Fields list in the Variables: column and double-click the PLIST_PERSON_ID expression to add it as the Layout Source.

53. Click OK to close the Layout Source [ID] dialog box.

54. Slowly move your pointer over the DetailSection title bar. At the top of the title bar the pointer will change to the vertical resize pointer. Use the vertical resize to move the bottom of the HeadPLIST_PERSON_IDSection up to just underneath the items you just added. Your layout should look similar to the following example:



55. Use the Text Tool to add a field towards the right side of the DetailSection to display the encounter type from the record structure.

56. In Properties dialog box, modify the Name property to **E_Type**.

57. In the Source box, click the ellipsis button. The Layout Source [E_Type] dialog box opens.

58. Expand the Select Fields list in the Variables: column and double-click the ELIST_TYPE expression to add it as the Layout Source.

59. Click OK to close the Layout Source [E_Type] dialog box.

60. Use the Text Tool to add a field towards the middle of the DetailSection to display the registration date and time from the record structure.

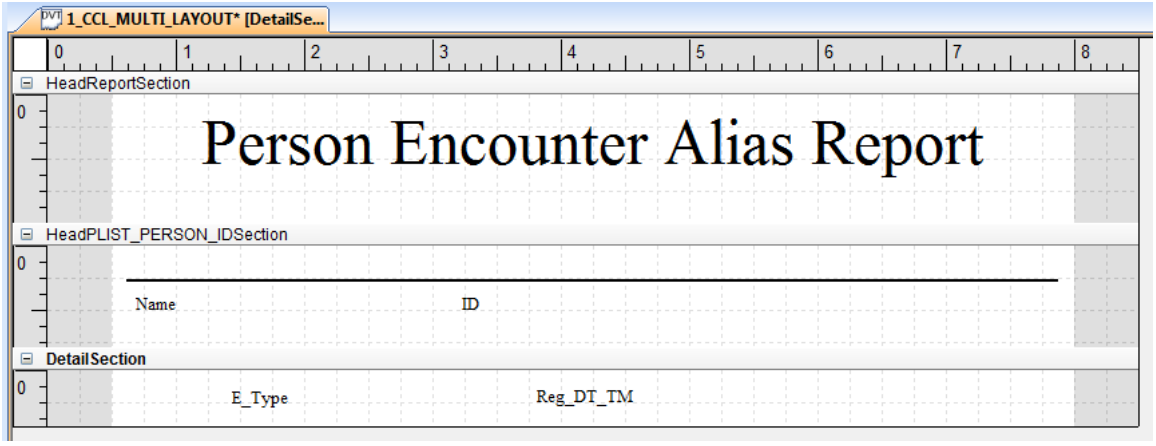
61. In the Properties dialog box, modify the Name property to **Reg_DT_TM**.

62. In the Source box, click the ellipsis button. The Layout Source [Reg_DT_TM] dialog box opens.

63. Expand the Select Fields list in the Variables column and double-click the Elist_Reg_DT_TM expression to add it as the Layout Source.

64. Click OK to close the Layout Source [Reg_DT_TM] dialog box.

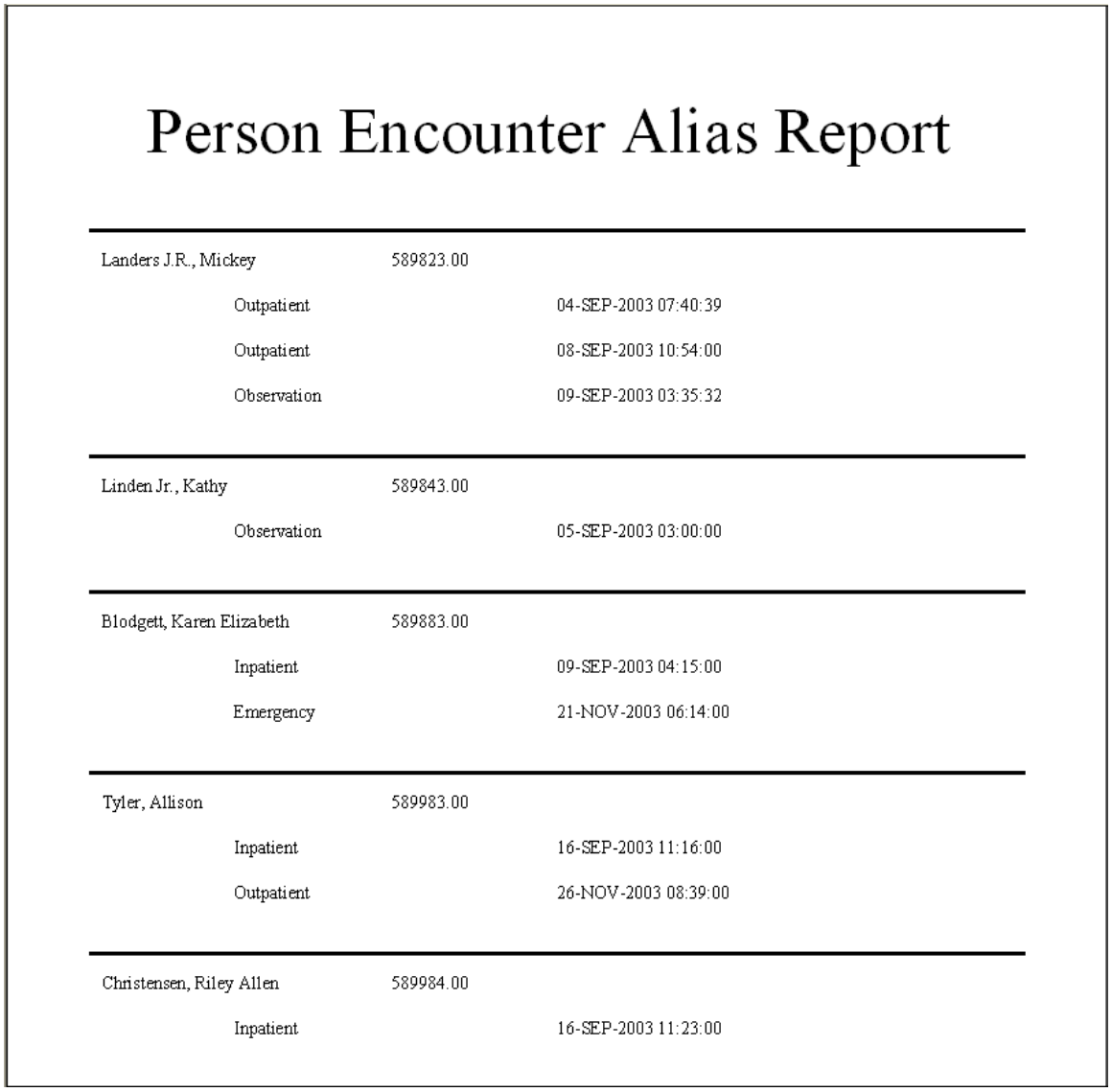
65. Use the vertical resize to move the bottom of the DetailSection up to just underneath the items you just added. Your layout should look similar to the following example:



66. Save the layout.

67. Execute the layout by selecting Run "1_*your_initials*_Multi_Layout" from the Build menu or clicking Run Prompt Program from the toolbar. Your output should look similar to the following example:

Doc ID: A1001001A08G07B34433A94851



Person Encounter Alias Report

Landers J.R., Mickey	589823.00	
Outpatient		04-SEP-2003 07:40:39
Outpatient		08-SEP-2003 10:54:00
Observation		09-SEP-2003 03:35:32
Linden Jr., Kathy	589843.00	
Observation		05-SEP-2003 03:00:00
Blodgett, Karen Elizabeth	589883.00	
Inpatient		09-SEP-2003 04:15:00
Emergency		21-NOV-2003 06:14:00
Tyler, Allison	589983.00	
Inpatient		16-SEP-2003 11:16:00
Outpatient		26-NOV-2003 08:39:00
Christensen, Riley Allen	589984.00	
Inpatient		16-SEP-2003 11:23:00

68. Close the output display.

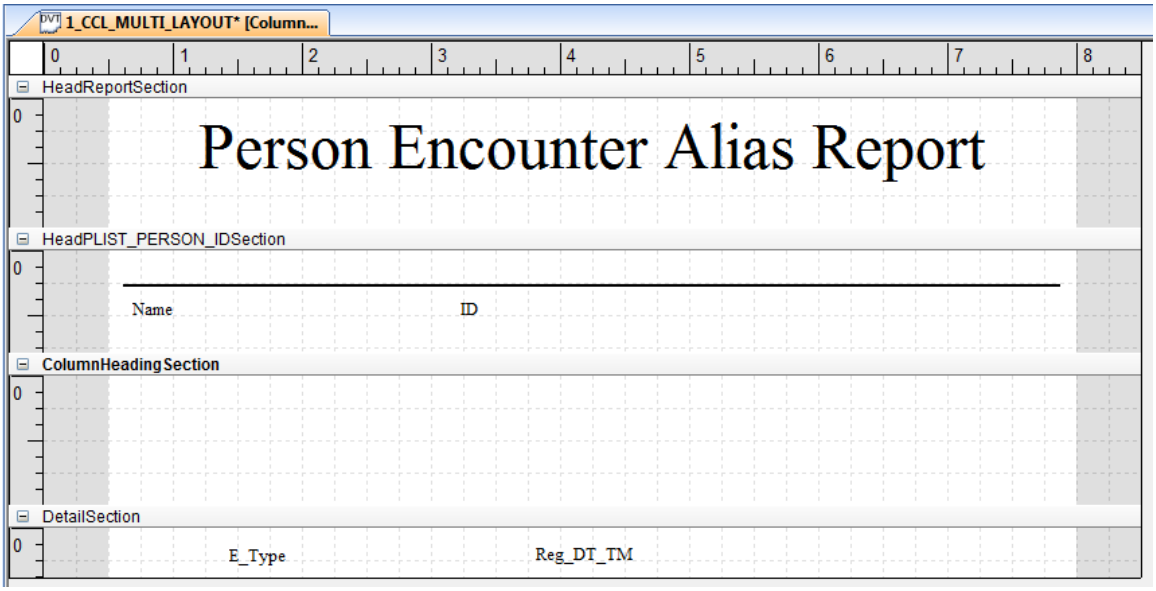
The appearance of the report would be enhanced if column headings were displayed above the encounter type and the registration date and time.

69. Click the HeadPLIST_PERSON_IDSection to ensure it is active.

70. From the Select/Modify Section of the Workspace dialog box, select the Head PLIST_PERSON_ID Reportwriter section (click the label, not the check box).

71. Click the New Layout Section button  . A new layout section called HeadPLIST_PERSON_IDSection1 is created and associated to the Head PLIST_PERSON_ID reportwriter section.

72. In the Properties dialog box, modify the name of the new section to **ColumnHeadingSection**. Your layout should be similar to the following:

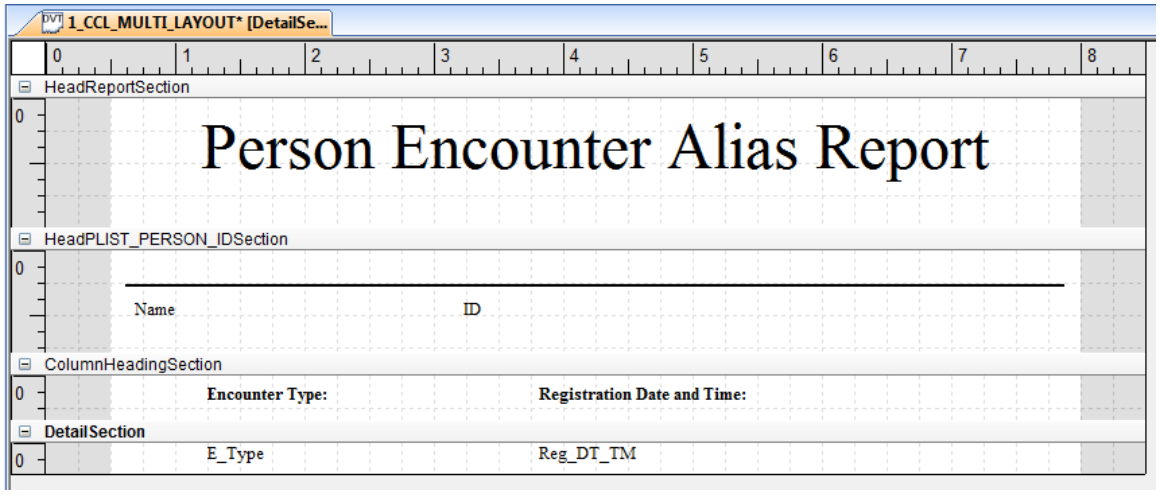


73. Use the Label Tool  to add Encounter Type: to the Source property to use as a column heading.

74. Use the Label Tool  to add Registration Date and Time: to the Source property to use as a column heading.

75. Use the formatting toolbar to bold both of the labels you have added.

76. Use the vertical resize to move the bottom of the DetailSection up to just underneath the items you just added. Your layout should look similar to the following example:



We are now ready to display the aliases for each person. The Get_Data query that you built using the Query Builder creates select expressions using an internal Discern Explorer process to traverse the Level 1 PLIST and the Level 2 ELIST. We need to create a process that will traverse the Level 2 ALIST each time we get a new person and

display that person's aliases on the layout. One way to accomplish this is to add a layout section that is associated with the Head PLIST_PERSON_ID reportwriter section. Code segments can be used to determine the number of aliases that exist in the ALIST for the person and execute a For Loop to traverse the ALIST and call the layout section to display all of the person's aliases.

77. Click the HeadPLIST_PERSON_IDSection to ensure it is active.

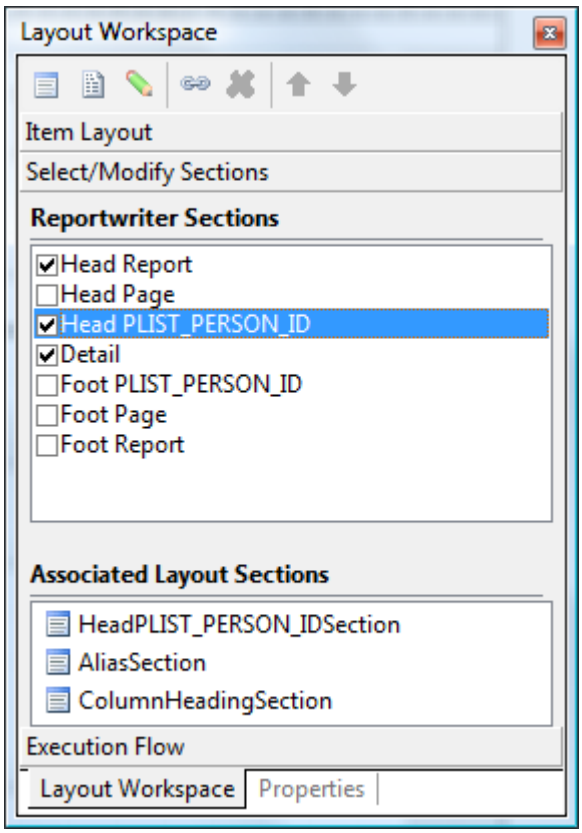
78. From the Select/Modify Section of the Workspace dialog box, select the Head PLIST_PERSON_ID Reportwriter section (click the label, not the check box)

79. Click the New Layout Section button  . A new layout section called HeadPLIST_PERSON_IDSection1 is created and associated to the Head PLIST_PERSON_ID reportwriter section.

80. In the Properties dialog box, modify the name of the new section to **AliasSection**.

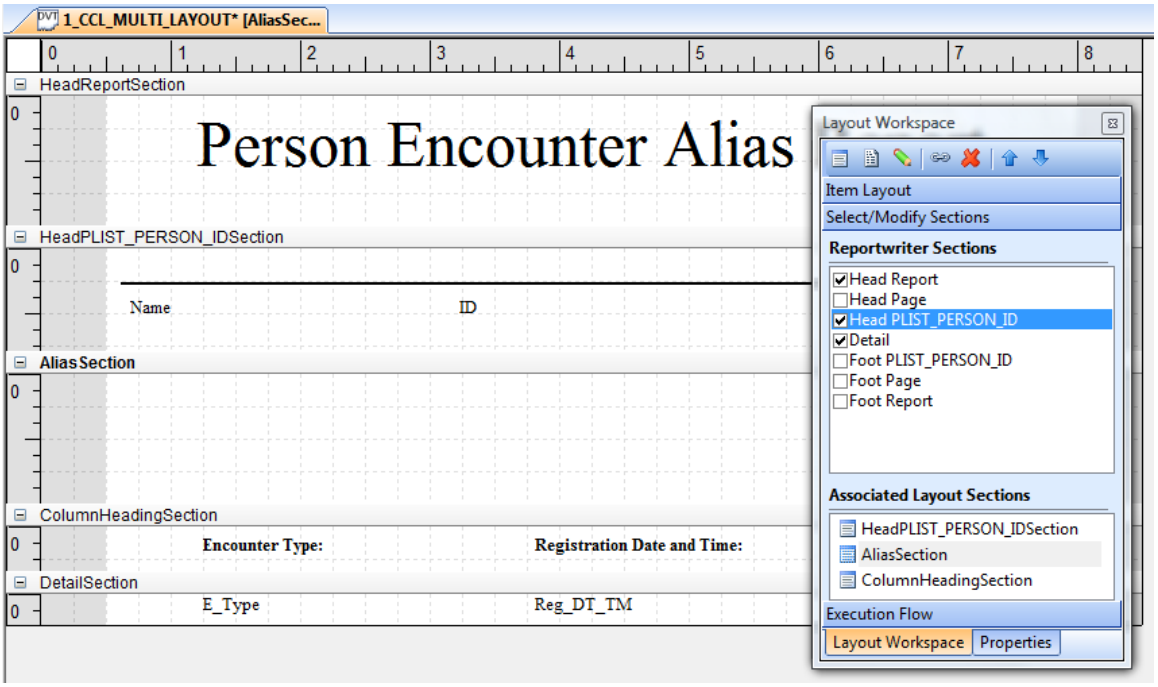
81. From the Select/Modify Section of the Workspace dialog box, select the Head PLIST_PERSON_ID Reportwriter section (click the label, not the check box).

82. Select the AliasSection from the Associated Layout Section and use the Move up button  to place the section in the middle. Your Associated Layout Section should be displayed similar to the following example:



Doc ID: A1001001A08G07B34433A94851

83. Move the AliasSection so that visually it is between the HeadPLIST_PERSON_ID and the ColumnHeadingSection. This can be done by placing your cursor on the ruler for the AliasSection, and then dragging and dropping it to the middle of the ruler for the ColumnHeadingSection. Your layout should look similar to the following example:

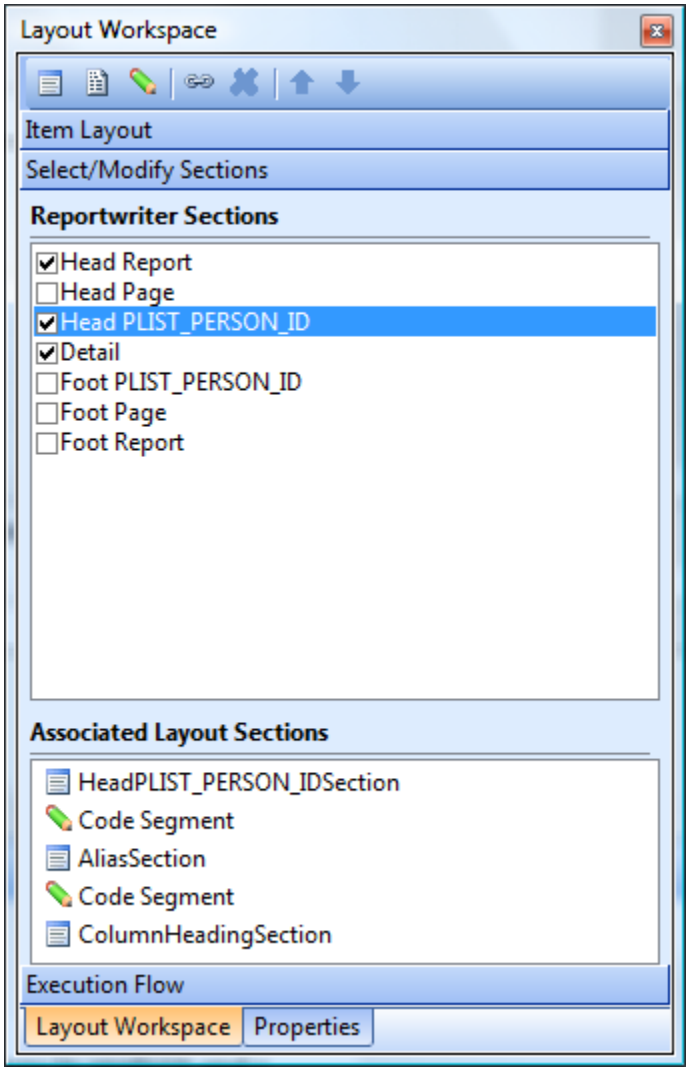


84. With the Head PLIST_PERSON_ID reportwriter section still selected, click the New Code Segment button on the Layout Workspace toolbar twice. Two new code segments will be inserted in to the Reportwriter Section.

85. Select the first Code Segment and place it underneath the HeadPLIST_PERSON_IDSection using the Move Up button.

86. Select the second Code Segment and place it underneath the AliasSection using the Move Up button. Your Associated Layout Section for the Head PLIST_PERSON_ID reportwriter section should resemble the following:

Doc ID: A1001001A08G07B34433A94851



Adding the layout sections and code segments to the Head PLIST_PERSON_ID reportwriter section causes each layout section and code segment to be executed in the order they appear in the Associated Layout Sections list. This is done each time a new person ID is encountered in the result set of the query that is associated with the layout.

87. Double-click the first Code Segment section you added above. The Layout Source [Code Segment] dialog box opens.

88. Enter the following command to set a variable named TotalAcnt equal to the number of items in the ALIST for each person in the PLIST:

```
totalacnt = size(PERSON_ENC_ALIAS->PLIST [d1.seq].ALIST ,5)
```

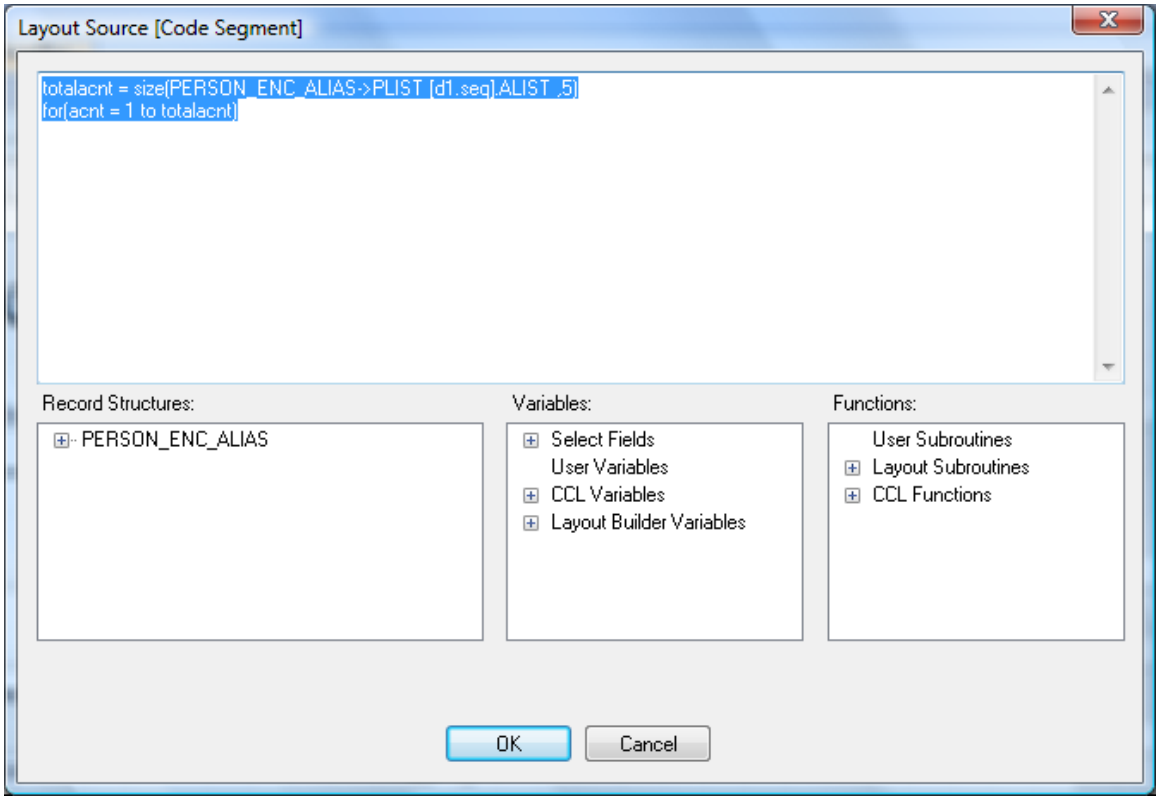
You can type the entire command, or build it by using a combination of typing and selecting the Size function from the CCL Functions list in the Functions column, expanding the Person_Enc_Alias record structure in the Record Structures: column, and then double-clicking the ALIST.

89. Enter the following command to start a For Loop that traverses all of the aliases in the ALIST for the current person:

Doc ID: A1001001A08G07B34433A94851

```
for(acnt = 1 to totalacnt)
```

Your Layout Source [Code Segment] dialog box should look similar to the following example:



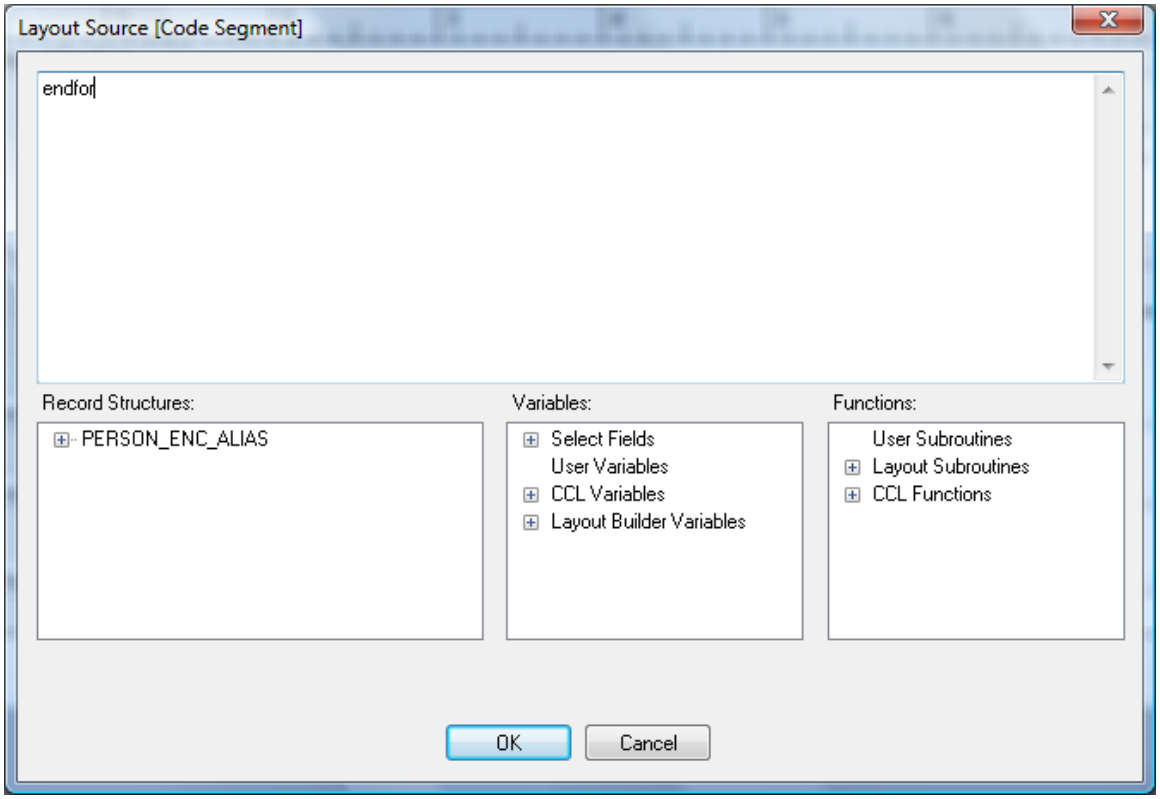
90. Click OK to close the Layout Source [Code Segment] dialog.

91. Double-click the second Code Segment section you added above. The Layout Source [Code Segment] dialog box opens.

92. Enter the following code to terminate the For Loop:

```
endfor
```

Your Layout Source [Head Plist_Person_ID] dialog box looks similar to the following example.



93. Click OK to close the Layout Source [Code Segment] dialog box.

94. Use the Text Tool  to add a field to the AliasSection to display each alias from the record structure.

95. In the Properties dialog box, modify the Name property to **Alias**.

96. In the Source box, click the ellipsis  button to open the Layout Source [Alias] dialog box.

97. In the Record Structures list expand the PERSON_ENC_ALIAS tree.

98. Expand the PLIST [*] tree.

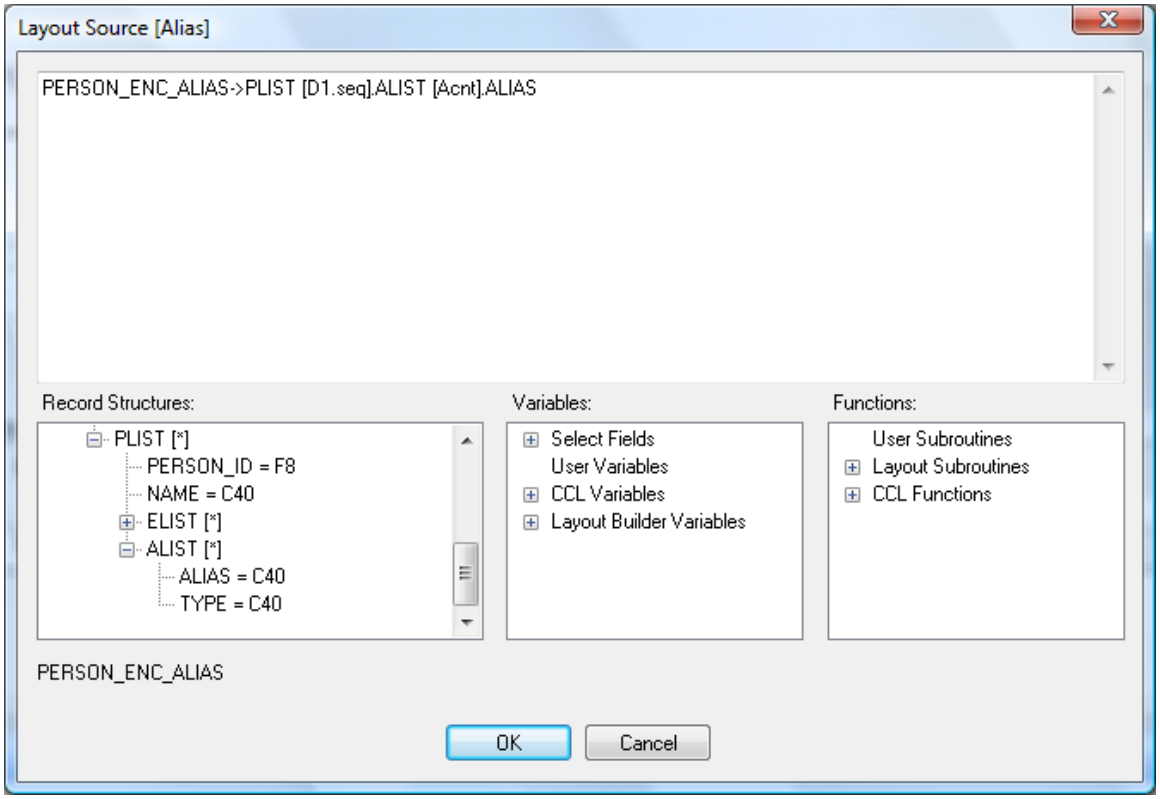
99. Expand the ALIST [*] tree.

100. Double-click Alias = C40 to add the following code as the layout source:

```
PERSON_ENC_ALIAS->PLIST[].ALIST[].ALIAS
```

101. Enter **D1.seq** as the subscript for the PLIST.

102. Enter **Acnt** as the subscript for the ALIST. You Layout Source now should look similar to the following example:



103. Click OK to close the Layout Source [Alias] dialog box.

104. Use the Text Tool  to add a field to the AliasSection to display each alias type from the record structure.

105. In the Properties dialog box, modify the Name property to **Alias_Type**.

106. In the Source box, click the ellipsis  button to open the Layout Source [Alias_Type] dialog box.

107. In the Record Structures list, expand the PERSON_ENC_ALIAS tree.

108. Expand the PLIST [*] tree.

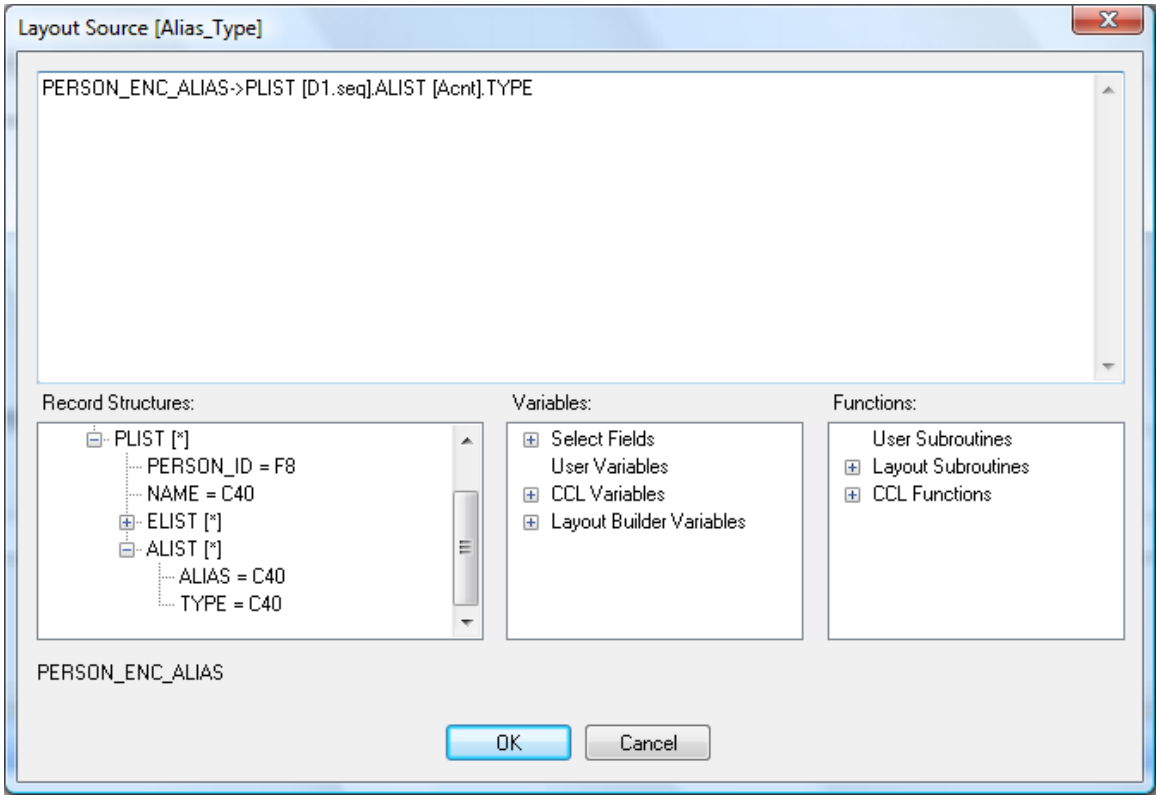
109. Expand the ALIST [*] tree.

110. Double-click Type = C40 to add the following code as the layout source:

```
PERSON_ENC_ALIAS->PLIST[].ALIST[].TYPE
```

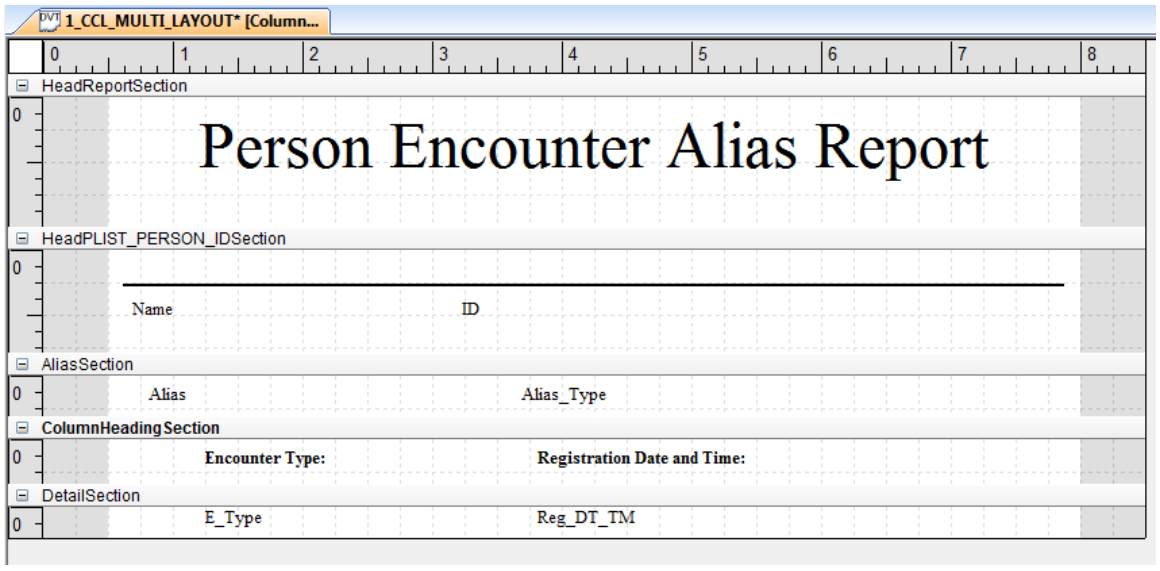
111. Enter **D1.seq** as the subscript for the PLIST.

112. Enter **Acnt** as the subscript for the ALIST. Your Layout Source now should look similar to the following example:



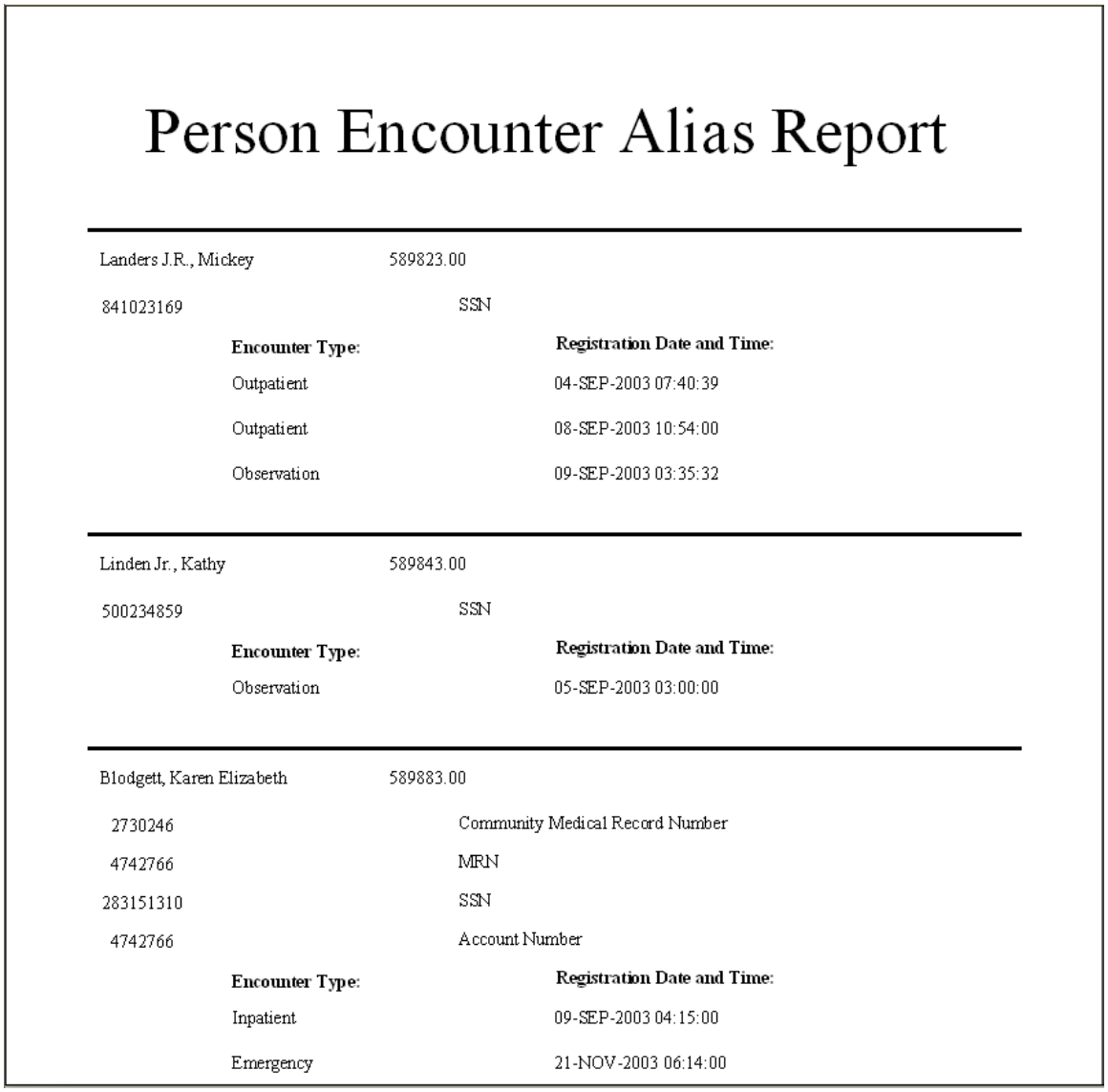
113. Click OK to close the Layout Source dialog box.

114. Use the vertical resize to move the bottom of the Column/HeadingSection up to just underneath the items you just added. Your layout should look similar to the following example:



115. Save the layout.

116. Execute the layout by selecting Run "1_*your_initials*_Multi_Layout_Layout" from the Build menu or clicking Run Prompt Program from the toolbar. Your output should look similar to the following example:



Person Encounter Alias Report

Landers J.R., Mickey 589823.00

841023169 SSN

Encounter Type:	Registration Date and Time:
Outpatient	04-SEP-2003 07:40:39
Outpatient	08-SEP-2003 10:54:00
Observation	09-SEP-2003 03:35:32

Linden Jr., Kathy 589843.00

500234859 SSN

Encounter Type:	Registration Date and Time:
Observation	05-SEP-2003 03:00:00

Blodgett, Karen Elizabeth 589883.00

2730246 Community Medical Record Number

4742766 MRN

283151310 SSN

4742766 Account Number

Encounter Type:	Registration Date and Time:
Inpatient	09-SEP-2003 04:15:00
Emergency	21-NOV-2003 06:14:00

Sub Reports

This section introduces Sub-Reports and provides an example using simple reports to demonstrate the Sub-Report functionality. Following the example step-by-step also provides extra benefit by guiding you to:

- Use one data driver for two different layout programs.

- Use commands to control the display of data on the left side and a graph to the right of a report.
- Flex report headers based on whether they are called as a sub-report or stand-alone.
- Use commands to control the flow of the program.

This section assumes that you have already completed the sections Creating a Table View Layout and Converting Existing Programs to Layout Programs Using a Driver Program.

Sub-Reports are layout programs that can be executed as stand-alone programs or called from another layout program. Using Sub-Reports enables you to re-use and combine layout programs, and display information from multiple layout programs into one report executed by one parent program.

The flexibility and usability of layout programs is increased by the ability to call and display them from within another layout program. You may have multiple layout programs that contain information that you want to combine in to one report. Marking the layout programs as Sub-Reports enables them to be called and displayed in other layout programs.

Another common use of Sub-Reports is to execute a driver program to gather data and store it in a record structure. The information is then available for other sub-reports or sections in the layout program.

The following steps are covered to demonstrate the use of Sub-Reports:

1. Create a data driver program that stores general information about persons in a record structure. We will use the data in the record structure for two separate layout programs that will be marked as Sub-Reports. The naming convention to be used for the data driver program is **1_*your_initials*_PRSN_Data_Driver**.

2. Create a layout program that graphs the distribution of birthdates by month. The naming convention to be used for the first layout program is **1_*your_initials*_Graph_BD**.

3. Create a table view layout program to display information about persons on a grid. The naming convention to be used for the second layout program is **1_*your_initials*_Person_View**.

4. Create a parent program that calls the graph and table view programs to display the information in one report. The naming convention to be used for the parent layout program is **1_*your_initials*_Parent**.

Step 1: Creating the Driver Program

This driver program is used to retrieve general person information, such as names, birthdates and gender and stores them in to a record structure.

1. Using DVDev, from the File menu select New and then Program.

2. In the Program Name box, enter **1_*your_initials*_PRSN_Data_Driver** and click OK.

3. Copy the following commands into your file. Place the code after the Create command and before the key words *End Go*.

```
/* The record structure created in this driver will be used by more than
one layout program. Ultimately a parent program will call both of the
Sub-reports that use this record structure. At that time, the record
structure will not need to be defined and loaded multiple times. The
following IF/ENDIF statement will determine if the record structure is
defined and loaded. If it is defined and loaded, the control will return
to the parent and not re-execute the query. */

if(validate(prsn_info)) ;if the record structure has scope
    if(size(prsn_info->p_list,5)); if it has a defined size
        return ;return control to the parent
    endif
else
   record PRSN_INFO (
   1 P_LIST[*]
     2 personid = f8
     2 personname = vc
     2 birthdate = dq8
     2 sex_code = f8
) with PersistScript
Endif

declare gender = f8 with noconstant(0.0),protect

/* the following commands checks the data type of the second parameter
and sets a global variable to the value of the second parameter. */

if(reflect(parameter(2,0))="F8")
   set gender = parameter(2,0)
endif

/* If a gender is chosen at the prompt of the layout program, then the
gender is used in the qualification. If the Any(*) option is chosen at
the prompt, then the qualification qualifies all genders*/

select into "nl:"
   p.person_id
  , p.name_full_formatted
  , p.birth_dt_tm
  , p.sex_cd
from person p
  where p.birth_dt_tm is not null
  and p.person_id > 0
  and p.active_ind = 1
  and p.sex_cd = gender
     or (gender = 0.0 and p.sex_cd > 0.0)

head report
cnt = 0
stat = alterlist(PRSN_INFO->P_LIST,100)
detail
  cnt = cnt +1
  if(mod(cnt,10)=1 and cnt > 100)
 stat = alterlist(PRSN_INFO->P_LIST,cnt+9)
  endif

  PRSN_INFO->P_LIST[cnt].personid = p.person_id
  PRSN_INFO->P_LIST[cnt].personname  = p.name_full_formatted
  PRSN_INFO->P_LIST[cnt].birthdate  = p.birth_dt_tm
  PRSN_INFO->P_LIST[cnt].sex_code = p.sex_cd
foot report
  stat = alterlist(PRSN_INFO->P_LIST,cnt)

with maxrec=200
```

```
call echorecord(PRSN_INFO)
```

4. At the end of the file, enter **1_*your_initials*_PRSN_Data_Driver go** after the key words *End Go*.

5. Save and compile the source code file.

6. Press Ctrl+L to access the listing and verify there is data loaded in the record structure.

```
>>>Begin EchoRecord PRSN_INFO   ;PRSN_INFO
1 P_LIST[1,100*]
  2 PERSONID=F8   {589763.0000000000                              }
  2 PERSONNAME=VC0   {Spencer,Ike}
  2 BIRTHDATE=DQ8   {62859266600000000       ,year(1999) mon(3) day(9)
time(18:44:20.00) utc(0)}
  2 SEX_CODE=F8   {362.0000000000                                 }
 1 P_LIST [2,100*]
  2 PERSONID=F8   {589763.0000000000                            }
  2 PERSONNAME=VC18   {Flanigan, Erin}
  2 BIRTHDATE=DQ8   {63629280000000000       ,year(2001) mon(8) day(17)
time(00:00:00.00) utc(0)}
  2 SEX_CODE=F8   {363.0000000000                                 }
 1 P_LIST [3,100*]
  2 PERSONID=F8   {589824.0000000000                            }
  2 PERSONNAME=VC15   {Taney, Maria}
  2 BIRTHDATE=DQ8   {63629280000000000       ,year(2001) mon(8) day(17)
time(00:00:00.00) utc(0)}
  2 SEX_CODE=F8   {364.0000000000                               }
 1 P_LIST [4,100*]
  2 PERSONID=F8   {589846.0000000000                            }
  2 PERSONNAME=VC16   {Stein, Jordon}
  2 BIRTHDATE=DQ8   {52864704000000000       ,year(1967) mon(7) day(8)
time(00:00:00.00) utc(0)}
  2 SEX_CODE=F8   {363.0000000000                               }
```

If you do not have data loaded in the record structure, look for any error messages that may need to be corrected, or validate that the query generated data. Once you can validate that the record structure is loaded, the driver program is completed.

The next step is to create a new layout program to call the driver program.

Step 2: Creating Layout Program for Sub-Report 1

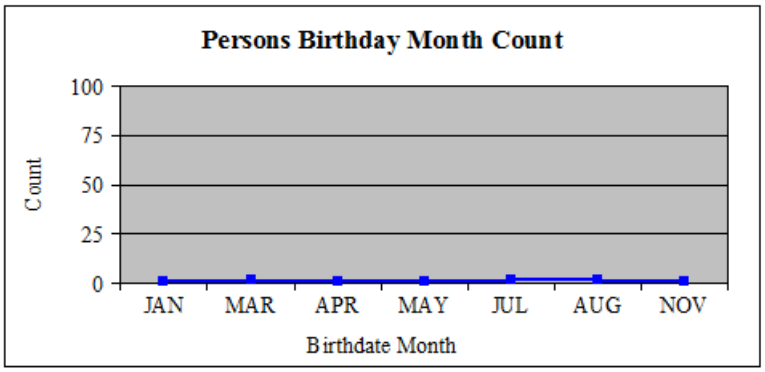
We will create a layout program to calculate the number of birthdays that fall within a month for a specific gender, and display that information in a list and also in a graph. The final output will be a report formatted like the following example:

Birthdate Distribution by Month

Page: 1 of 2

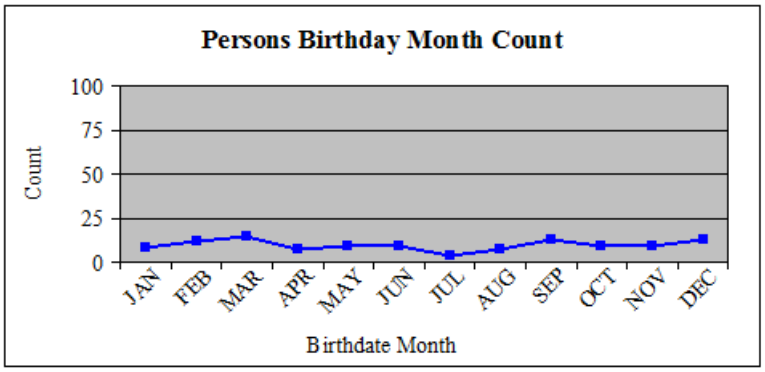
N/A

JAN	1.00
MAR	2.00
APR	1.00
MAY	1.00
JUL	2.00
AUG	2.00
NOV	1.00



Female

JAN	8.00
FEB	12.00
MAR	15.00
APR	7.00
MAY	9.00
JUN	9.00
JUL	4.00
AUG	7.00
SEP	13.00
OCT	9.00
NOV	9.00
DEC	13.00



1. From the File menu, select New.
2. From the File Type list, select Layout Program.
3. In the Program Name box, enter **1_your_initials_BD_Graph** and click OK.
4. On the Report Properties dialog box, click Finish. A layout with a single section named DetailSection is created.
5. From the Tools menu, select Prompt Builder.
6. Click Add to add a second prompt to ask for a gender from the user.
7. On the General tab, set the options as follows:

   Prompt Display: Select a specific gender or Any(*) for all types

   Prompt Name: GEN

Control Type: Code Set

Prompt Type: Expression

8. On the Code Set tab, set the options as follows:

   - Enter **57** for the Code Set
   - Check the Include Any(*) option
   - Click the Define Any(*) box and change the Primary key value to **0.0** and click OK.
   - Click the Set Default option and highlight the Any(*) option and save the prompt form.

   Set the Layout Driver that will be used to populate the record structure.

9. From the Tools menu, select Set Layout Driver and enter **1_your_initials_PRSN_Data_Driver** in the Name box.

If your 1_*your_initials*_PRSN_Data_Driver program is a cclgroup1 object, you will need to append **:group1** to the program name when you enter it in the Name box. Use CCLPROT to determine if your program is a cclgroup1 or cclgroup0 (DBA) object.

10. In the Parameters box enter **$Outdev, $GEN** and click OK to close the Layout Driver dialog box.

When this program is run, it executes the driver program which loads data into a record structure. In order for this layout program to format the data from the record structure, the record structure needs to be recognized by the layout program and in the driver program. In this example, the record structure is defined in the data driver program with the scoping option of PersistScript. The PersistScript scoping option allows a child process to declare a public variable and make it known to the parent process. Using this option allows the record structure to be known to the layout program.

We need to access the values in the record structure list. A simple way to access the values in the record structure list is to use a query to assign the values in the list to select expressions.

11. From the Tools menu, select Query Builder and then click Add.

12. In the Query Name box, enter **Get_Person_Data,** select the Associate Layout option and click OK.

13. On the Tables tab expand the Record Structures tree and select the User Defined category. The PRSN_INFO record structure is displayed in the Tables column.

14. Double-click the PRSN_INFO record structure to move it to the Selected Tables column.

15. Click the Fields tab, expand the P_list tree, and double-click each of the items to add them to the Selected Fields list. Double-clicking a record structure list item adds the

DUMMYT table to the Tables tab and creates expressions in the Selected Fields column using *dummyt_alias*.seq to reference each position of the record structure list. Creating an expression for each item in a record structure makes it easier to reference the items in your layout.

Now we will create a new expression to return an integer value representing the month of the birth date for a person by using the MONTH( ) function. This expression will be used as a sub sort so that our data will be logically grouped by gender and then sequenced by their birth month.

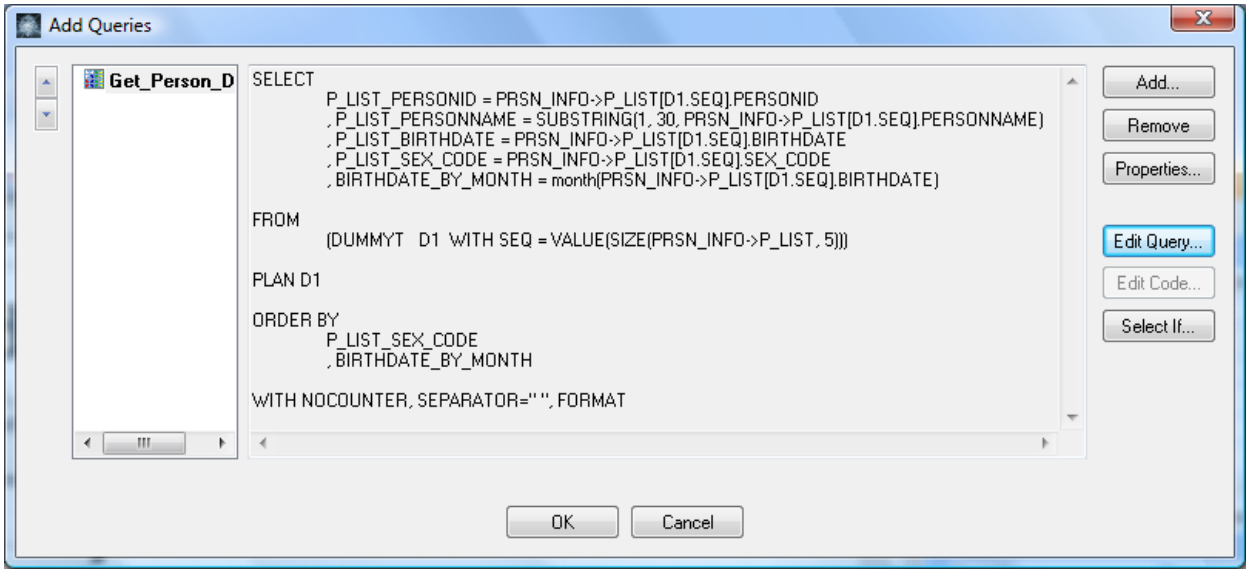
16. Click Add Expr and create a new expression. Copy and paste the following command:

    BIRTHDATE_BY_MONTH = month(PRSN_INFO->P_LIST[D1.SEQ].BIRTHDATE)

17. Click OK to close the Add Expr dialog box and click the Sort tab.

18. Select P_LIST_SEX_CODE for the primary sort. Then sub-sort by BIRTHDATE_BY_MONTH.

19. Click Close. The Add Queries dialog box opens and displays the query similar to the following example:



20. Click OK to close the Add Queries dialog box.

21. Check the Head Report Reportwriter Section checkbox and click Yes to add the HeadreportSection layout section.

22. Use the Label tool to add a report title, **Birthdate Distribution by Month,** to the HeadReportSection.

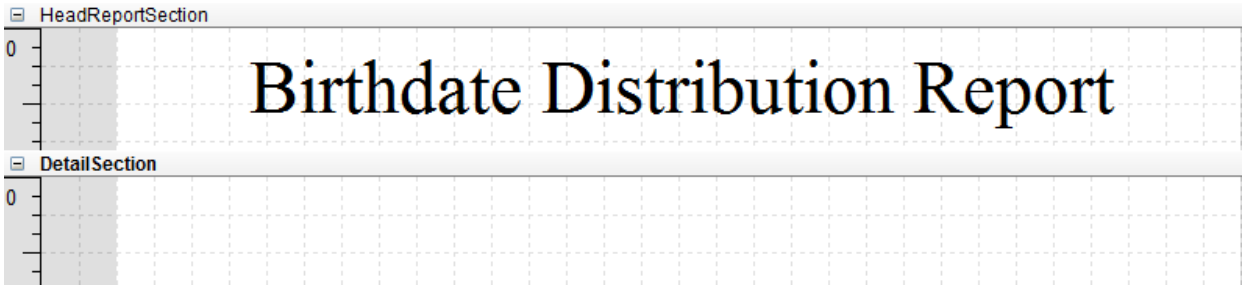
23. Use the format toolbar to set the following:

    - The font to Times
    - The font size to 36

- Center the text

24. Use the vertical resize to drag the DetailSection bar close to the title text.

    Your layout should resemble the following example:



25. Click the Head Page Reportwriter Section checkbox and click Yes to the message to add the HeadpageSection layout section.

26. Use the Text tool and add a field to top left side of the section to display the page number and the number of pages.

27. Double-click on the text field and enter **RPT_PageOfPage**.

28. Select the Properties for the item and enter **Page_Of_Page** for the Name property and select Yes for the Font Bold property.

29. Use the vertical resize to drag the DetailSection up to the bottom of the field you just added.

30. From the Layout Workspace check the Head P_List_Sex_Code Reportwriter Section checkbox and click Yes to add the HeadP_List_Sex_Code layout section.

31. Select the Properties for the section and change the Name property to **GenderSection**.

32. Use the Line tool to draw a line horizontally across the very top of the section and change the Pen Size property to .025.

33. Use the Text tool to add a field that is under the line to the top left hand side of the section to the display the gender.

34. Double-click on the text field and copy and paste the following command:

    EVALUATE(P_list_sex_code,0.0,"N/A",uar_get_code_display(P_LIST_SEX_CODE))

35. Select the Properties for the item and change the Name property to **Gender**, the Font Bold to Yes and the Font Color to Blue.

36. Use the vertical resize to drag the DetailSection up to the bottom of the field you just added.

    We need to add logic to the program to determine if there is enough room on the page to display the information in the GenderSection which consists of the line, the

gender and the 12 months of the year, all displayed vertically down the page. Here is an example of the GenderSection:

If there is not enough room to display the line, gender and 12 months, logic must be logic added to the program to create a page break and the section displayed on the next page. The commands we will use to determine if there is enough space on the page needs to happen before the GenderSection executes.

37. From the Layout Workspace dialog, add a Code Segment above the GenderSection. Copy and paste the following commands:

```
; calculate section height
/* determines the total height needed to draw the line,
the gender and also room for the 12 months of the year
that will list vertically down the page*/

_fDrawHeight = GenderSection(RPT_CALCHEIGHT)*14

; break if bottom of page exceeded
/* If there is not enough room to draw the entire section
before the end of the page, then issue the break command to
start the section on the next page*/

if (_YOffset+_fDrawHeight>_fEndDetail)
 break
endif
```

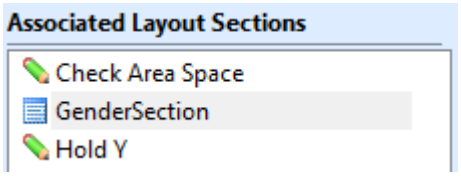
38. Click on the Properties for the Code Segment and change the Name property to **Check Area Space**.

    We need to add commands after the GenderSection has rendered to hold off where the Y value is, so that it can be used later on to properly place the graph to right of the text.

39. Add a code segment below the GenderSection. Copy and paste the following commands:

```
/* The variable will hold the value of the Y offset to be used to place
the graph properly */

_HoldMyY = _YOffset
```

40. Click the Properties for the Code Segment and change the Name property to **HOLD Y**. Your Associated Layout Sections for the Head P_List_Sex_Code Reportwriter section should resemble the following example:



    In this layout program, we do not need the DetailSection to display any items, so it can be deleted.

41. Click in the DetailSection and from the Edit menu select Remove > Section. Click Yes to remove the section.

42. Click the Foot Birthdate_By_Month option and click Yes to add the FootBirthdate_By_MonthSection.

43. Change the Name property to **CountSection**.

44. Use the Text tool to add a field to the top left-hand side of the section to display the birth month.

45. Double-click the text field and copy and paste the following command to be used as the source:

    FORMAT(P_LIST_BIRTHDATE,"MMM;;Q")

46. Select the Properties for the item and change the Name property to **BirthMonth**.

47. From the LayoutWorkspace dialog, add a new Code Segment above the CountSection and add the following command:

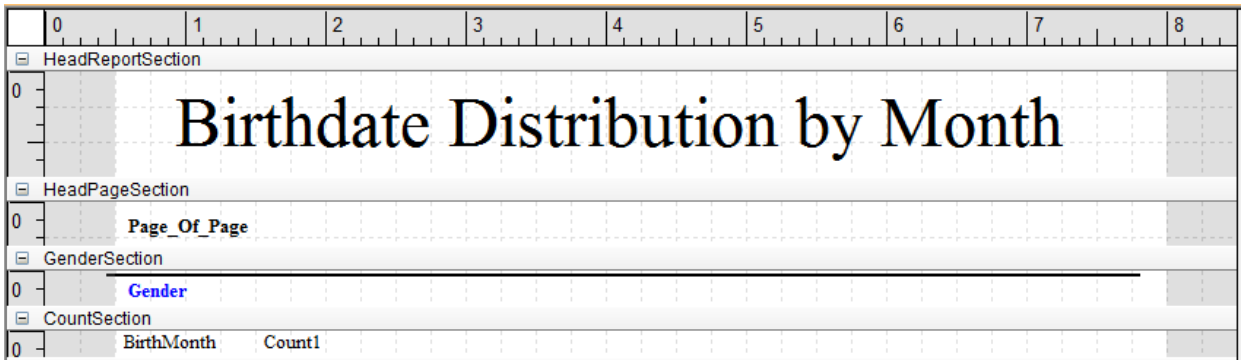
    MyCount = count(P_LIST_BIRTHDATE)

48. Use the Text tool to add a field to the right of the BirthMonth to display the number of birthdates calculated for each month.

49. Double-click on the text field and enter **MyCount**.

50. Select the Properties for the item and change the Name property to **Count1**.

51. Use vertical re-size to drag the bottom of the section to the bottom of the fields you just added. Your layout should display similar to the following example:



52. Press Ctrl+F5 to execute your layout program. Click Yes when prompted to save the layout. The prompt form opens. Select Any(*) for the second parameter and click Execute. Your output should display similar to the following format:

Birthdate Distribution by Month

Page: 1 of 2

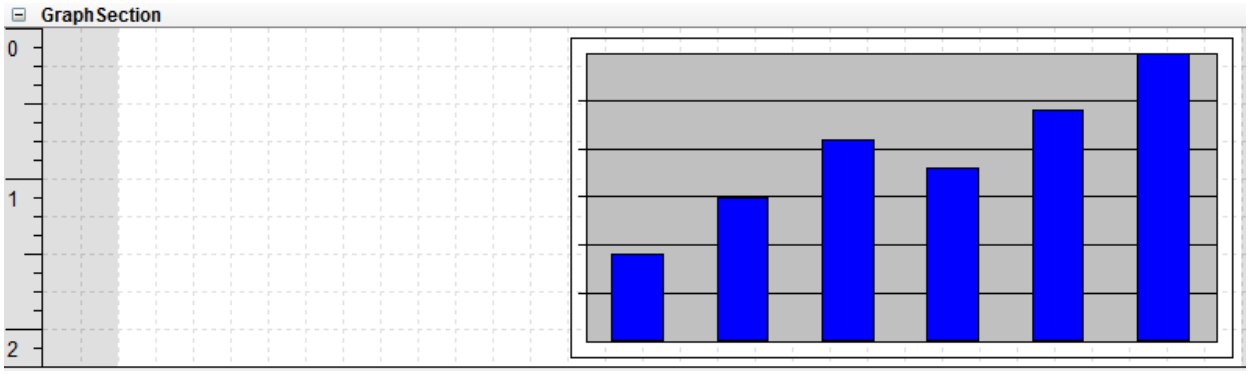
N/A	
JAN	1.00
MAR	2.00
APR	1.00
MAY	1.00
JUL	2.00
AUG	2.00
NOV	1.00
Female	
JAN	8.00
FEB	12.00
MAR	15.00
APR	7.00
MAY	9.00
JUN	9.00
JUL	4.00
AUG	7.00
SEP	13.00
OCT	9.00
NOV	9.00
DEC	13.00

To finish this report, add a graph to show the number of birthdays in each month per gender that displays to the right of the months listed vertically.

53. From the Layout Workspace select the Foot P_List_Sex_Code option and click Yes to the message to add a layout section.

54. Select the Properties for the section and change the Name property to **GraphSection** and the Height property to **2.25**.

55. Using the Graph tool, create a graph that covers all of the right side of the Graph Section. Your GraphSection should look like the following example:



56. Double-Click in the Graph and change the Graph Type to Line.

57. Click the Data Source tab and from the Select Query list choose Get_Person_Data.

58. Select BIRTHDAY_BY_MONTH for the Axis Field.

59. Enter **FORMAT(P_LIST_BIRTHDATE,"MMM;;Q")** as the Axis Label.

60. Click the Series tab and enter **Count** as the Series Name.

61. Enter **count(P_LIST_BIRTHDATE)** as the Data Values.

62. Click the Data Range tab and enter **0** for the Minimum, **100** for the Maximum and **25** for the Index.

63. Click the Titles tab. Enter **Persons Birthday Month Count** as the Chart Title**.** Enter **Birthdate Month** as the (X) Axis Title. Enter **Count** as the (Y) Axis Title.

64. Click OK to exit the graph properties dialog box.

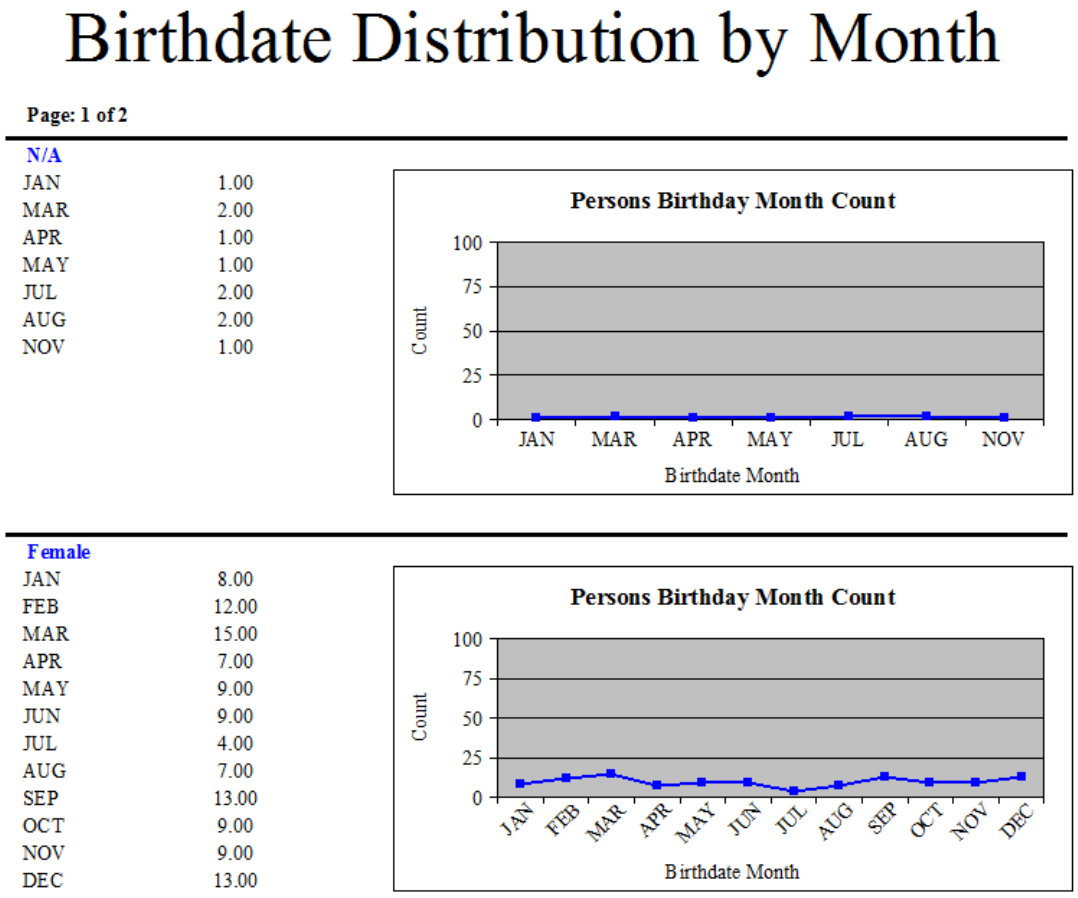
65. From the Workspace dialog, click on the Foot P_LIST_SEX_CODE reportwriter section (click the text not the box) and add a Code Segment above the GraphSection. Copy and paste the following command:

```
;resets the Y position to position the graph in relation to the text data
_YOffset = _HoldMyY
```

66. Add a Code Segment below the GraphSection to reset the Y position. Copy and paste the following commands:

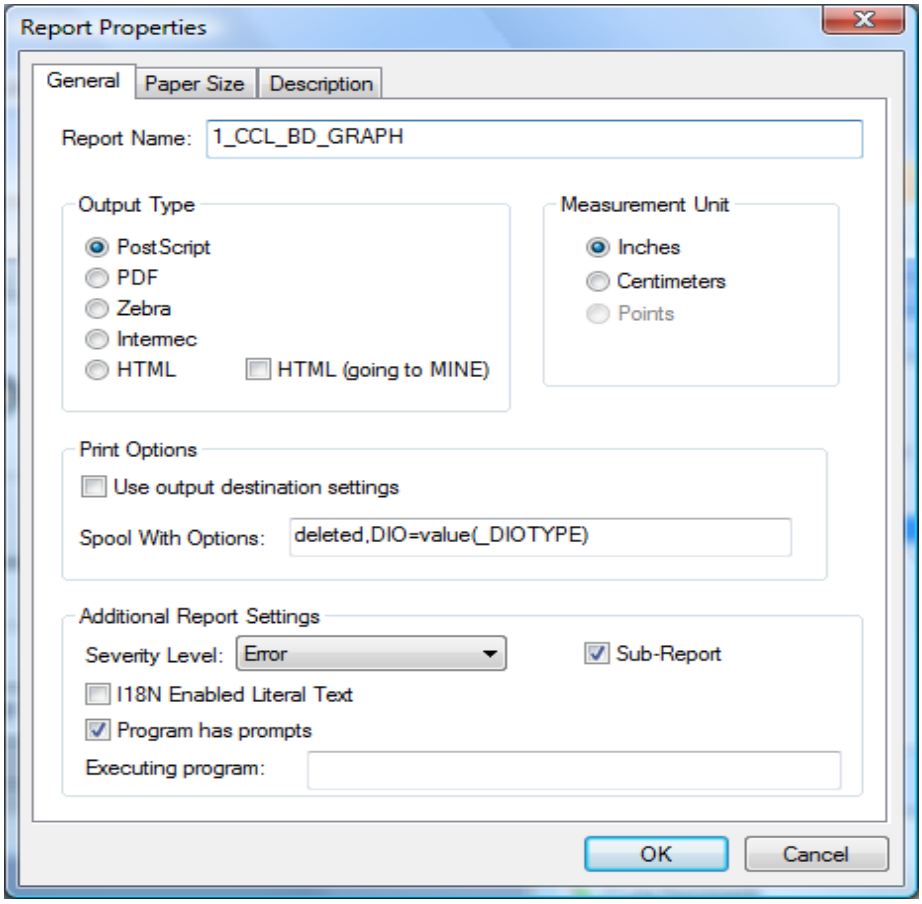
```
_YOffset = _YOffset + GenderSection(Rpt_CalcHeight)
```

67. Press Ctrl+F5 to execute your layout program. Click Yes when prompted to save the layout. The prompt form opens. Select Any(*) for the second parameter and click Execute. Your output should display similar to the following example:



This report runs as a stand-alone program, however, we also want to call it from another layout program as a Sub-Report. It needs to be made a Sub-Report by changing its Report Properties.

68. From the Edit menu select Report Properties. Select the Sub-Report option located in the Additional Report Writer Settings.



The sub-report option places code in the program that is used to detect if the program is being called or used as a sub-report from another layout program.

69. Save the file.

You have completed one of the Sub-Reports and will now create one more.

Step 3: Creating the Table View Program – Sub-Report 2

This next layout program will call the same data driver as 1_*your_initials*_BD_ Graph and display the detailed information about the persons stored in the record structure using a table view. The final output will be a report formatted like the following example:

	Person ID	Name	Birth Date	Gender
1	593694	Smith, Trina	01/01/01 00:00:00	Female
2	594976	Tremmel, Sue	07/29/23 00:00:00	Female
3	593758	Lee, Khim	03/10/28 00:00:00	Female
4	595361	Baker, Ginger	10/03/28 00:00:00	Female
5	593747	Limshue, Katia	02/17/30 00:00:00	Female
6	595125	Lou, Marta	07/08/30 00:00:00	Female
7	594102	Pill, Jillene	11/10/33 00:00:00	Female
8	593927	Boban, Leanne	06/18/35 00:00:00	Female
9	595232	Imara, Lauris	07/09/35 00:00:00	Female
10	593924	Dock, Alison	10/17/35 00:00:00	Female
11	594274	Teach, Kathleen	09/15/36 00:00:00	Female
12	594937	Linn, Kerri	08/21/37 00:00:00	Female
13	595112	Friends, Colleen	05/05/38 00:00:00	Female

1. In DVDev, from the File menu select New.
2. From the File Type list, select Layout Program.
3. In the Program Name box, enter **1_your_initials_Person_View** and click OK.
4. From the Report Layout options, select the Table View option. Enter **5** for the number of column options.
5. From the Tools menu, select Prompt Builder.

   Copy the prompts from the 1_*your_initials*_BD_Graph layout program to 1_*your_initials*_Person_View.
6. Select Transfer objects from the Tools menu and enter **1_*your_initials*_BD_Graph**.
7. Click Browse to find the program.
8. From Task menu, select Copy Object. Enter **1_*your_initials*_Person_View** and click OK.

   The same data driver program used to populate the graph in 1_*your_initials*_BD_Graph will also be used to populate this table view.
9. From the Tools menu, select Set Layout Driver. The Layout Driver dialog box is opened.
10. In the Name box, enter **1_*your_initials*_PRSN_Data_Driver**.

    If your 1_*your_initials*_PRSN_Data_Driver program, is a cclgroup1 object, you will need to append **:group1** to the program name when you enter it in the Name box. Use CCLPROT to determine if your program is a cclgroup1 or cclgroup0 (DBA) object.
11. In the Parameters box enter **$Outdev, $GEN** and click OK to close the Layout Driver dialog box.

We need to access each item in the record structure list, which can be done by using a query to assign the values in the list to select expressions.

12. From the Tools menu, select Query Builder and click Add.

13. In the Query Name box, enter **Get_Person_Data**, select the Associate Layout option and click OK.

14. On the Tables tab, expand the Record Structures tree and select the User Defined category. The PRSN_INFO record structure is displayed in the Tables column.

15. Double-click the PRSN_INFO record structure to move it to the Selected Tables column.

16. Click the Fields tab, expand the P_list tree, and double-click each of the items to add them to the Selected Fields list.

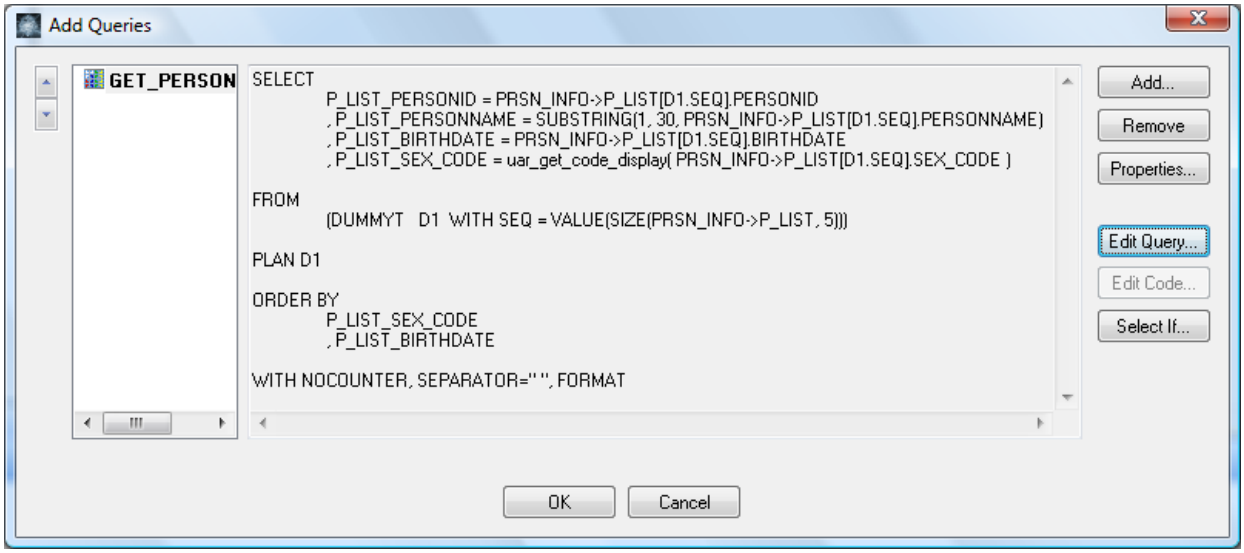
17. In the Selected Fields list, edit the expression created for the PRSN_INFO->P_LIST[D1.SEQ].SEX_CODE by adding **UAR_GET_CODE_DISPLAY( )** to the item:

    Your expression should look similar to the following example:

    P_LIST_SEX_CODE = uar_get_code_display(PRSN_INFO->P_LIST[D1.SEQ].SEX_CODE)

18. From the SORT tab, select P_LIST_SEX_CODE and then P_LIST_BIRTHDATE.

19. Click Close. Your query should look similar to the following example:



20. Click OK to close the Add Queries dialog box.

The HeadReportRow will be used to display column headers. The DetailRow will be used to display the actual ID and name of each person returned by the query. The FootReportRow will be used to show a total count of how many persons are listed. Currently, these rows are divided into cells of equal size. The first cell will be used to

display a sequence number and the second cell will be used to display a person's ID. These cells should be made smaller to accommodate the data.

21. With no cell selected, move the pointer over the right vertical edge of the first cell. When the pointer changes to the horizontal resize ,click and drag the vertical edge to the left to make all of the cells in the first column about a half inch in width.

22. Click in the second cell of the HeadReportRow. Click again and enter **"Person ID"** as the text. Remember that quotes must be placed around any text item.

23. Modify the Font Bold property to Yes.

24. Deselect the cell and change the vertical edge of all of the cells in the second column to three quarters inch in width.

25. Click the third cell of the HeadReportRow. Wait a moment, click again, and enter **"Name"** as the text.

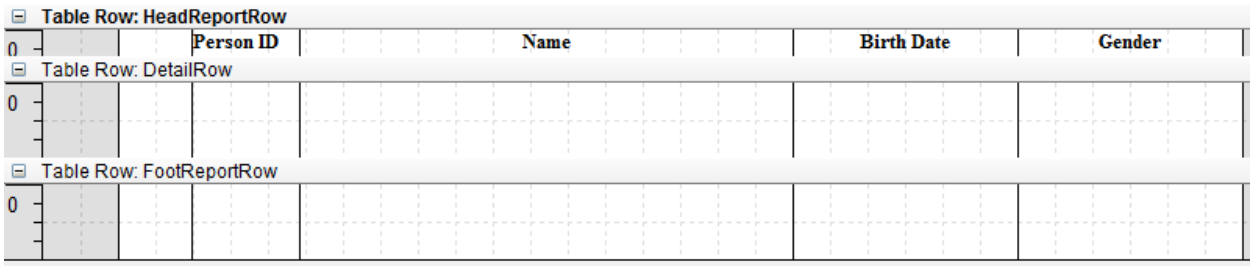
26. Modify the Font Bold to Yes and the Align Horizontal property to Center.

27. Click the fourth cell of the HeadReportRow. Wait a moment, click again, and enter **"Birth Date"** as the text.

28. Modify the Font Bold to Yes and the Align Horizontal property to Center.

29. Click the fifth cell of the HeadReportRow. Wait a moment, click again, and enter **"Gender"** as the text.

30. Use the vertical resize to click and drag the bottom of the HeadReportRow up to just below the items you added above. Your table should look similar to the following example:



The first cell of the DetailRow will be used to display the row number. Initialize a variable in the HeadReportRow that will be incremented and used to display the row number.

31. From the Layout Workspace click the Head Report reportwriter section (the text not the checkbox) and add a Code Segment below the HeadReportRow and enter the following command:

    CNT = 0

32. Click the Detail Section (the text, not the checkbox) and add a Code Segment above the DetailRow and enter the following command:

CNT = CNT + 1

33. Click in the first cell of the DetailRow. Click again and enter **CNT**.

34. From the Drop Items dialog box, drag the following items to the DetailRow:

    - P_LIST_PERSONID to the second cell
    - P_LIST_PERSONNAME to the third cell
    - P_LIST_BIRTHDATE to the fourth cell
    - P_LIST_SEXCODE to the fifth cell

35. Access the Properties for P_LIST_PERSONID and wrap the CNVTINT() function around the item so that it looks similar to the following example: **cnvtint(P_LIST_PERSONID)**. Using the function converts the data to an integer data type and keeps trailing zeros from displaying with the person ID.

36. Change the Align Horizontal property to Right.

37. Access the Properties for P_LIST_BIRTHDATE and wrap the FORMAT() function around the value so that it looks similar to the following example:

    format(P_LIST_BIRTHDATE, "@SHORTDATETIME")

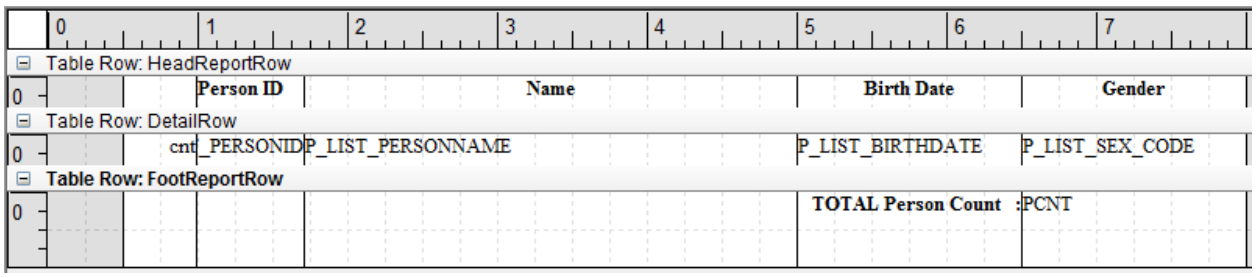
    Use vertical resize to click and drag the bottom of the DetailRow up to just below the items you have just added.

38. Click the fourth cell of the FootReportRow. Click again and enter **"TOTAL Person Count"** as the text.

39. Modify the Font Bold property to Yes and Align Horizontal property to Right.

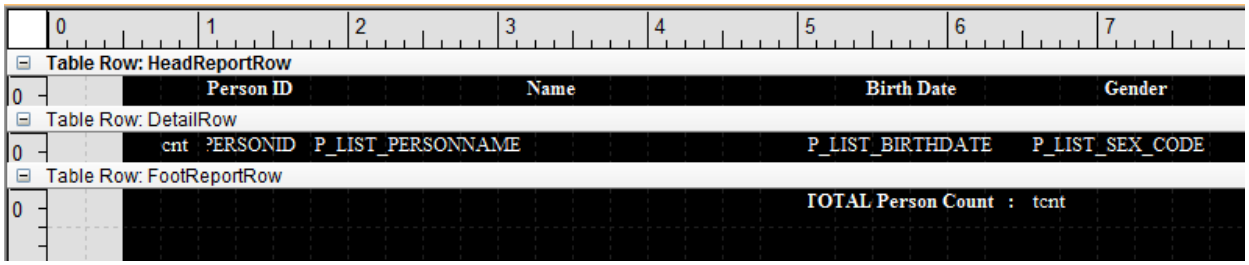
40. Click the fifth cell of the FootReportRow. Click again and enter **CNT.**

41. Change the Name property to **PCNT**. Your table should look similar to the following example:



For display purposes, add extra spaces to each of the cells so that the data is not displayed directly next to a vertical edge.

42. Click on the box directly to the left of the Horizontal ruler. The entire table is highlighted.



If your ruler is not visible, select it from the View menu.

43. From the Properties, change Padding Left and Padding Right to Yes.

44. Change the Padding Size to **0.050**.

45. Press Ctrl+F5 to execute your layout program. Click Yes when prompted to save the layout. The prompt form opens. select Any(*) for the second parameter and click Execute. Your report should look similar to the following example:

	Person ID	Name	Birth Date	Gender
1	593694	Smith, Trina	01/01/01 00:00:00	Female
2	594976	Tremmel, Sue	07/29/23 00:00:00	Female
3	593758	Lee, Khim	03/10/28 00:00:00	Female
4	595361	Baker, Ginger	10/03/28 00:00:00	Female
5	593747	Limshue, Katia	02/17/30 00:00:00	Female
6	595125	Lou, Marta	07/08/30 00:00:00	Female
7	594102	Pill, Jillene	11/10/33 00:00:00	Female
8	593927	Boban, Leanne	06/18/35 00:00:00	Female
9	595232	Imara, Lauris	07/09/35 00:00:00	Female
10	593924	Dock, Alison	10/17/35 00:00:00	Female
11	594274	Teach, Kathleen	09/15/36 00:00:00	Female
12	594937	Linn, Kerri	08/21/37 00:00:00	Female
13	595112	Friends, Colleen	05/05/38 00:00:00	Female

46. Look at the last page of your report and verify that the total number of people are being calculated and displayed.

200	8773482	Revoir, Gene	12/12/23 00:00:00	Male
			TOTAL Person Count	200

47. Close the output display window.

This report runs as a stand-alone program, however, we also want to call it from another layout program as a Sub-Report. It needs to be made a Sub-Report by changing its Report Properties.

48. From the Edit menu, select Report Properties. Select the Sub-Report option located in the Additional Report Writer Settings.

49. Save the file.

We have created two layout programs that can be executed as stand-alone programs. These programs are independent of each other and allow the user to retrieve information from the specific program. However, we have also marked these programs as Sub-Reports so that they can be called from another layout program to combine the information into one report. To demonstrate how to combine the reports, we will create a parent program to call each of the programs and display their information on one report.

Step 4: Creating the Parent Layout Program

1. From the File menu, select New.

2. From the File Type List, select Layout Program.

3. In the Program Name box, enter **1_*your_initials*_Parent**, click OK and then click Finish.

   Copy the prompts from the 1_*your_initials*_Person_View layout program to 1_*your_initials*_Parent.

4. Select Transfer objects from the Tools menu and enter **1_*your_initials*_Person_View**.

5. Click Browse to find the program.

6. From Task menu, select Copy Object. Enter **1_*your_initials*_Parent** for the destination object name and click OK.

In this layout program, the DetailSection is not tied to a query and will only be executed one time. We can use this section to create the title for our report.

7. In the DetailSection, use the Label tool to add a report title, **Person Data - Birthdate Distribution Report**.

8. Use the format toolbar to set the following example:

   - The font to Times
   - The font size to 24
   - Center the text

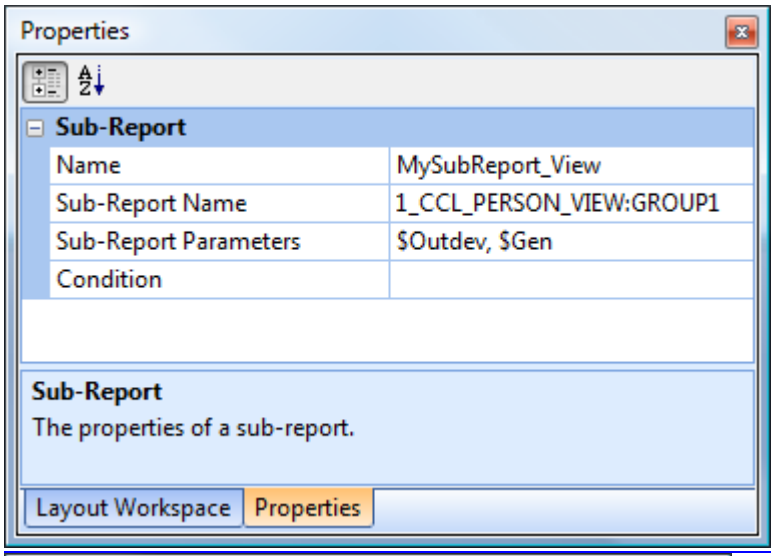
9. From the Edit > Insert menu option, select Sub-Report. A new sub-report section named Sub-Report: MySubReport(0) is created.

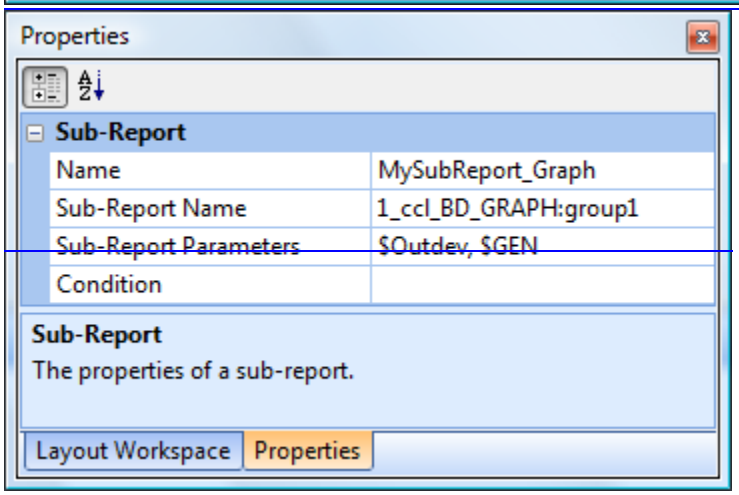
10. Modify the Name property to **MySubReport_View**.

11. Modify the Sub-Report Name property to the program containing the table view, 1_*your_initials*_Person_View. Append **:group1** to the object name if necessary.

12. Enter **$Outdev, $GEN** for the Sub-Report Parameters property.

The properties for the Sub-Report section should look like the following example:





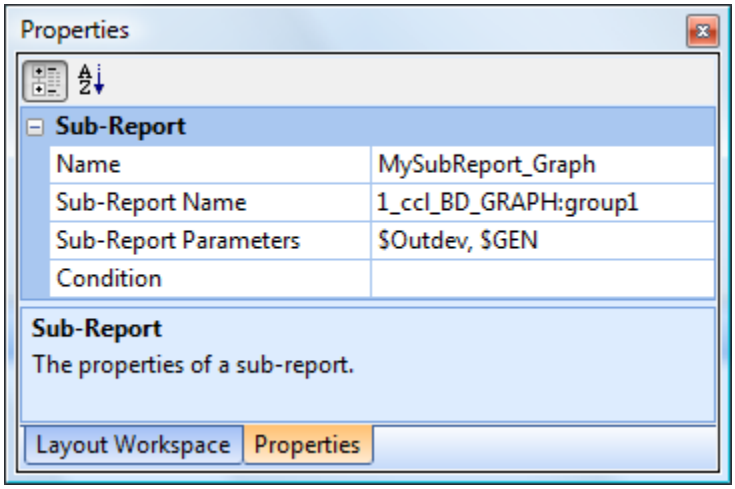
13. From the Edit > Insert menu option, select Sub-Report. A new sub-report section named Sub-Report: MySubReport(1) is created.

14. Modify the Name property to **MySubReport_Graph**.

15. Modify the Sub-Report Name property to the program containing the graph, 1_*your_initials*_BD~~PRSN~~_Graph. Append :**group1** to the object name if necessary.

16. Enter **$Outdev $GEN** for the Sub-Report Parameters property.

    The properties for the Sub-Report section should look like the following example:



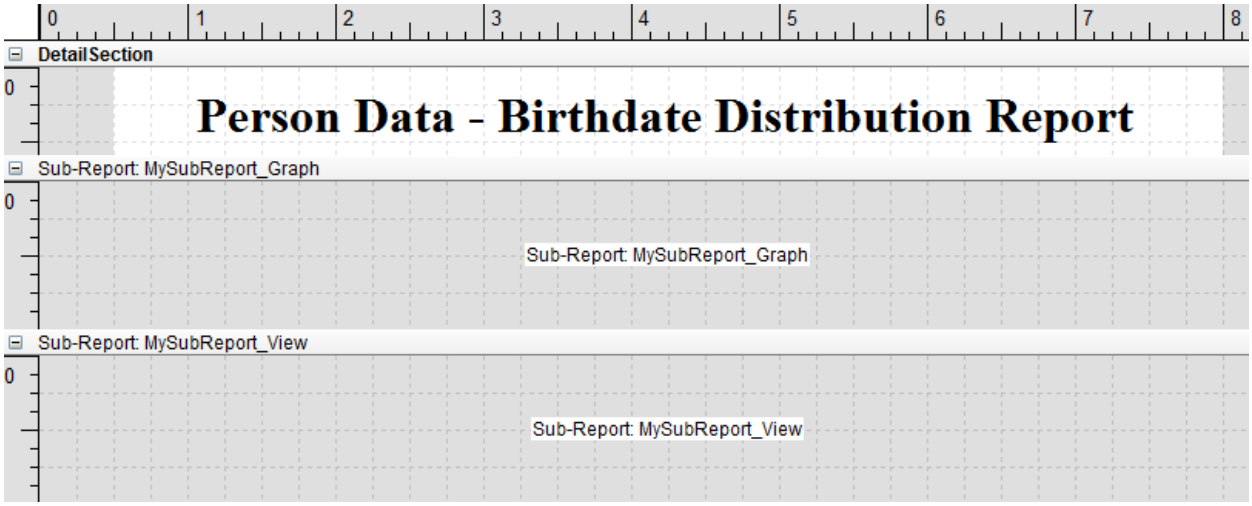
17. Validate that the sections are displayed in the layout in the following order and rearrange them if needed:

    DetailSection

    MySubreport_Graph

    MySubreport_View

    Your layout should look similar to the following example:



18. Press Ctrl+F5 to execute your layout program. Click Yes when prompted to save the layout. The prompt form opens.

19. Select Female for the second parameter and click Execute. The report should have two titles, and then below them display the information from the two sub-reports. Your report should look similar to the following format:

Doc ID: A1001001A08G07B34433A94851

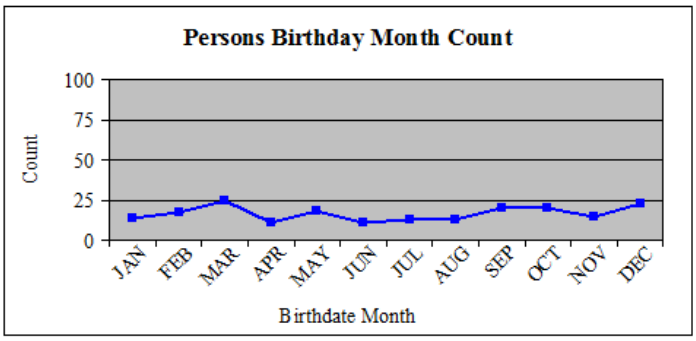
Person Data - Birthdate Distribution Report

Birthdate Distribution by Month

Page: 1 of 5

Female

JAN	14.00
FEB	17.00
MAR	25.00
APR	11.00
MAY	18.00
JUN	11.00
JUL	13.00
AUG	13.00
SEP	20.00
OCT	20.00
NOV	15.00
DEC	23.00



	Person ID	Name	Birth Date	Gender
1	593694	Smith, Trina	01/01/01 00:00:00	Female
2	594976	Tremmel, Sue	07/29/23 00:00:00	Female
3	593758	Lee, Khim	03/10/28 00:00:00	Female
4	595361	Baker, Ginger	10/03/28 00:00:00	Female
5	593747	Limshue, Katia	02/17/30 00:00:00	Female
6	595125	Lou, Marta	07/08/30 00:00:00	Female
7	594102	Pill, Jillene	11/10/33 00:00:00	Female
8	593927	Boban, Leanne	06/18/35 00:00:00	Female
9	595232	Imara, Lauris	07/09/35 00:00:00	Female
10	593924	Dock, Alison	10/17/35 00:00:00	Female
11	594274	Teach, Kathleen	09/15/36 00:00:00	Female

We need to control the logic so that when the 1_*your_initials*_BD_Graph is executed as a stand-alone program it displays the title *Birthdate Distribution Report*, however, when it executes as a Sub-Report from 1_*your_initials*_Parent, the title should display as *Person Data - Birthdate Distribution Report*. The conditional logic in 1_*your_initials*_BD_Graph controls the display of the title by checking for a variable populated when the program is called as a Sub-Report.

20. Open the layout program 1_*your_initials*_BD_Graph.

21. Access the Properties for the HeadreportSection and click the ellipses button on the Condition property.

22. Enter the following command and then click OK:

    _bSubReport=0

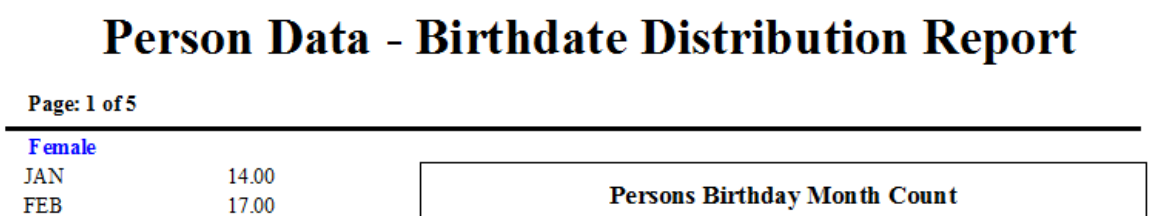
    If the program is executed as a stand-alone program the _bSubReport variable is 0 and the title from 1_*your_initials*_BD_Graph is rendered. If it is not 0, then the report is being called from a parent process and will not render the title.

23. Press Ctrl+S to save and include the change.

Execute 1_*your_intials*_Parent to verify only one title is displayed.

24. Most likely, you now have multiple files open. Click on the tab for 1_*your_intials*_Parent and press Ctrl+F5 to execute the program.

25. Select Female for the second parameter. Now your report should only have the title from the parent program displaying, similar to the following example:



26. Close the output window.

    Execute your program for all genders.

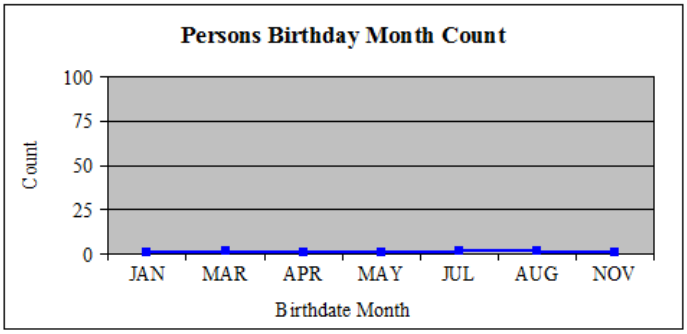
27. Press Ctrl + F5 and select Any(*) for the second parameter. The report should have one title and display the graphs for each gender and the table view of the person information. The first page of the report will display the graphs, similar to the following example:

Person Data - Birthdate Distribution Report

Page: 1 of 6

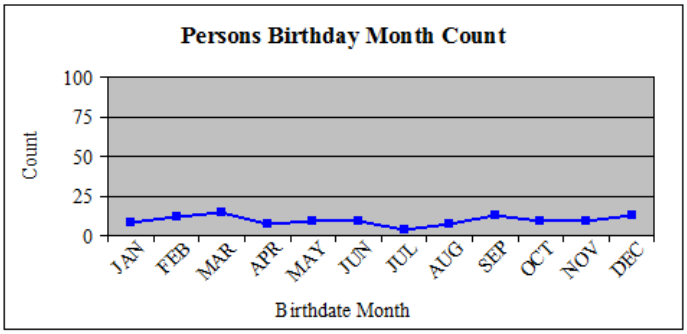
N/A

JAN	1.00
MAR	2.00
APR	1.00
MAY	1.00
JUL	2.00
AUG	2.00
NOV	1.00



Female

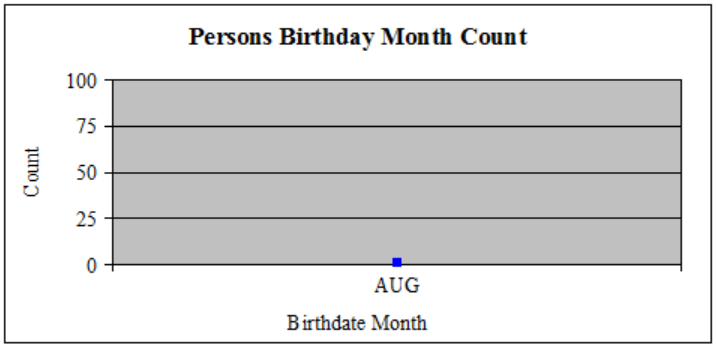
JAN	8.00
FEB	12.00
MAR	15.00
APR	7.00
MAY	9.00
JUN	9.00
JUL	4.00
AUG	7.00
SEP	13.00
OCT	9.00
NOV	9.00
DEC	13.00



There are a series of graphs for each gender, and once they are done, the next page displays the person information, similar to the following example:

Unknown

AUG	1.00



	Person ID	Name	Birth Date	Gender
1	593694	Smith, Trina	01/01/01 00:00:00	Female
2	594976	Tremmel, Sue	07/29/23 00:00:00	Female
3	593758	Lee, Khim	03/10/28 00:00:00	Female
4	595361	Baker, Ginger	10/03/28 00:00:00	Female
5	593747	Limshue, Katia	02/17/30 00:00:00	Female
6	595125	Lou, Marta	07/08/30 00:00:00	Female
7	594102	Pill, Jillene	11/10/33 00:00:00	Female
8	593927	Boban, Leanne	06/18/35 00:00:00	Female
9	595232	Imara, Lauris	07/09/35 00:00:00	Female
10	593924	Dock, Alison	10/17/35 00:00:00	Female
11	594274	Teach, Kathleen	09/15/36 00:00:00	Female

When Any(*) is chosen at the prompt, the series of graphs are displayed for each gender, and at the end displays the complete list of the person information used to create the graphs. We could re-arrange the display of the report so that for each gender the graph displays, and immediately following the gender specific person information that makes up the graph We need to detect when the user select Any(*) at prompt and control the flow of the program to display the graph and person information for one gender, then create a page break and display the graph and person information for each subsequent gender. One way to accomplish this is to create looping logic to execute the Sub-Reports for each gender.

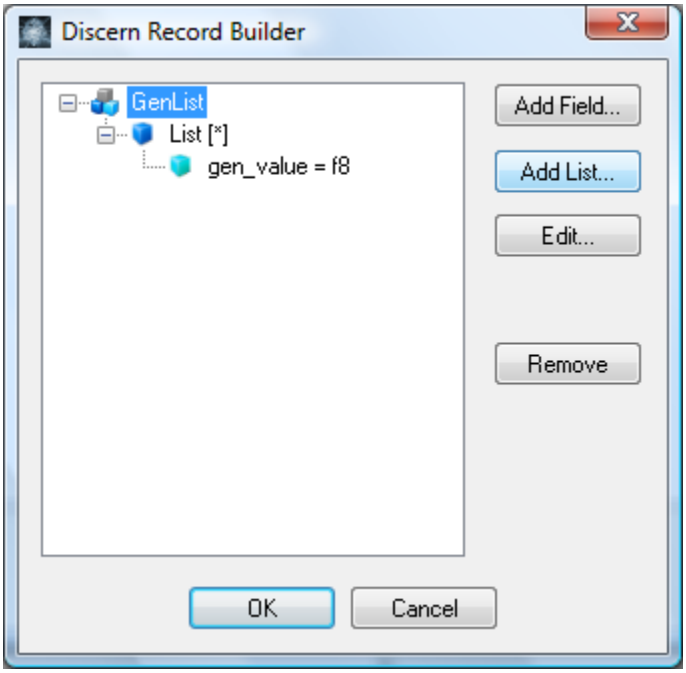
In the parent program we can build a record structure containing the code_value of the genders or gender that the user enters at the gender prompt. The sub-reports can execute for each of the code_values stored in the record structure using *For/Endfor* looping logic. Each code_value can be passed individually to the sub-reports.

28. With the 1_*your_intials*_Parent open, from the Tools menu, select Record Builder and click New Record.

29. Enter **GenList** for the Record Name and click OK.

30. Click Add List and enter **List** for the Name and ***** for the Occurs. Click OK.

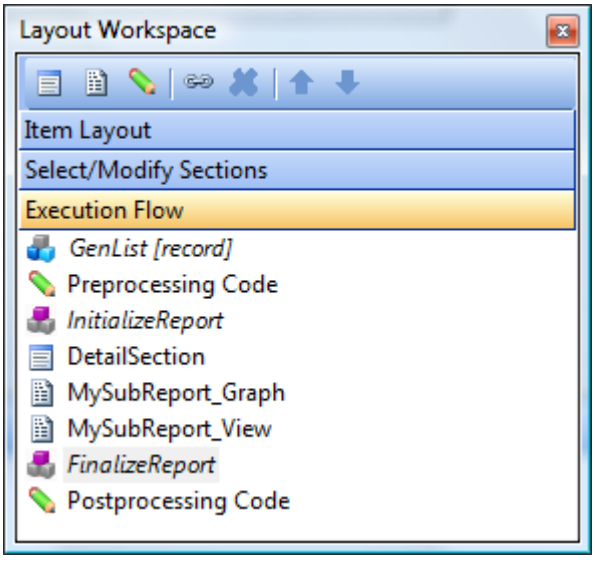
31. Click the newly added List. Click Add Field and enter **gen_value** for the Name and select **f8** for the Type, and click OK. Your record structure should look like the following example:



32. Click OK to close the Discern Record Builder dialog box, and click OK again to close the Add Records dialog box.

    This record structure should be loaded before the Sub-Reports are executed. We can use the Preprocessing Code section, which allows the ability to place commands that are needed for processing in the program.

33. From the LayoutWorkspace, select Execution Flow. Your Execution Flow should resemble the following example:



34. Double-click the Preprocessing Code, then copy and paste the following query which is used to load the record structure with one or multiple code_values:

```
IF ($GEN=0)  ;The user has select Any(*) at the prompt
   select into "nl:"
   from code_value c
   where code_set = 57
   head report
       cnt = 0
   detail
      cnt = cnt + 1
      stat = alterlist(genlist->list,cnt)
      genlist->list[cnt].gen_value = c.code_value
   with nocounter
ELSE ; The user has selected a specific gender at the prompt
   set stat = alterlist(genlist->list,1)
   set genlist->list[1].gen_value = $GEN
ENDIF
```

35. Click OK to close the Layout Source[Preprocessing Code].

    The *For* loop needs to be wrapped around the both of the Sub-Reports.

36. From the Execution Flow, click the New Code Segment button.

37. Use the Up arrow to move the Code Segment above MySubReport_Graph.

38. Double-click the code segment, then copy and paste the following commands:

```
FOR (_MyList = 1 to size(GenList->list,5))
  IF(_MyList > 1)  ; User select ANY(*) at the prompt and the first
gender has been processed
     call PageBreak(0)
     set stat = initrec(Prsn_Info) /*deallocate memory but retain the
definition */
  ENDIF
```

39. Click OK and access the Properties for the Code Segment.

40. Enter **For Loop** for the Name property.

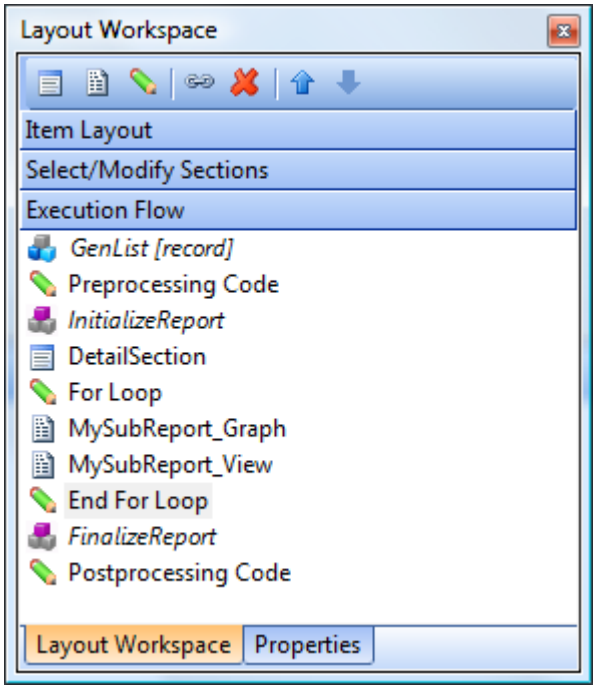
41. From the Execution Flow, click the New Code Segment button.

42. Verify that the new code segment is below MySubReport_View. Move the segment if necessary.

43. Double-click on the code segment and enter **ENDFOR**.

44. Click OK and access the Properties for the Code Segment.

45. Enter **End For Loop** for the Name property. Your Execution Flow section should resemble the following example:



One final step is to modify the parameter passed to each of the Sub-Reports so that one code_value at a time is passed to the programs. The record item GenList->list[_MyList].gen_value contains the code_value.

46. Access the Properties for MySubReport_Graph and modify the Sub-Report Parameters to the following example:

    $Outdev, GenList->list[_MyList].gen_value

47. Access the Properties for MySubReport_View and modify the Sub-Report Parameters to the following example:

    $Outdev, GenList->list[_MyList].gen_value

48. Press Ctrl+F5 and click Yes when prompted to save the layout.

49. Select Any(*) for the second parameter and click execute. The report should display the graph and person information for one gender. At the end of the information for that gender, there should be a page break. Then the next page should show the graph and person information for the next gender.

Manually Calling Layout Subroutines From Report Programs

Most likely, the easiest way to add layout functionality to an existing query program is to convert the existing program to a driver program and call it from a layout program (see

the Converting Existing Programs to Layout Programs Using a Driver Program section for more information on converting an existing program into a driver program).

When you create a layout program, DVDev creates subroutines for each layout section and places commands to call those subroutines into the appropriate reportwriter sections of the query associated with the layout. Appendix A contains additional information about the subroutines that DVDev creates. You can use the Layout Builder to create layout sections and to add the code to call those sections to an existing program; however, doing so requires you to manually add the code to call the layout section subroutines to the source code that creates the query or program. You will also need to control page breaks and manage the overall flow of the program.

For example, suppose you had the following source code, which creates a program that prompts for a last name and then selects person and encounter information for people with a last name equal to the value entered at the prompt. You want to use the layout builder functionality to format the results of the query.

```
drop program 1_ccl_add_layt_to_prg go
create program 1_ccl_add_layt_to_prg

prompt
"Output to File/Printer/MINE" = "MINE"
, "Enter Last Name"  =  ""

with OUTDEV, LName
SELECT INTO $OUTDEV
 P.PERSON_ID,
 P.NAME_FULL_FORMATTED,
 P_SEX_DISP = UAR_GET_CODE_DISPLAY( P.SEX_CD ),
 AGE = CNVTAGE(P.BIRTH_DT_TM, E.REG_DT_TM,0),
 E.ENCNTR_ID,
 E_ENCNTR_TYPE_CLASS_DISP =
    UAR_GET_CODE_DISPLAY( E.ENCNTR_TYPE_CLASS_CD )
FROM
 PERSON  P,
 ENCOUNTER  E
PLAN P WHERE  P.NAME_LAST_KEY =  $LNAME
JOIN E WHERE  P.PERSON_ID =  E.PERSON_ID
ORDER BY
 P.PERSON_ID,
 E.ENCNTR_ID
WITH MAXREC = 1000 , NOCOUNTER, SEPARATOR=" ", FORMAT
end
go
```

50. Copy the text from the above example and paste it into a blank file in DVDev.

51. Modify the name of the program to match the naming conventions of your site. For example, modify 1_ccl_add_layt_to_prg to **1_*your_initials*_ add_layt_to_prg**. If you are logged in to DVDev using a cclgroup1 account, ensure that the total length of the program name is 23 or fewer characters. Subsequent steps in this exercise will create a layout and generate a file with a .DVL extension that contains the source code for the layout subroutines. The program name will be used as the file name when creating the .DVL file. When using a cclgroup1 account, if the program name exceeds 23 characters, the name of the .dvl file will be truncated. The subsequent instructions assume the name of the .DVL file is the same as the program name used in this step. To avoid confusion, ensure your program name is 23 or fewer characters in length when using a cclgroup1 account. If you are using a cclgroup0 (DBA) account, the name of the .DVL file will not be truncated, so the

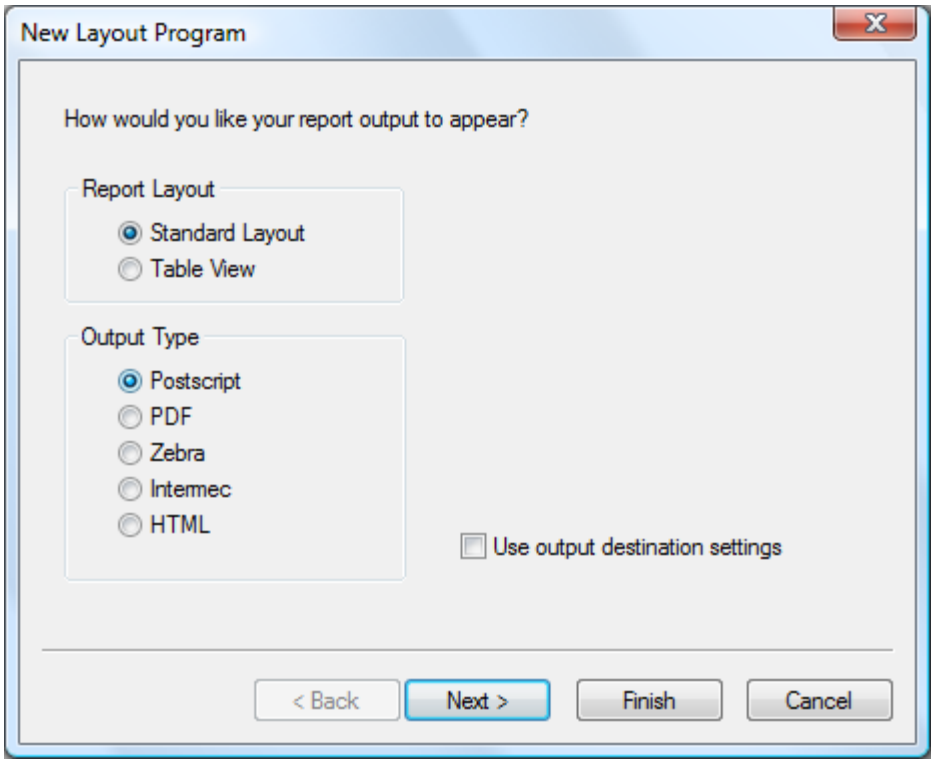
program name can use up to the 30 characters as allowed by Discern Explorer for program names.

52. Save the source code file and verify that it will include/compile without any errors.

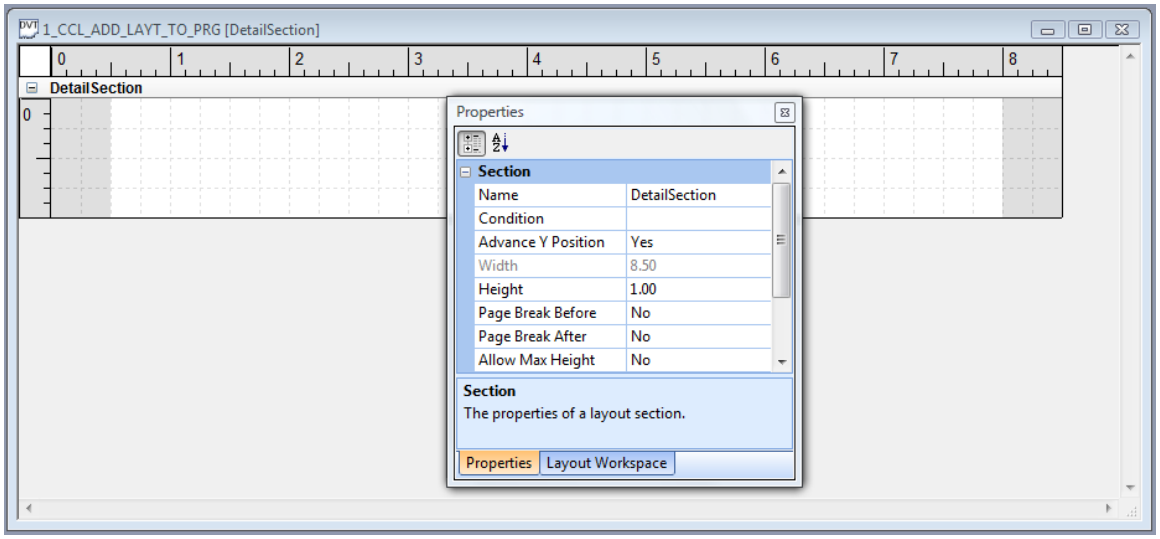
53. Execute the program as a prompt program to verify that it executes without error and returns information about people and their encounters.

Creating the Layout

54. Add a layout to the program created in the source code in the section above by placing your cursor within the body of the Create Program command; from the Tools menu, select Layout Builder. The New Layout Program dialog box opens.



55. Verify that the Standard Layout option for the Report Layout and PostScript option for the Output Type are selected, and click Finish. A layout with a single section named DetailSection is created.



For our example, we want to format the data returned from the PERSON and ENCOUNTER tables, so we need a report header at the top of the report, the page number and column headings at the top of each page, some specific information about each person, and some specific information about each encounter. To accomplish this we must add layout sections for the head report, head page, head PERSON_ID, and detail.

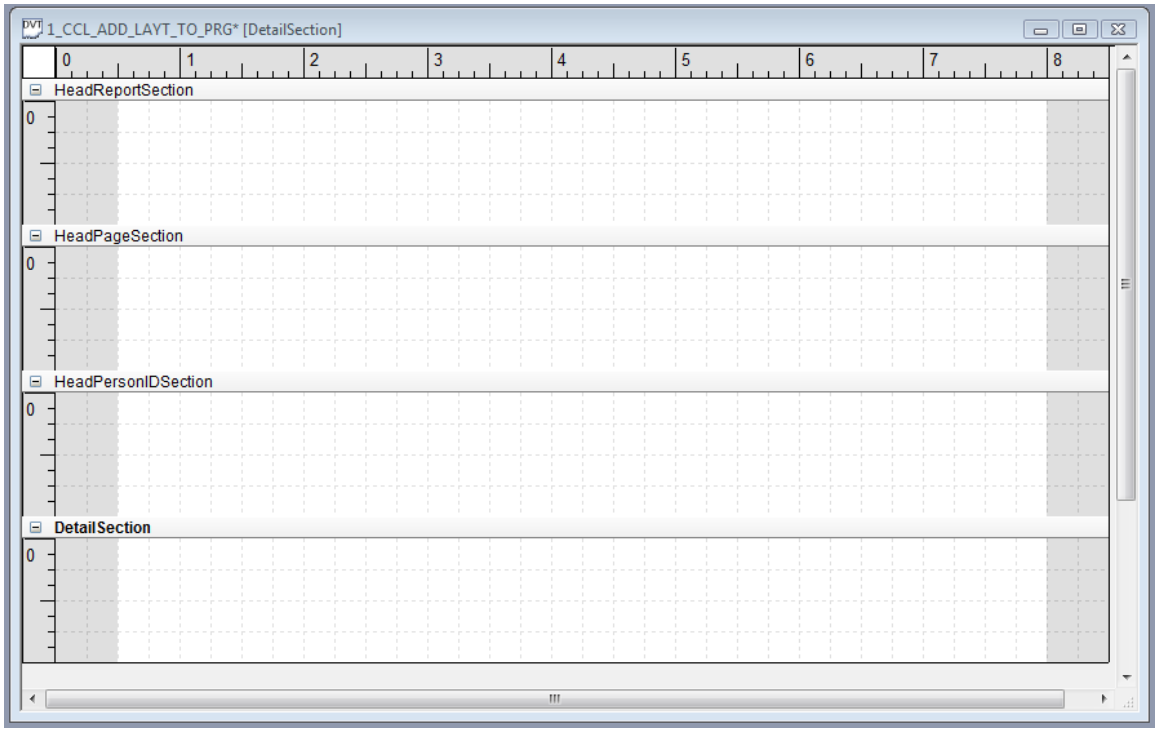
56. From the Edit menu, select Insert > Section to add three more sections to your layout.

57. Click in the first section; in the Properties dialog box, modify the Name property to **HeadReportSection**.

58. Click in the second section; in the Properties dialog box, modify the Name property to **HeadPageSection**.

59. Click in the third section; in the Properties dialog box, modify the Name property to **HeadPersonIDSection**.

60. The fourth section should already be labeled DetailSection. Modify the Name property if needed. You can expect your layout to be similar to the following screen:

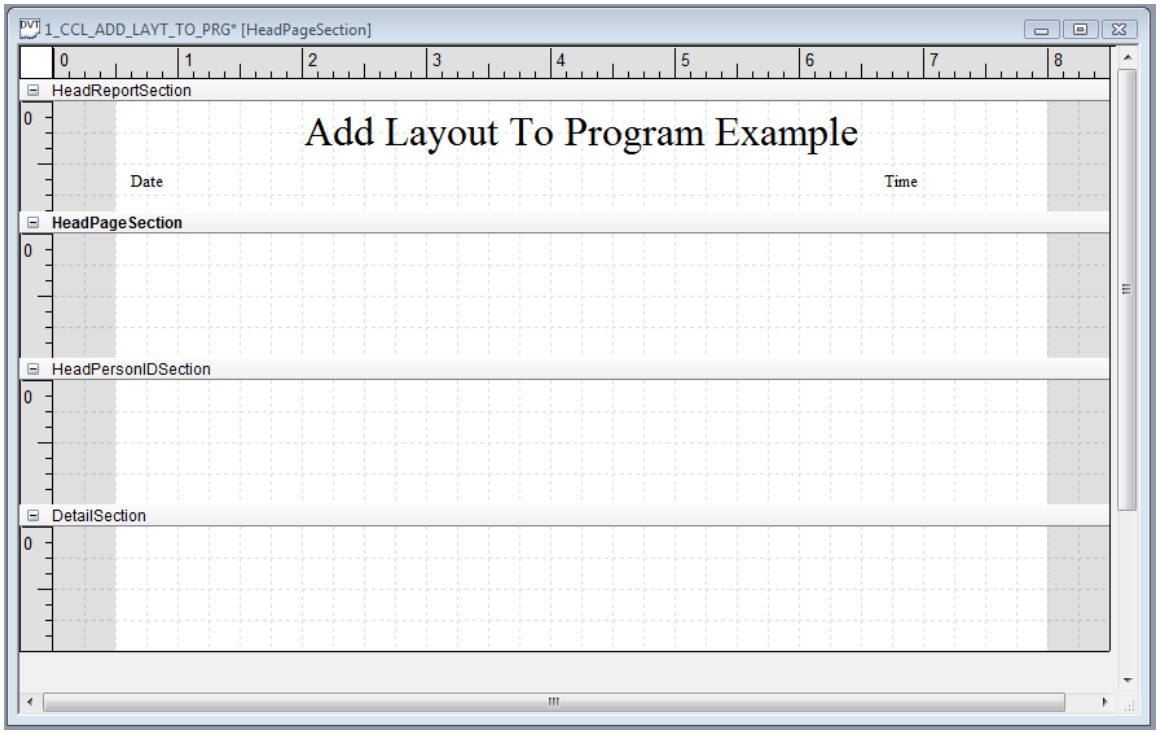


61. Use the Label Tool to add a report title Add Layout To Program Example to the HeadReportSection, and use the Formatting toolbar to set the following values:

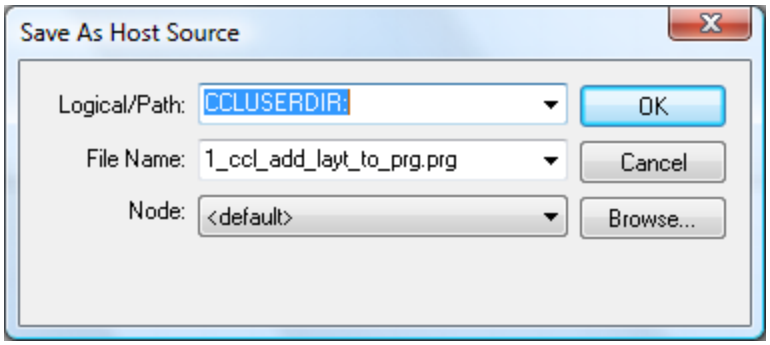
- The font to Times
- The font size to 24
- Center the text

62. Use the Text Tool to add a field to the HeadReportSection to display the current date when the layout program is executed. In the Properties dialog box, modify the Name property to **Date** and the Source to **CURDATE**.

63. Use the Text too to add a field to the HeadReportSection to display the current time when the layout program is executed. In the Properties dialog box, modify the Name property to **Time** and the Source to **CURTIME2**. You can expect your layout to be similar to the following screen:



64. From the File menu, select Save As. The Save As Host Source dialog box opens.



65. Verify the Logical/Path is CCLUSERDIR: and the File Name: is 1_*your_initials*_add_layt_to_prg.prg and click OK.

Saving the layout creates a file that has the same name as the program name that is used in the Create Program command with a .DVL extension. By default, the *Program_Name*.dvl file is saved in the CCLUSERDIR: directory on the back-end host. The *Program_Name*.dvl file contains the commands to create the layout subroutines used to render the output. We now need to add the commands to the source code to initialize and call the layout subroutines.

Doc ID: A1001001A08G07B34433A94851

Calling Layout Sections in the Existing Program/Query

1. Add the following commands to your source code file. These commands must be placed after the Prompt clause and before the Select command.

```
execute reportrtl
%i ccluserdir:1_your_initials_add_layt_to_prg.dvl
;%i must be in the first two columns of the source code file.
;Assuming you changed the program name in your file to
;1_your_initials_add_layt_to_prg.
;set a variable to initializereport(0)
set d0 = InitializeReport(0)
```

2. Modify Select Into $OutDev to **Select Into "NL:"**.

Instead of selecting into $OutDev, you will send the information to the Null Device ("NL;"). Then, later in the program, you will pass $OutDev to a subroutine to direct the output to the device that the user enters at the prompt.

3. Add a Head Report section to the Select command that contains the following command:

```
d0 = HeadReportSection(Rpt_Render)
```

This is assuming you created a layout section named HeadReportSection.

4. Add the following command after the Select command's With clause, but before the End Go:

```
set d0 = FinalizeReport($OutDev)
```

The following example shows where these commands are added to the program you copied above:

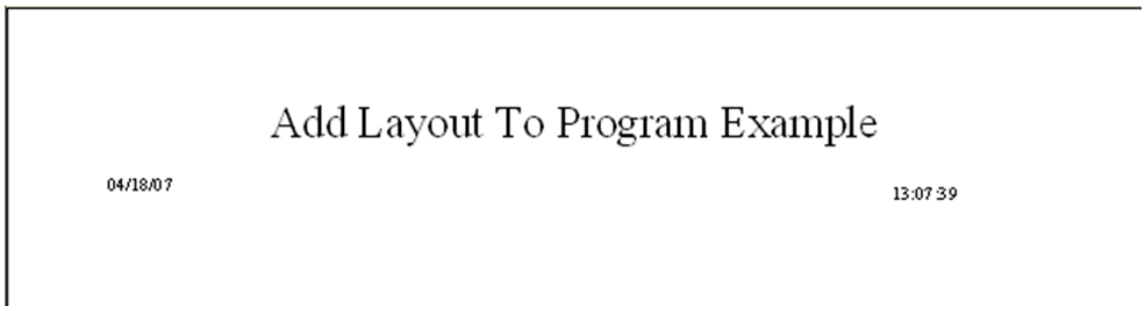
```
drop program 1_ccl_add_layt_to_prg go
create program 1_ccl_add_layt_to_prg

prompt
  "Output to File/Printer/MINE" = "MINE"
  , "Enter Last Name" = " "

with OUTDEV, LName

execute reportrtl
%i ccluserdir: 1_your_initials_add_layt_to_prg.dvl
;%i must be in the first two columns of the source code file.
;Assuming you changed the program name in your file to
;1_your_initials_add_layt_to_prg.
;set a variable to initializereport(0)
set d0 = InitializeReport(0)

SELECT INTO "NL:"  ;$OUTDEV
  P.PERSON_ID,
  P.NAME_FULL_FORMATTED,
  P_SEX_DISP = UAR_GET_CODE_DISPLAY( P.SEX_CD ),
  AGE = CNVTAGE(P.BIRTH_DT_TM, E.REG_DT_TM,0),
  E.ENCNTR_ID,
  E_ENCNTR_TYPE_CLASS_DISP =
     UAR_GET_CODE_DISPLAY( E.ENCNTR_TYPE_CLASS_CD )
FROM
```

```
 PERSON  P,
 ENCOUNTER  E
PLAN P WHERE  P.NAME_LAST_KEY =  $LNAME
JOIN E WHERE  P.PERSON_ID =  E.PERSON_ID
ORDER BY
 P.PERSON_ID,
 E.ENCNTR_ID
Head report
 d0 = HeadReportSection(Rpt_Render)

WITH MAXREC = 1000 , NOCOUNTER, SEPARATOR=" ", FORMAT

set d0 = FinalizeReport($OutDev)

end
go
```

5. After adding the commands to your source code, save the file and include/compile it.

6. Execute the program as a prompt program by using CTRL+R, by clicking Run Prompt Program, or by selecting the Run Prompt Program option from the Build menu.

7. Use MINE as the output device and enter the last name of a person that you know has encounters when prompted. You can expect your output to be similar to the following screen:



At this point your program creates only the output from the HeadReportSection. To display additional information in the output, you need to add the information to the layout and modify the source code to call the layout subroutines.

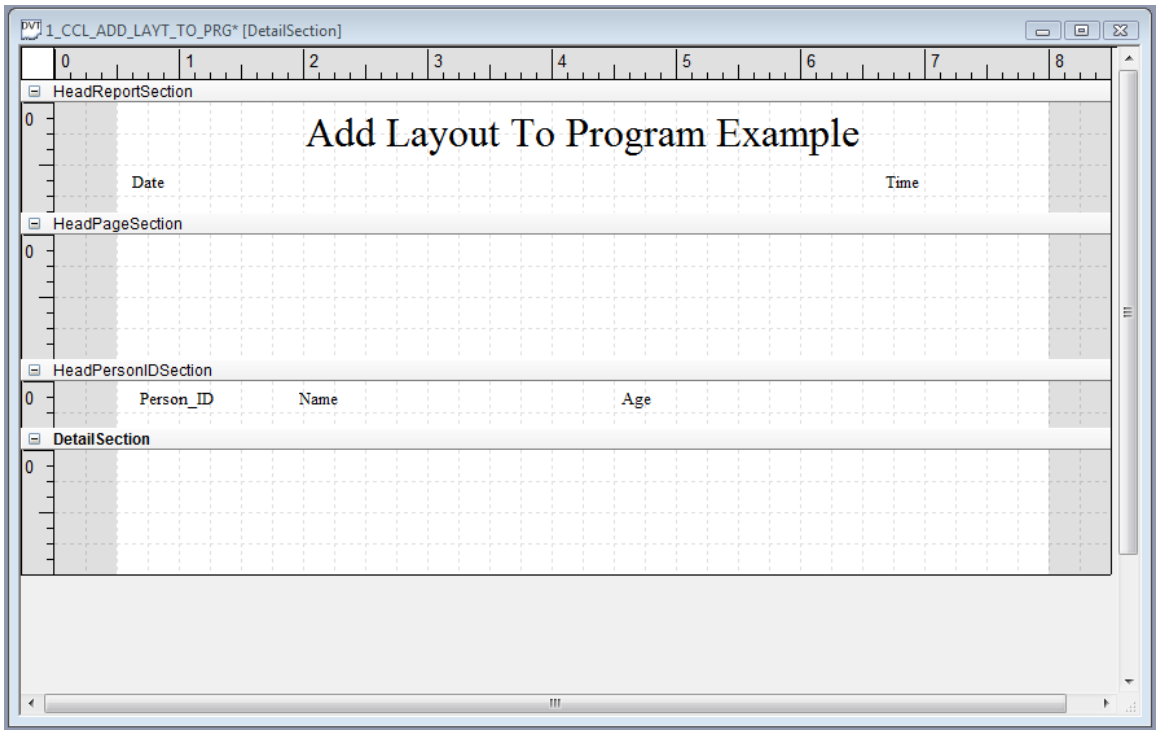
8. Use the Text Tool  to add a field to the HeadPersonIDSection to display the Person_ID. In the Properties dialog box, modify the Name property to **Person_ID** and the Source to **P.Person_ID**.

9. Use the Text Tool  to add a field to the HeadPersonIDSection to display the person's full name. In Properties dialog box, modify the Name property to **Name** and the Source to **P.Name_Full_Formatted**.

10. Use the Text Tool  to add a field to the HeadPersonIDSection to display the person's gender. In the Properties dialog box, modify the Name property to **Gender** and the Source to **P_Sex_Disp**.

11. Use the Text Tool  to add a field to the HeadPersonIDSection to display the person's age. In the Properties dialog box, modify the Name property to **Age** and the Source to **Age**.

12. Slowly move your pointer over the DetailSection title bar. At the top of the title bar the pointer will change to the vertical resize pointer. When the pointer changes, click and drag the DetailSection title bar up to the bottom of the fields you placed in the HeadPersonIDSection. You can expect your layout to be similar to the following screen:



13. Save the layout.

You are now ready to add the commands to your source code to execute the HeadPersonIDSection subroutine in the Head P.Person_ID reportwriter section of the Select command. Here, however, we run into an interesting issue that results from the way *Discern Explorer* performs the internal reportwriter section processing. If the query uses reportwriter sections, then only expressions created in the select list, the fields that are used as Head/Foot group expression sections, or fields that are used in a reportwriter section are recognized in the internal processing. Layout Builder creates subroutines that are outside the reportwriter sections. You then add the code in the reportwriter sections to call the subroutines.

Doc ID: A1001001A08G07B34433A94851

Any fields that are used on the layout must be referenced somewhere within a reportwriter section to be recognized by the internal processing. If the field is used only in the layout subroutine, it is not recognized by the internal processing and generates a %CCL-E-85 error. Currently, you have added the P.Person_ID field, the P.Name_Full_Formatted field, the expression P_Sex_Disp, and the expression Age to your HeadPersonIDSection layout section. In order for the fields P.Name_Full_Formatted and P.Person_ID to be recognized when the layout subroutine is called, they must be referenced somewhere in a reportwriter section before the subroutine is called.

14. Add the following command to the Head Report section of your source code:

```
F1=  p.name_full_formatted
```

The command above is used only to make the P.Name_Full_Formatted field known to the internal processing. If we did not use this command, *Discern Explorer* would return an error similar to the following when the layout program was executed:

```
%CCL-E-85-1_CCL_ADD_LAYT_TO_PRG(0,0)S58,L7,Rpt{}Report attribute
(P.NAME_FULL_FORMATTED) was not selected for retrieval in select clause.
```

We do not need to add a similar command for the P.Person_ID field because we use P.Person_ID to create a Head group expression section. Also, we also do not need to add similar commands for the P_Sex_Disp or Age expressions. However, we will need to add similar commands for other fields that we want to display that are not used as a Head/Foot group expression.

15. Use the following code to create a Head P.Person_ID reportwriter section and call the HeadPersonIDSection subroutine:

```
Head P.Person_ID
d0 = HeadPersonIDSection(Rpt_Render)
```

Using Head P.Person_ID makes the P.Person_ID field known to the internal processing.

Your source code should now resemble the following example:

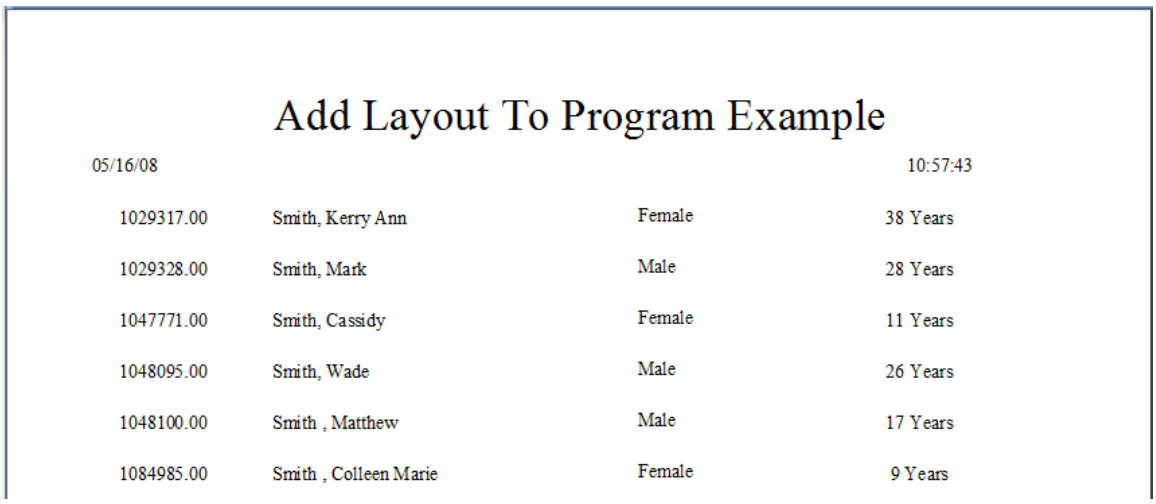
```
drop program 1_ccl_add_layt_to_prg go
create program 1_ccl_add_layt_to_prg

prompt
  "Output to File/Printer/MINE" = "MINE"
  , "Enter Last Name" = " "

with OUTDEV, LName

execute reportrtl
%i ccluserdir: 1_your_initials_add_layt_to_prg.dvl
;%i must be in the first two columns of the source code file.
;Assuming you changed the program name in your file to
;1_your_initials_add_layt_to_prg.
;set a variable to initializereport(0)
set d0 = InitializeReport(0)

SELECT INTO "NL:"  ;$OUTDEV
  P.PERSON_ID,
  P.NAME_FULL_FORMATTED,
  P_SEX_DISP = UAR_GET_CODE_DISPLAY( P.SEX_CD ),
  AGE = CNVTAGE(P.BIRTH_DT_TM, E.REG_DT_TM,0),
  E.ENCNTR_ID,
  E_ENCNTR_TYPE_CLASS_DISP =
     UAR_GET_CODE_DISPLAY( E.ENCNTR_TYPE_CLASS_CD )
FROM
  PERSON  P,
```

```
 ENCOUNTER  E
PLAN P WHERE  P.NAME_LAST_KEY =  $LNAME
JOIN E WHERE  P.PERSON_ID =  E.PERSON_ID
ORDER BY
 P.PERSON_ID,
 E.ENCNTR_ID
Head report
 F1 = p.name_full_formatted
 d0 = HeadReportSection(Rpt_Render)
Head P.Person_ID
 d0 = HeadPersonIDSection(Rpt_Render)

WITH MAXREC = 1000 , NOCOUNTER, SEPARATOR=" ", FORMAT

set d0 = FinalizeReport($OutDev)

end
go
```

16. After adding the commands to your source code, save the file and include/compile it.

17. Execute the program as a prompt program by using CTRL+R, by clicking Run Prompt Program, or by selecting the Run Prompt Program option from the Build menu.

18. Use MINE as the output device and, when prompted, enter the last name of a person that you know has encounters. Your output should resemble the following screen:



Add Layout To Program Example

05/16/08 10:57:43

1029317.00	Smith, Kerry Ann	Female	38 Years
1029328.00	Smith, Mark	Male	28 Years
1047771.00	Smith, Cassidy	Female	11 Years
1048095.00	Smith, Wade	Male	26 Years
1048100.00	Smith , Matthew	Male	17 Years
1084985.00	Smith , Colleen Marie	Female	9 Years

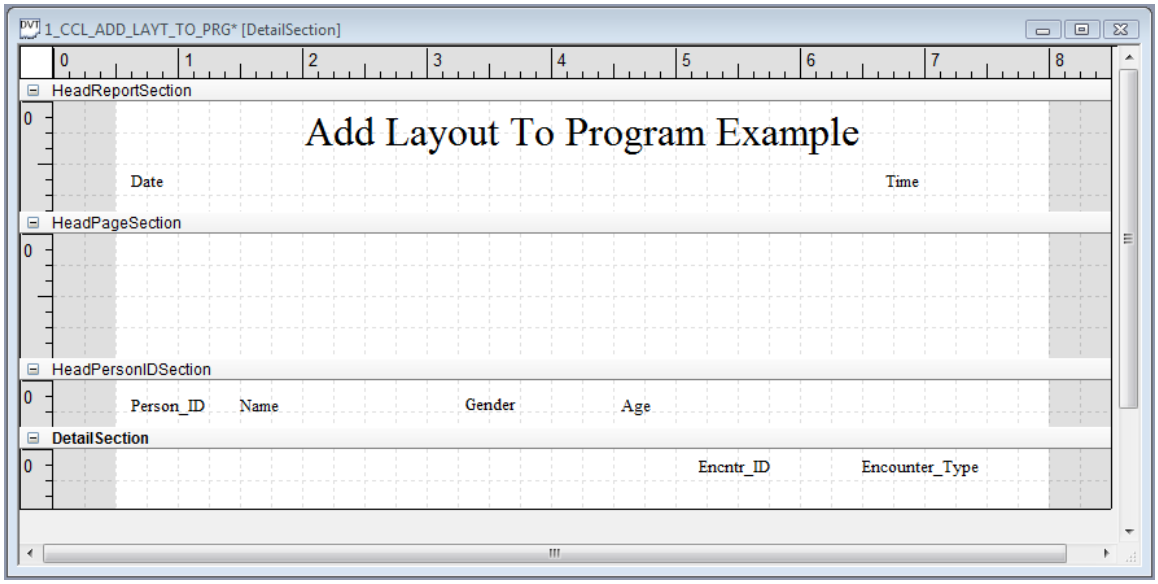
To continue, we need to add information about each person's encounters to the DetailSection layout section.

19. Close the output window and return to the layout.

20. Use the Text Tool  to add a field towards the right hand side of the DetailSection to display the Encntr_ID. In the Properties dialog box, modify the property to **Encntr_ID** and the Source to **E.Encntr_ID**.

21. Use the Text Tool  to add a field to the right of the Encntr_ID of the DetailSection to show the display value of the encounter type code. In the Properties dialog box, modify the Name property to **Encounter_Type** and the Source to **E_Encntr_Type_Class_Disp**.

22. Use the vertical resize to move the bottom of the DetailSection up to just below the items you added. If necessary, use the scroll bar on the right of the layout to scroll down until you can see the bottom of the DetailSection so you can grab and resize it.



23. Save the layout.

You are now ready to add the commands to your source code to execute the DetailSection subroutine in the Detail reportwriter section of the Select command.

24. Add the following command to the Head Report section of your source code:

```
F2 =  e.encntr_id
```

25. Use the following code to create a Detail reportwriter section and call the DetailSection subroutine:

```
Detail
d0 = DetailSection(Rpt_Render)
```

Your source code should now resemble the following example:

```
drop program 1_1_ccl_add_layt_to_prg go
create program 1_1_ccl_add_layt_to_prg

prompt
  "Output to File/Printer/MINE" = "MINE"
  , "Enter Last Name" = " "

with OUTDEV, LName

execute reportrtl
```

```
%i ccluserdir: 1_your_initials add_layt_to_prg.dvl
;%i must be in the first two columns of the source code file.
;Assuming you changed the program name in your file to
;1_your_initials_add_layt_to_prg.
;set a variable to initializereport(0)
set d0 = InitializeReport(0)

SELECT INTO "NL:"  ;$OUTDEV
 P.PERSON_ID,
 P.NAME_FULL_FORMATTED,
 P_SEX_DISP = UAR_GET_CODE_DISPLAY( P.SEX_CD ),
 AGE = CNVTAGE(P.BIRTH_DT_TM, E.REG_DT_TM,0),
 E.ENCNTR_ID,
 E_ENCNTR_TYPE_CLASS_DISP =
    UAR_GET_CODE_DISPLAY( E.ENCNTR_TYPE_CLASS_CD )
FROM
 PERSON  P,
 ENCOUNTER  E
PLAN P WHERE  P.NAME_LAST_KEY =  $LNAME
JOIN E WHERE  P.PERSON_ID =  E.PERSON_ID
ORDER BY
 P.PERSON_ID,
 E.ENCNTR_ID
Head report
 F1 = p.name_full_formatted
 F2 = e.encntr_id
 d0 = HeadReportSection(Rpt_Render)
Head P.Person_ID
 d0 = HeadPersonIDSection(Rpt_Render)
Detail
 d0 = DetailSection(Rpt_Render)

WITH MAXREC = 1000 , NOCOUNTER, SEPARATOR=" ", FORMAT

set d0 = FinalizeReport($OutDev)

end
go
```

26. After adding the commands to your source code, save the file and include/compile it.

27. Execute the program as a prompt program by using CTRL+R, by clicking Run Prompt Program, or by selecting Run Prompt Program option from the Build menu.

28. Use MINE as the output device and, when prompted, enter the last name of a person that you know has encounters. You can expect your output to be similar to the following screen:

Add Layout To Program Example

05/16/08 11:14:03

1029328.00	Smith, Mark	Male	28 Years	
			7648588.00	Emergency
1047771.00	Smith, Cassidy	Female	11 Years	
			5896428.00	Inpatient
1048095.00	Smith, Wade	Male	26 Years	
			5973103.00	Inbox Message
			6167553.00	Inbox Message
1048100.00	Smith , Matthew	Male	17 Years	
			6205055.00	Inpatient
1236848.00	Smith, Test	Male	4 Months	
			6681012.00	Inbox Message
1599582.00	Smith, Will E.	Male	8 Years	
			6669029.00	Inbox Message
			6673130.00	Inbox Message
			6675003.00	Inbox Message
			6675015.00	Inbox Message
			6733012.00	Inbox Message

You may notice a couple of things about the output that you want to modify. For example, a person's first encounter is displayed one row below their person information. To align the first encounter with the person information you need to set the Advance Y Position section property to No.

29. Close the output window and return to the layout.

30. Click in the HeadPersonIDSection and in the Properties dialog box; modify the Advance Y Position from Yes to No.

31. Save the layout.

32. Include/compile your source code file.

33. Execute the program as a prompt program.

34. Use MINE as the output device and, when prompted, enter the last name of a person that you know has encounters. When the output is displayed, verify the person's first encounter is displayed on the same line as their person information.

35. Close the output window and return to the layout.

Generating Page Breaks

You may have noticed that you are getting only one page of output. Depending on the last name that you have been using at the prompt and the amount of data you have in your environment, you may have already noticed that when the output appears to fill a page and you attempt to click the next page button  , the next page and previous page buttons are unavailable and your output will still display the same page. If your query has not returned enough data to fill the first page, try entering a different name or part of a name, followed by an asterisk (*) to qualify enough data so that more than one page is required to display it. The output only has one page because, up to this point, we have only added code to call the layout section subroutines. We have not added the code needed to generate page breaks when the amount of space used by the subroutines exceeds the amount of space on the page. Since we are calling the subroutines, we also need to control when a page break is created. It is important to realize that since the code does not generate a page break, any information that does not fit on the page will not be displayed when we attempt to render.

When you first create a layout, the Report Properties default paper size is 8.5 by 11 inches with .5 inch margins and portrait orientation. The default properties can be modified using the Report > Report Properties command from the Edit menu. Layout Builder subroutines use a variable named _YOffset to track the position going down the page. Using the default settings, the bottom line on the page is rendered 10.5 inches down from the top of the page. When the height of a layout section, plus the _YOffset is greater than the height of the page minus the bottom margin, the program needs to generate a page break before the section is rendered. Using the defaults, if the height of a layout section plus the current value of _YOffset is greater than 10.5, a page break must be generated before rendering the layout section.

Two steps are required to generate the page break. First, the Break command needs to be issued to force *Discern Explorer* to execute the Head/Foot Page reportwriter sections. Next the PageBreak() layout subroutine, created by Layout Builder, must be executed to generate the page break command and reset _YOffset to the top margin of the page. A layout section subroutine is called using the format *Layout_Section_Subroutine(parameter)* where *parameter* is either Rpt_Render or Rpt_CalcHeight. When passing either Rpt_Render or Rpt_CalcHeight, the height of the layout section is returned. If Rpt_Render is used, the layout section is rendered and the _YOffset variable is incremented. If Rpt_CalcHeight is used, the layout section is not rendered and the _YOffset variable is not incremented.

1. Add the following IF statement to the top of your reportwriter detail section to generate a page break if the layout DetailSection will not fit on the current page:

```
if(_YOffset + DetailSection(Rpt_CalcHeight) > 10.5)
 Break
endif
```

2. Add the following statement after your reportwriter Head Report section to create a reportwriter Head Page section and generate a page break when necessary:

```
Head Page
 if( curpage > 1)
  d0 = PageBreak(0)
 endif
```

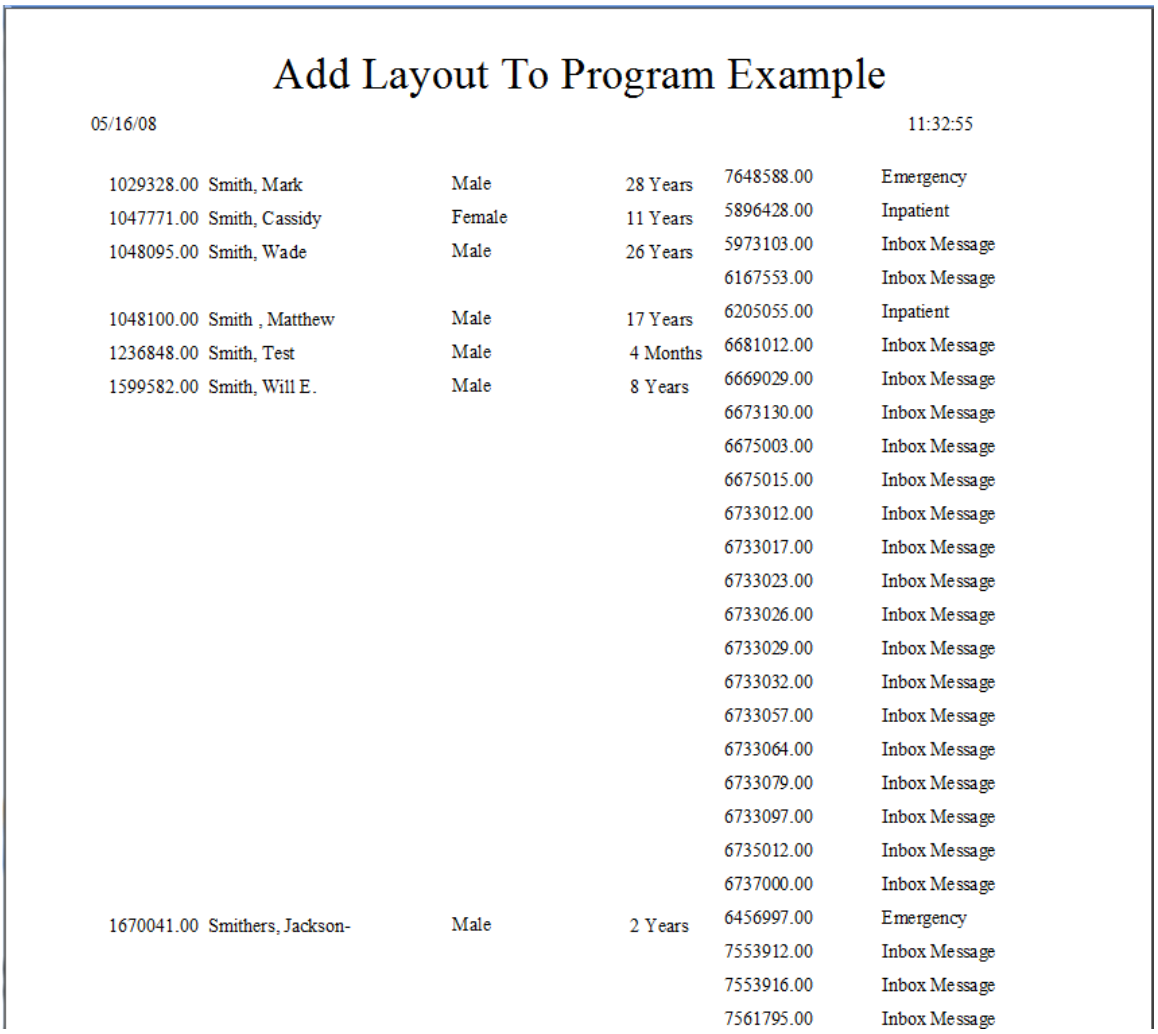
Your source code should now resemble the following example:

```
drop program 1_ccl_add_layt_to_prg go
create program 1_ccl_add_layt_to_prg

prompt
  "Output to File/Printer/MINE" = "MINE"
  , "Enter Last Name" = " "

with OUTDEV, LName

execute reportrtl
%i ccluserdir: 1_your_initials_add_layt_to_prg.dvl
;%i must be in the first two columns of the source code file.
;Assuming you changed the program name in your file to
;1_your_initials_add_layt_to_prg.
;set a variable to initializereport(0)
set d0 = InitializeReport(0)

SELECT INTO "NL:"  ;$OUTDEV
  P.PERSON_ID,
  P.NAME_FULL_FORMATTED,
  P_SEX_DISP = UAR_GET_CODE_DISPLAY( P.SEX_CD ),
  AGE = CNVTAGE(P.BIRTH_DT_TM, E.REG_DT_TM,0),
  E.ENCNTR_ID,
  E_ENCNTR_TYPE_CLASS_DISP =
     UAR_GET_CODE_DISPLAY( E.ENCNTR_TYPE_CLASS_CD )
FROM
  PERSON  P,
  ENCOUNTER  E
PLAN P WHERE  P.NAME_LAST_KEY =  $LNAME
JOIN E WHERE  P.PERSON_ID =  E.PERSON_ID
ORDER BY
  P.PERSON_ID,
  E.ENCNTR_ID
Head report
  F1 = p.name_full_formatted
  F2 = e.encntr_id
  d0 = HeadReportSection(Rpt_Render)
Head Page
  if( curpage > 1)
    d0 = PageBreak(0)
  endif
Head P.Person_ID
  d0 = HeadPersonIDSection(Rpt_Render)
Detail
  if(_YOffset + DetailSection(Rpt_CalcHeight) > 10.5)
    Break
  endif
  d0 = DetailSection(Rpt_Render)

WITH MAXREC = 1000 , NOCOUNTER, SEPARATOR=" ", FORMAT

set d0 = FinalizeReport($OutDev)

end
go
```

3. After adding the commands to your source code, save the file and include/compile it.

4. Execute the program as a prompt program.

5. Use MINE as the output device and, when prompted, enter the last name of a person that you know has encounters. You can expect your output to be similar to the following screen:



Add Layout To Program Example

05/16/08 11:32:55

1029328.00	Smith, Mark	Male	28 Years	7648588.00	Emergency
1047771.00	Smith, Cassidy	Female	11 Years	5896428.00	Inpatient
1048095.00	Smith, Wade	Male	26 Years	5973103.00	Inbox Message
				6167553.00	Inbox Message
1048100.00	Smith , Matthew	Male	17 Years	6205055.00	Inpatient
1236848.00	Smith, Test	Male	4 Months	6681012.00	Inbox Message
1599582.00	Smith, Will E.	Male	8 Years	6669029.00	Inbox Message
				6673130.00	Inbox Message
				6675003.00	Inbox Message
				6675015.00	Inbox Message
				6733012.00	Inbox Message
				6733017.00	Inbox Message
				6733023.00	Inbox Message
				6733026.00	Inbox Message
				6733029.00	Inbox Message
				6733032.00	Inbox Message
				6733057.00	Inbox Message
				6733064.00	Inbox Message
				6733079.00	Inbox Message
				6733097.00	Inbox Message
				6735012.00	Inbox Message
				6737000.00	Inbox Message
1670041.00	Smithers, Jackson-	Male	2 Years	6456997.00	Emergency
				7553912.00	Inbox Message
				7553916.00	Inbox Message
				7561795.00	Inbox Message

Provided your query returned enough data you are able to view multiple pages of output.

Adding and Calling Additional Layout Sections

To make the output more readable, you can add column headings and page numbers to the HeadPageSection layout section.

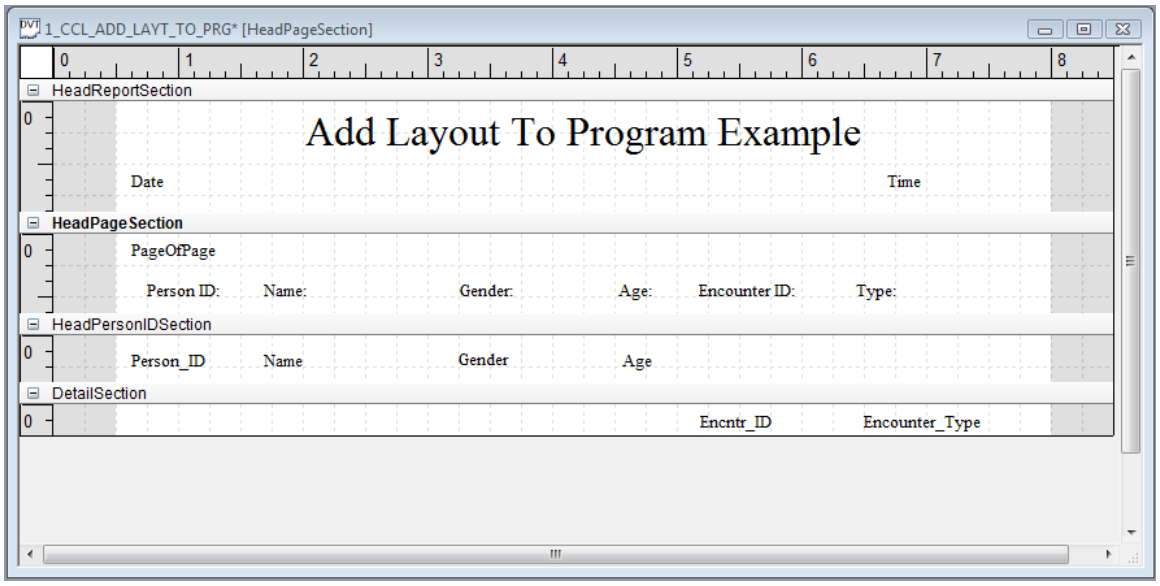
1. Close the output and return to the layout.

2. Use the Text Tool  to add a field to the HeadPageSection to display the special Rpt_PageOfPage variable created by Layout Builder. In the Properties dialog box, modify the Name property to **PageOfPage** and the Source to **Rpt_PageOfPage**.

3. Use the Label Tool  to place the column headings Person ID:, Name:, Gender:, Age:, Encounter ID:, and Type: in the HeadPageSection of your layout.

4. Use the vertical resize to move the bottom of the HeadPageSection up to just underneath the items you just added.

5. Save the layout. Your layout should look similar to the following example:



6. Add the following code to your reportwriter Head Page section to render the HeadPageSection layout section:

```
d0 = HeadPageSection(rpt_render)
```

Your source code should now resemble the following example:

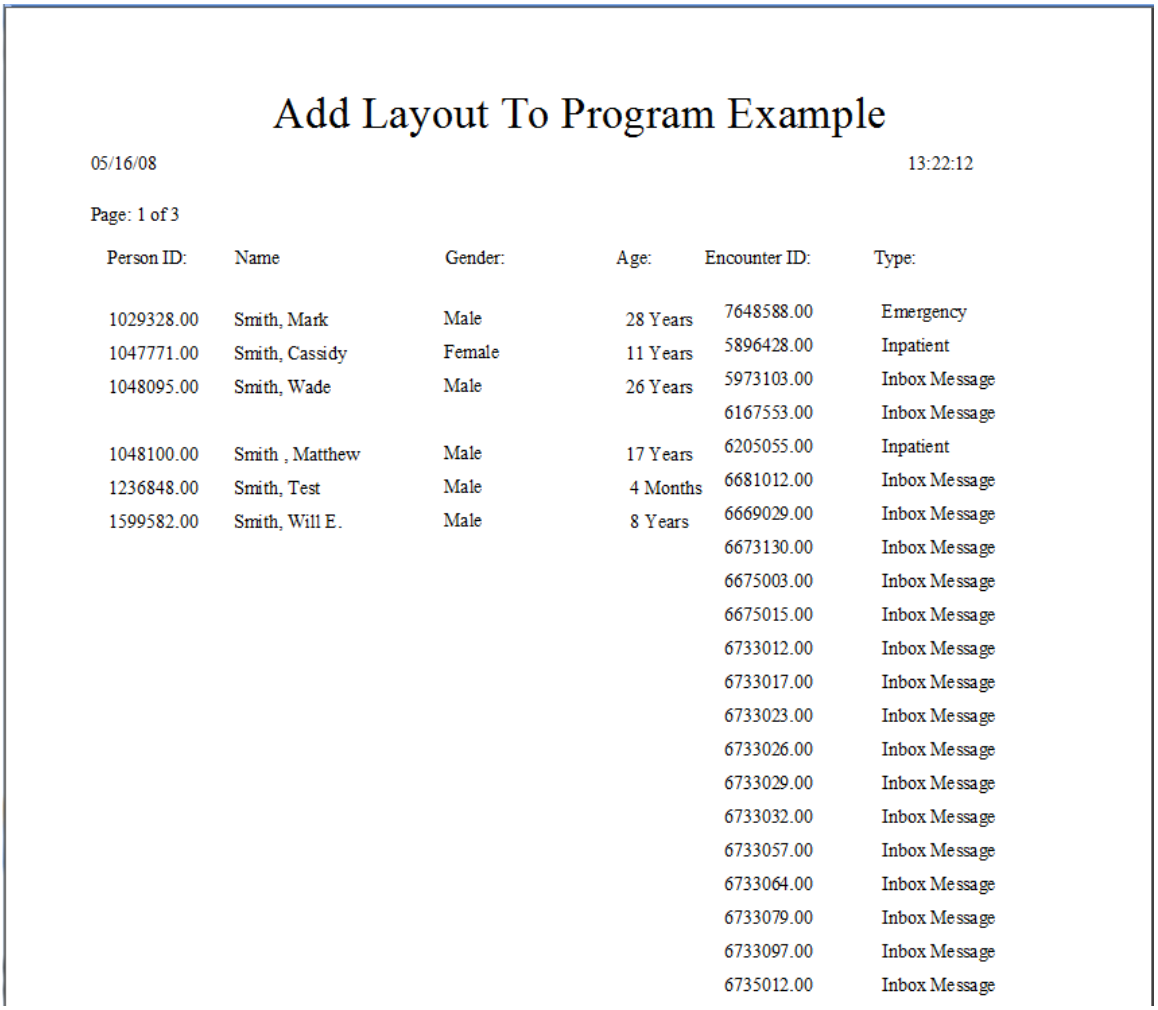
```
drop program 1_1_ccl_add_layt_to_prg go
create program 1_1_ccl_add_layt_to_prg

prompt
  "Output to File/Printer/MINE" = "MINE"
  , "Enter Last Name" = " "

with OUTDEV, LName

execute reportrtl
%i ccluserdir: 1_your_initials_add_layt_to_prg.dvl
;%i must be in the first two columns of the source code file.
;Assuming you changed the program name in your file to
;1_your_initials_add_layt_to_prg.
;set a variable to initializereport(0)
set d0 = InitializeReport(0)
```

```
SELECT INTO "NL:"  ;$OUTDEV
 P.PERSON_ID,
 P.NAME_FULL_FORMATTED,
 P_SEX_DISP = UAR_GET_CODE_DISPLAY( P.SEX_CD ),
 AGE = CNVTAGE(P.BIRTH_DT_TM, E.REG_DT_TM,0),
 E.ENCNTR_ID,
 E_ENCNTR_TYPE_CLASS_DISP =
   UAR_GET_CODE_DISPLAY( E.ENCNTR_TYPE_CLASS_CD )
FROM
 PERSON  P,
 ENCOUNTER  E
PLAN P WHERE  P.NAME_LAST_KEY =  $LNAME
JOIN E WHERE  P.PERSON_ID =  E.PERSON_ID
ORDER BY
 P.PERSON_ID,
 E.ENCNTR_ID
Head report
 F1 = p.name_full_formatted
 F2 = e.encntr_id
 d0 = HeadReportSection(Rpt_Render)
Head Page
 if( curpage > 1)
  d0 = PageBreak(0)
 endif
 d0 = HeadPageSection(rpt_render)
Head P.Person_ID
 d0 = HeadPersonIDSection(Rpt_Render)
Detail
 if(_YOffset + DetailSection(Rpt_CalcHeight) > 10.5)
  Break
 endif
 d0 = DetailSection(Rpt_Render)

WITH MAXREC = 1000 , NOCOUNTER, SEPARATOR=" ", FORMAT

set d0 = FinalizeReport($OutDev)

end
go
```

7. After adding the commands to your source code, save the file and include/compile it.

8. Execute the program as a prompt program.

9. Use MINE as the output device and, when prompted, enter the last name of a person that you know has encounters. You can expect your output to be similar to the following screen:



Add Layout To Program Example

05/16/08 13:22:12

Page: 1 of 3

Person ID:	Name	Gender:	Age:	Encounter ID:	Type:
1029328.00	Smith, Mark	Male	28 Years	7648588.00	Emergency
1047771.00	Smith, Cassidy	Female	11 Years	5896428.00	Inpatient
1048095.00	Smith, Wade	Male	26 Years	5973103.00	Inbox Message
				6167553.00	Inbox Message
1048100.00	Smith , Matthew	Male	17 Years	6205055.00	Inpatient
1236848.00	Smith, Test	Male	4 Months	6681012.00	Inbox Message
1599582.00	Smith, Will E.	Male	8 Years	6669029.00	Inbox Message
				6673130.00	Inbox Message
				6675003.00	Inbox Message
				6675015.00	Inbox Message
				6733012.00	Inbox Message
				6733017.00	Inbox Message
				6733023.00	Inbox Message
				6733026.00	Inbox Message
				6733029.00	Inbox Message
				6733032.00	Inbox Message
				6733057.00	Inbox Message
				6733064.00	Inbox Message
				6733079.00	Inbox Message
				6733097.00	Inbox Message
				6735012.00	Inbox Message

Since you are creating the layout section subroutines using Layout Builder and then calling them in your reportwriter sections, you can use any of *Discern Explorer* commands allowed in a reportwriter section, such as to create variables, load record structure elements, and so on. You can then reference those items in the layout sections. For example, suppose you wanted to display a count of the number of encounters for each person. This can be accomplished by using the aggregate Count() function in the Foot P.Person_ID reportwriter section to set a local reportwriter variable, and then by using that variable as the source for a field in the layout.

10. Close the output and return to your source code file.

11. Switch to Layout Builder and click in the DetailSection to ensure it is active.

12. From the Edit menu, select Insert > Section to add a new section to the layout. A new layout section is inserted before the DetailSection.

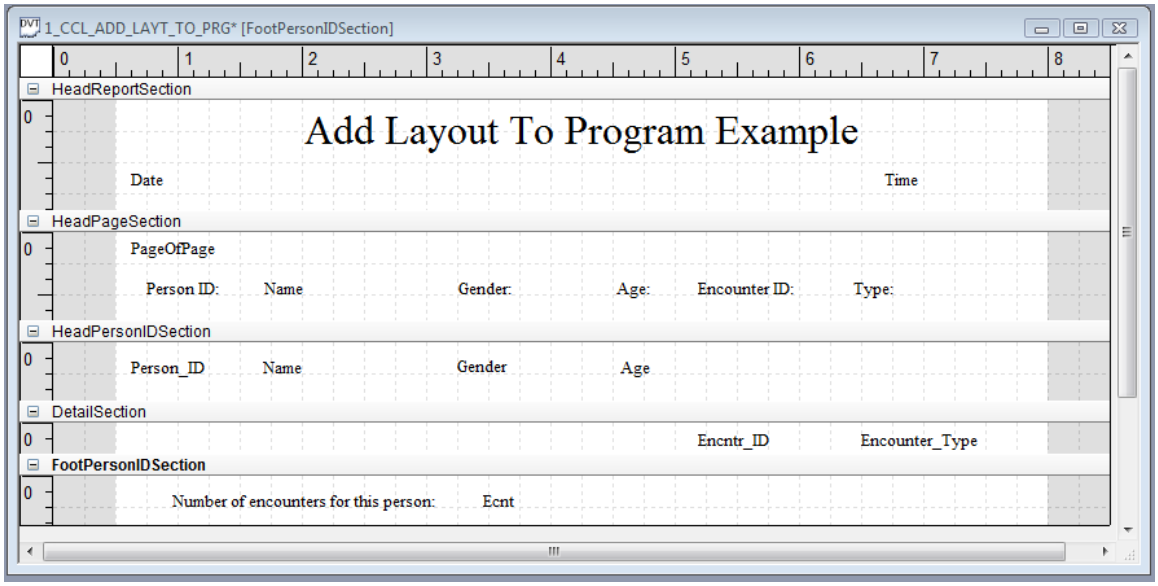
13. Move the new section below the DetailSection by placing your cursor on the rule of the new section and drag the cursor to the middle of the DetailSection's ruler.

14. In the Properties Dialog box, modify the Name property to **FootPersonIDSection**.

15. Use the Label Tool  to add the following text to the FootPersonIDSection on your layout: **Number of encounters for this person**.

16. Use the Text Tool  to add a field to the FootPersonIDSection, and in the Properties dialog box modify the Name property to **Ecnt** and the Source to **Ecnt** (Ecnt is the name of a variable you will create in the Foot P.Person_ID reportwriter section of your source code file using the Count() function).

17. Use the vertical resize to move the bottom of the FootPersonIDSection up to just below the items you have added; leave a little blank space after these items to ensure a better display of the output. Your layout should look similar to the following example:



18. Save the layout.

19. Add the following code after the Detail reportwriter section:

```
Foot p.person_id
 ecnt = count(e.encntr_id)
 d0 = FootPersonIDSection(Rpt_Render)
Foot Page
 row +0
Foot Report
 row +0
```

Notice in the example above that the Foot Page and Foot Report sections are added to conform to the *Discern Explorer* recommendation of having a matching foot reportwriter section for every head reportwriter section.

Your source code should resemble the following example:

```
drop program 1_ccl_add_layt_to_prg go
create program 1_ccl_add_layt_to_prg

prompt
```

```
 "Output to File/Printer/MINE" = "MINE"
 , "Enter Last Name" = " "

with OUTDEV, LName

execute reportrtl
%i ccluserdir: 1_your_initials_add_layt_to_prg.dvl
;%i must be in the first two columns of the source code file.
;Assuming you changed the program name in your file to
;1_your_initials_add_layt_to_prg.
;set a variable to initializereport(0)
set d0 = InitializeReport(0)

SELECT INTO "NL:"  ;$OUTDEV
 P.PERSON_ID,
 P.NAME_FULL_FORMATTED,
 P_SEX_DISP = UAR_GET_CODE_DISPLAY( P.SEX_CD ),
 AGE = CNVTAGE(P.BIRTH_DT_TM, E.REG_DT_TM,0),
 E.ENCNTR_ID,
 E_ENCNTR_TYPE_CLASS_DISP =
   UAR_GET_CODE_DISPLAY( E.ENCNTR_TYPE_CLASS_CD )
FROM
 PERSON  P,
 ENCOUNTER  E
PLAN P WHERE  P.NAME_LAST_KEY =  $LNAME
JOIN E WHERE  P.PERSON_ID =  E.PERSON_ID
ORDER BY
 P.PERSON_ID,
 E.ENCNTR_ID
Head report
 F1 = p.name_full_formatted
 F2 = e.encntr_id
 d0 = HeadReportSection(Rpt_Render)
Head Page
 if( curpage > 1)
  d0 = PageBreak(0)
 endif
 d0 = HeadPageSection(rpt_render)
Head P.Person_ID
 d0 = HeadPersonIDSection(Rpt_Render)
Detail
 if( YOffset + DetailSection(Rpt_CalcHeight) > 10.5)
  Break
 endif
 d0 = DetailSection(Rpt_Render)
Foot p.person_id
 ecnt = count(e.encntr_id)
 d0 = FootPersonIDSection(Rpt_Render)
Foot Page
 row +0
Foot Report
 row +0

WITH MAXREC = 1000 , NOCOUNTER, SEPARATOR=" ", FORMAT

set d0 = FinalizeReport($OutDev)

end
go
```

20. Save the source code file and include/compile it.

21. Execute the program as a prompt program.

22. Use MINE as the output device and, when prompted, enter the last name of a person that you know has encounters. You can expect your output to be similar to the following screen:

Add Layout To Program Example

05/16/08 13:32:00

Page: 1 of 5

Person ID:	Name	Gender:	Age:	Encounter ID:	Type:
1029328.00	Smith, Mark	Male	28 Years	7648588.00	Emergency
Number of encounters for this person: 1					
1047771.00	Smith, Cassidy	Female	11 Years	5896428.00	Inpatient
Number of encounters for this person: 1					
1048095.00	Smith, Wade	Male	26 Years	5973103.00	Inbox Message
				6167553.00	Inbox Message
Number of encounters for this person: 2					
1048100.00	Smith , Matthew	Male	17 Years	6205055.00	Inpatient
Number of encounters for this person: 1					
1236848.00	Smith, Test	Male	4 Months	6681012.00	Inbox Message
Number of encounters for this person: 1					

Executing Programs With Layouts

Up to this point, you have executed your program with a layout as a prompt program in DVDev. Most likely you will create programs with layouts that need to be executed by people who do not have access to DVDev. When you include/compile the source code that calls the subroutines you created using Layout Builder, a *Discern Explorer* program is created in the object library. This program can be executed from Explorer Menu (ExplorerMenu.exe) just like any other *Discern Explorer* program. Simply use the program name when adding a program item to the Explorer Menu.

Moving Your Program With a Layout to Another Environment

Cerner recommends creating and testing all layout programs in a non-production environment. After the program has been created and tested, it can be moved into the production environment. Moving a layout program to a different environment requires you to export the layout program from a source environment (where the program is currently located) into a target environment (where you want the program to be moved).

If you have a front-end file share that can be accessed from both environments, an easy way to move the program with a layout is to open the *program_name*.PRG file in DiscernVisualDeveloper.exe (DVDev) and use the Export command from the File menu to export the file to the front-end file share. The export creates a *program_name*.DVT file. If you have a prompt form associated with the program, the export also creates a *program_name*.DPB file. You can then close DVDev and reopen it connected to the target environment. Use the Import command from the File menu to import the .PRG file from the common front-end file share. The import process uses the .DPB file to re-create the prompt form and the .DVT file to re-create the .DVL file automatically. You are prompted to save the prompt form. When the import completes, include/compile the .PRG file to create the object in the object library.

If you do not have a front-end file share that can be accessed from both environments, but do have access to a different front-end file share from each environment, the process is basically the same. You need to copy the .PRG file, the .DBP file, and the .DVT file to the front-end file share that can be accessed from the target environment before performing the import. The .DPB file is a binary file, so FTP must be used in binary mode when copying the file to the target environment.

If you cannot access front-end file shares, you can move the program and layout to the target environment using the back-end, end file structure. In addition to copying the .PRG file, this method also requires you to export and import the prompt form and the layout. To export the prompt form, select Transfer Objects from the Tools menu to open the Transfer Objects dialog box. Ensure that the Prompt Forms category is selected in the tree on the left side of the Transfer Objects dialog. Enter the name of your layout program in the Source Object: field. Use the Save to Backend… option on the Task menu to save the prompt form to a backend file with a .DPB extension. Use binary FTP to copy the .DPB file to a directory in the target environment. The .DPB file is a binary file, so FTP must be used in binary mode when copying the file to the target environment.

To export the layout, select Transfer Objects from the Tools menu to open the Transfer Objects dialog box. Ensure that the Associated Layouts category under Layouts is selected in the tree on the left side of the Transfer Objects dialog. Enter the name of your layout program in the Source Object: field. Use the Save to Backend… option on the Task menu to save the layout to a back-end file with a .DVT extension. Use ASCII FTP to copy the .DVT file to a directory in the target environment. Use ASCII FTP to copy the .PRG file to a directory in the target environment. Open DVDev in the target environment and select File > Open > Source to open the .DPB file. Opening the .DPB file opens Prompt Builder and imports the prompt form. Save the prompt form and close the Prompt Builder. Use File > Open > Source to open the .DVT file. Opening the .DVT file opens Layout Builder and imports the layout. Save the layout and close the Layout Builder. Use File > Open > Source to open the .PRG file. Place the cursor inside the Create Program/End Go commands and use Tools > Layout Builder to open the layout. Save the layout. Include/Compile the .PRG file to create the program object in the new environment.

Appendix

Layout Builder Sections and Subroutines

Two subroutines are created for each layout section created in Layout Builder. One subroutine is named using the section name as the subroutine name. The second subroutine is created using the section name with **ABS** appended to it as the subroutine name.

A third subroutine is created if HTML is selected as the output type using the section name with HTML appended to as the subroutine name. The subroutine takes zero parameters because there is no height calculation involved for HTML. An HTML section will render immediately after the previous HTML section. There is no concept of absolute X and Y offsets.

The first subroutine that is named using the section name takes an integer parameter of **RPT_RENDER** (value of 0) or **RPT_CALCHEIGHT** (value of 1), which is used to designate whether the section should be rendered. This subroutine is created for ease of use when the margins and automatic y offset incrementation are needed. This automatic incrementation is held in the generated **_YOffset** variable. If the **RPT_CALCHEIGHT** is passed in, the calculated section height is returned without the section being rendered to the page and without the **_YOffset** variable being incremented. When **RPT_RENDER** is passed in, the calculated section height is returned, the section is rendered, and the **_YOffset** variable is incremented.

The second subroutine is named using the section name with **ABS** appended. The first subroutine contains a call to execute the second subroutine. This is the absolute subroutine call and contains all of the logic making the calls to the Report API for the report section generation. The three parameters that this subroutine will take are the calculation flag (**RPT_RENDER** or **RPT_CALCHEIGHT**), as well as the X and Y offsets for the section as real numbers in terms of the measure of units selected for the report (inches or centimeters).

The X,Y offsets are relative to the 0,0 origin at the top left corner of the page. A section can be rendered at any location, but the developer should ensure that the section fits on the page. Any items that do not fit on the page are not printed or displayed.

The third subroutine that is named using the section name with HTML appended, takes zero parameters because there is no height calculation involved for HTML. An HTML section will render immediately after the previous HTML section. There is no concept of absolute X and Y offsets.

Layout Builder represents sections as the full width of the page, based on orientation, although the intended section may not consume the full width of the section and may actually only apply to a column of data in the section that is repeatable across the width of the section. This does not pose any problems since the sections are virtual. It is just a draw back of representing sections as full page widths in Layout Builder.

Layout Builder allows the selection of a page size, orientation, type of report (Postscript, .PDF, Zebra Stripe (ZPL) 200dpi, or Intermec 3400 (IPL) 200 dpi), margins, and unit of measure (inches or centimeters).

Sections are displayed in the orientation selected; page width and height are based on a portrait report. Only left and right margins are represented in Layout Builder based on the orientation of the report since Layout Builder only uses these guides for item placement and visual representation.

The top and bottom margins are not represented in the builder since these margins do not directly apply to each section, but instead to the report in general. The Report API does use these margins in limited circumstances for generation of items that may grow either in width, height, or both, in which case the margins are the boundary in which the item can grow.

Layout Builder also generates other subroutines necessary for report generation, **InitializeReport**, **FinalizeReport**, **PageBreak**, **_LoadImages**, **_CreatePens**, and **_CreateFonts**.

InitializeReport takes an integer dummy parameter reserved for future use which must currently be 0 or unexpected results can occur. **InitializeReport** sets the RPT properties, creates the report context, loads any images, creates any pens needed, creates any fonts used, starts the report, and begins the page. It does so by calling the necessary Layout Builder created subroutines beginning with an underscore ('_').

After the **InitializeReport** has been called, the sections created with Layout Builder can be called in the order they are to be rendered. The **_YOffset** is incremented by the height of the section when it is rendered. By using the **_YOffset** and adding the result from the section to be rendered using the **RPT_CALCHEIGHT** as the parameter, you can compare against the appropriate value to determine page break.

To create a page break for Layout Builder sections, call the **PageBreak** subroutine with a 0 for the reserved parameter (0 must be used or unexpected results can occur). This subroutine calls the appropriate Report API routines to end the page and begin a new page.

When no more pages are to be generated and the report is ready to be finalized, call the **FinalizeReport.** This report passes in a string that contains the queue to print to or the file to generate. This subroutine ends the final page, ends the report, and writes the report to a temporary file. If the report is to be printed to a queue, the file is spooled and deleted.

What You See Is What You Get (WYSIWYG):

Layout Builder attempts to provide a WYSIWYG display. However, the standard fonts that are available in a Windows system and the fonts that are available on a printer can be different. To achieve true WYSIWYG, the printer fonts and the windows display fonts must be the same. Most organizations do not purchase additional display fonts to match the fonts that are shipped with a printer. To help match printer fonts to display fonts Layout Builder provides the Map Fonts option on the Edit menu. This option can be used to manually associate an existing display font to a True Type printer font. For example, the default font for a Zebra printer is most likely not available as a display font, but the Arial Black display font is very close to the Zebra's default font. When creating a layout for a Zebra printer, you should select Map Fonts from the Edit menu and modify the Display font to Arial Black. This provides a very near WYSIWYG display; what you see on your screen when creating the layout is almost exactly what is printed by the Zebra printer.

WYSIWYG does not apply for HTML. Layout Builder sometimes has to *massage* the items in order to make them render. For example, HTML cannot stack one text item on top of another. So, Layout Builder may need to *scoot* a time to create a valid arrangement.

Document Revision History

Revision Number:	Revision Date:	Description:
001	July 16, 2008	Initial release of this *Cerner Millennium* Support Guide. This guide covers *Discern* Layout Builder in the 2007.18 code release of *Cerner Millennium* and therefore reflects the functionality changes that pertain to that release.
002	July 24, 2008	Added missing Doc ID.
003	December 17, 2008	Updated some steps for clarity. In section *Creating HTML Output*, added new info on inserting hyperlinks.
004	March 11, 2009	Added a new section covering the use of *Sub Reports*.