argos

Enterprise Reporting Solution

Argos Advanced Training - DataBlock Designer Workbook

Document Version 1.0

Last Updated 2.11.2014

Table of Contents

Introduction

Welcome to Advanced DataBlock Designer Training

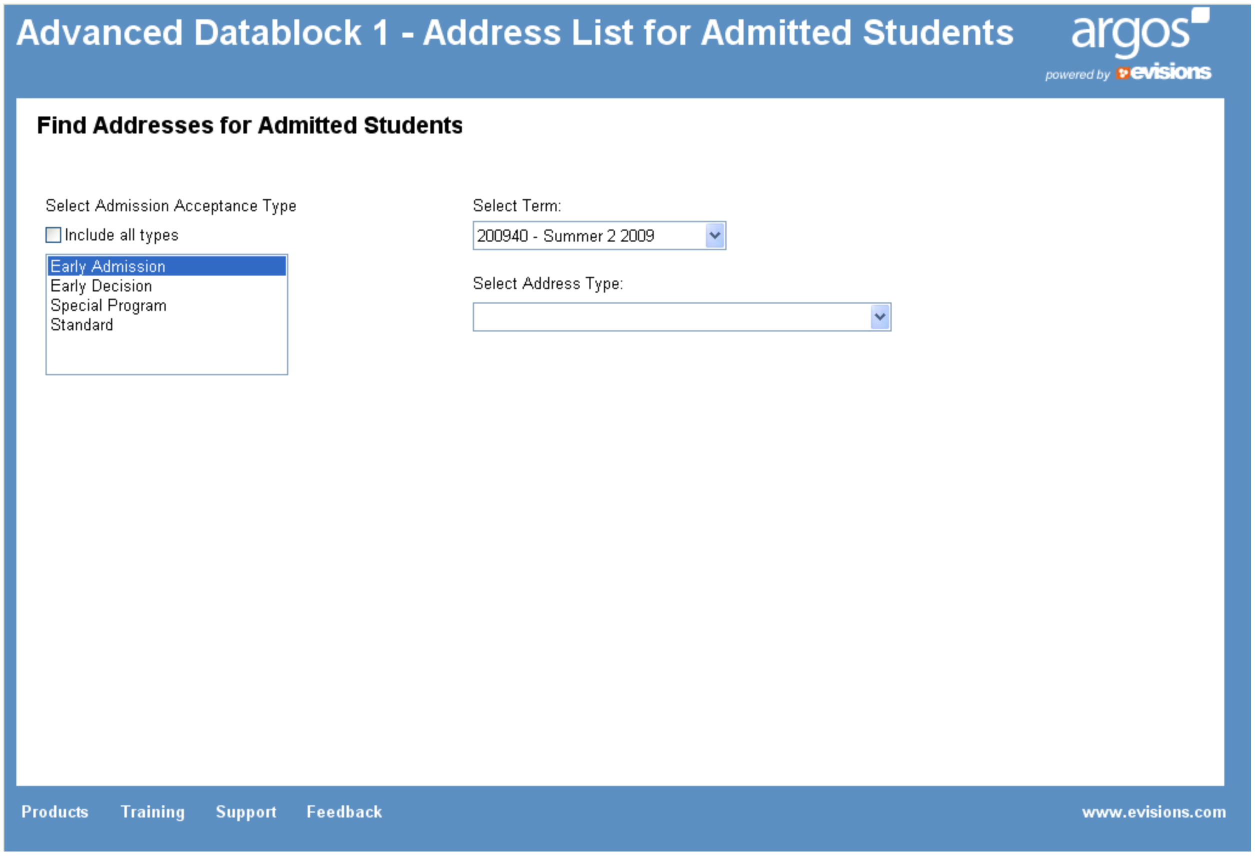
Welcome to the Advanced DataBlock Designer training sessions for Argos. As you move through the exercises that follow, you will learn how to use some of the more advanced DataBlock features. These exercises will give you practice using advanced user input controls, charts and drill downs, and OLAP cubes.

Important Note About This Workbook

This is a continuation of the Basic DataBlock Designer Workbook. The exercises in this workbook are based upon the exercises in the Basic Designer workbook. Several of the exercises that follow will build on the DataBlocks you created in the Basic Designer sessions. You will need to have completed all of the Basic Designer exercises before continuing on with this workbook.

NOTE: If you need a refresher on any of the term definitions we use in this workbook, please refer to the glossary in the Basic Designer workbook.

Fourth Exercise: Advanced DataBlock 1 – Address List for Admitted Students



Exercise Description

In this exercise we are going to copy the DataBlock we created in the first exercise from the Basic Workbook and edit it to add more sophisticated functionality. The updated DataBlock will include a new checkbox variable that uses a union to allow users to select all of the options in a list box with one click. You will then add a correlated subquery and a scalar subquery to your report query.

For this exercise, you will only be adding one new variable:

Include all Admission Types – checkbox that a user can check to automatically select all the admission types in the Admissions Type list box

Instructions

Copy the 'Address List for Admitted Students' DataBlock

1. Find the DataBlock you created in the first exercise, Address List for Admitted Students.
   NOTE: If you did not complete Exercise 1, you can request a copy of the DataBlock from the instructor.
2. Right-click on the DataBlock in the **Explorer** tab, and choose **Copy** from the menu.
   NOTE: If the view needs to be refreshed, the Copy function will appear grayed out and won't be available. If this happens, click on a different object (folder or DataBlock) in the Explorer tab and then right-click on the 'Address List for Admitted Students' DataBlock, and try clicking Copy again.
3. Click on your user folder to highlight it.
4. Right-click on your folder, and choose **Paste** from the menu.
5. Click the **Paste** button in the Paste window that appears.
6. In the **Explorer** tab, click the name of the new DataBlock and rename it: 'Advanced Address List for Admitted Students'.
7. Click the **Edit** button to open the DataBlock Designer.

Add the 'Include All Types' checkbox

8. Click the **Add Check Box** tool (the button in the toolbar that shows a box with an 'x' in it)
9. Click on the panel to add the button.
10. On the **Properties** tab:
    a. Next to **Checked Value**, type 'Y'
    b. Next to **Text**, type 'Include all types'
    c. Next to **UnChecked Value**, type 'N'
    d. Next to **Variable Name**, type 'main_CB_AdmitType'
11. Move the Admissions Type list box (main_LB_AcceptCode) down so there is room between it and its label for you to add the checkbox.
12. Move the checkbox under the list box label, 'Select Admission Acceptance Type'.

Modify the SQL in the ‘Admissions Acceptance Type’ list box

We are going to modify the SQL in the Admission Acceptance Type List Box (‘main_LB_AcceptCode’) to use the checkbox we just created as a variable. If the user does not want to restrict the resulting data set by a particular acceptance type, they can check the box to include students of all acceptance types. If they do want to choose (and therefore restrict by) acceptance type, they can leave the box unchecked, and they can choose an acceptance type (or types) from the list box.

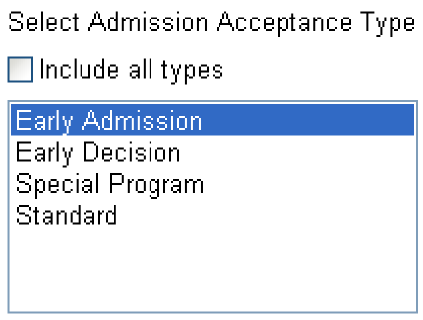


Figure 1: With the box unchecked, the list box shows the list of admission acceptance types, allowing the user to choose which type they want.

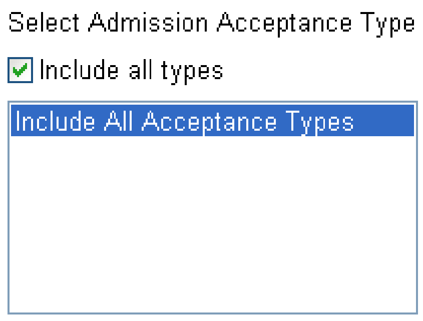
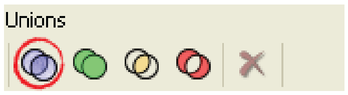

Select Admission Acceptance Type
Include all types
Include All Acceptance Types

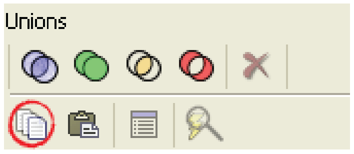
Figure 2: With the box checked, the user does not have to select an admission acceptance type, and all types will be returned in the data set.

13. Before we modify the SQL we need to make one change to the list box. On the **Properties** tab:
    a. Next to **Auto Select**, choose ‘Yes’.
14. Modify the list box SQL query that supplies the list of acceptance types for the list box
    a. Double click on the list box.
        i. Open the Visual Designer.
            (1) Click on the **Conditional Fields (WHERE)** tab:
                (a) Click in the **and/or** row and select: and
                (b) Click in the **Field** row and click the ellipsis button to open the **SQL Editor**:
                    (i) Click the **User-Defined Variable** button (the ABC button on the toolbar) to bring up a list of DataBlock variables:
                        1) Select ‘main_CB_AdmitType’
                        2) Click the **OK** button.
                    (ii) Back in the **SQL Editor** window, click the **OK** button again.
                (c) Click in the **Condition** row and type: = ’N’
            (2) Click on the **Ordering (ORDER BY)** tab:
                (a) Click in the **Table** row in the first column, where it says ‘STVADMT,’ and choose ‘<calculated>’
                (b) In the **Field** row, type ‘2’
            (3) Click on the **Show Unions** button at the top of the window, which will open the **Unions** panel on the left side of the screen:
            NOTE: If you would like more information on unions, please consult a SQL reference guide.
                (a) Click on the **Add Union** button (the blue circle icon on the **Union** tab toolbar).



(b) Select 'Main Query' from the list in the **Union** tab.

(c) Click the **Copy** button (the icon with the stack of pages on the **Union** tab toolbar).



NOTE: DO NOT click on the Copy button in the main toolbar; that will copy the entire Visual Design.

(d) Click the 'Union' query in the list (below where it says 'Main Query') to select it.

(e) Click the **Paste** button (the clipboard icon on the **Union** tab toolbar), to paste a copy of the main query into your union query.

(f) With the 'Union' query highlighted, click on the **Visible Fields (SELECT)** tab:

  (i) Click in the **Table** row in the first column, where it says 'STVADMT,' and choose '<calculated>'

  (ii) Click in the **Field** row of the first column and type: 'Dummy Code'

  NOTE: In this case, you'll include the single quotes when you type the text above.

  (iii) Click in the **Table** row of the second column and choose '<calculated>'

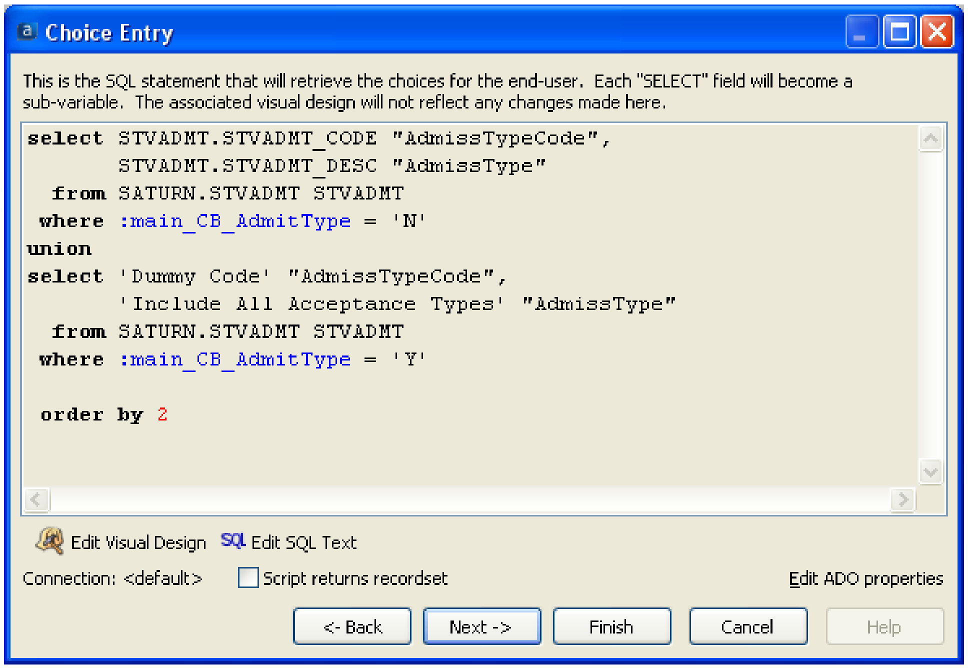
  (iv) Click in the **Field** row of the second column and type 'Include all Acceptance Types' (also including the single quotes).

(g) Click on the **Conditional Fields (WHERE)** tab:

  (i) In the **Condition** row, change it to: = 'Y'

ii. Click the **OK** button to exit the Visual Designer.

b. Review the SQL query you just built. It should look like this:



This is the SQL statement that will retrieve the choices for the end-user. Each "SELECT" field will become a sub-variable. The associated visual design will not reflect any changes made here.

```
select STVADMT.STVADMT_CODE "AdmissTypeCode",
       STVADMT.STVADMT_DESC "AdmissType"
  from SATURN.STVADMT STVADMT
 where :main_CB_AdmitType = 'N'
union
select 'Dummy Code' "AdmissTypeCode",
       'Include All Acceptance Types' "AdmissType"
  from SATURN.STVADMT STVADMT
 where :main_CB_AdmitType = 'Y'

 order by 2
```

Connection: <default>   ☐ Script returns recordset   Edit ADO properties

c. Click the **Next** button.

   d. Click the **OK** button on the **Test Values** window.
   e. On the next screen, provided there were no errors, hit the **Finish** button.

15. Commit and test your changes.
    NOTE: At this point you cannot test your report query. We have changed the variable the report query is based on, and we have not yet updated the report query to match. For the moment, you only need to test that the checkbox and Acceptance Type list box are behaving correctly. The first row in the list box should be highlighted, indicating that the box is automatically selecting the first result. With the checkbox unchecked, you should see a listing of acceptance types. With the checkbox checked, the list box should simply say 'Include All Acceptance Types'.

Modify the Report Query to include the 'Include All Types' checkbox

16. Click on the **Report Query – Visual Design** tab:
    a. Click on the **Conditional Fields (WHERE)** tab:
       i. Find the 'SARADAP_ADMT_CODE' condition and delete it by clicking the **X** button at the top of the column.
       ii. In the lower left-hand corner of the tab, click on the **+** button to create a conditional group:
           (1) You now have a new conditional group under your root query. Select the group, double-click 'SARADAP_ADMT_CODE' in the 'SARADAP' table to add it to the conditional group.
           (2) Click in the **Condition** row and click the ellipsis button to open the **SQL Editor:**
               (a) Click the **User-Defined Variable** button (the ABC button on the toolbar) to bring up a list of DataBlock variables:
                   1) Select 'main_LB_AcceptCode.AdmissTypeCode'
                   2) Click the **OK** button.
           (3) In the second column of your conditional group, click in the **and/or** row and select 'or'
               NOTE: Make sure you select 'or' not 'and' or this condition won't work.
           (4) Click in the **Field** row and click the ellipsis button to open the **SQL Editor:**
               (a) Click the **User-Defined Variable** button (the ABC button on the toolbar) to bring up a list of DataBlock variables:
                   (i) Select 'main_CB_AdmitType'
                   (ii) Click the **OK** button.
               (b) Back in the **SQL Editor**, click the **OK** button.
           (5) Click in the **Condition** row and type: ='Y'

17. Commit and test your changes.
    NOTE: When you test, if you check the checkbox you should get all types of admission types. If you uncheck the box you should only get students that match the admit types that you selected. Close the preview when done.

Modify the Report Query to create a scalar subquery

We're going to add a scalar subquery to the SELECT clause of the Report Query. Specifically, we are going to add 'STVLEVL_DESC' to the SELECT clause so we can see the student level description. Using a scalar subquery allows us to do that without adding the entire table just for that one field.
NOTE: For more information on scalar subqueries, please consult a SQL reference guide.

18. Click on the **Report Query – Visual Design** tab:
    a. Click on the **Visible Fields (SELECT)** tab, and scroll to the empty column all the way to the right.
    b. Click in the **Table** row and select '<calculated>'
    c. Click in the **Field** row and click the ellipsis button to bring up the **SQL Editor**:
        i. Type the following:

```
(select STVLEVL_DESC
from STVLEVL
where SARADAP_LEVL_CODE=STVLEVL_CODE and rownum=1)
```

NOTE: Line breaks and spacing do not affect the way the SQL query runs. You can type everything on one row or several, depending on what is more readable.

        ii. Click the **OK** button to exit the **SQL Editor**
    d. Click in the **As** row and change 'calc1' to 'LevelDesc'

19. Commit and test your changes.
NOTE: Make sure the subquery is working by checking that the student level description is appearing.

Modify the Report Query to create a correlated subquery

We're also going to add a correlated subquery to the WHERE clause of the Report Query. We want to modify the resulting data set to only use the information from the student's current application.
NOTE: For more information on subqueries, please consult a SQL reference guide.

20. Click on the **Report Query – Visual Design** tab:
    a. Click the **Sub Query** button at the top of the tab, which will open up a new, blank, visual design window:
        i. Add table: 'SARADAP'
        NOTE: The alias for the new table will default to 'SARADAP1' in order to distinguish it from the 'SARADAP' table that has already been added to the main query.
        ii. Add 'SARADAP_APPL_NO' to the **Visible Fields (SELECT)** tab.
            (1) Click on the **Summing** button on the left side of the tab, and choose 'Max' in the **Summing** row
        iii. Add 'SARADAP_PIDM' to the **Conditional Fields (WHERE)** tab.
            (1) Click in the **Condition** row and click the ellipsis button to open the **SQL Editor**:
                (i) Click the **Insert Field** button (the green neutron on the toolbar) to open the **Pick a Field** window:
                    1) Click the **+** button to expand out 'SARADAP'
                    NOTE: Be very careful to choose 'SARADAP', and NOT 'SARADAP1' in this step.
                    2) Click 'SARADAP_PIDM' to select it.

3) Click the **OK** button.

(ii) Back in the **SQL Editor**, click the **OK** button to close the window.

iv. Add 'SARADAP_TERM_CODE_ENTRY' to the **Conditional Fields (WHERE)** tab.

(1) Click in the **Condition** row and click the ellipsis button to open the **SQL Editor**:

(i) Click the **Insert Field** button (the green neutron on the toolbar) to open the **Pick a Field** window:

1) Click the **+** button to expand out 'SARADAP'
NOTE: Again, be careful to choose 'SARADAP', and NOT 'SARADAP1'.

2) Click 'SARADAP_TERM_CODE_ENTRY' to select it.

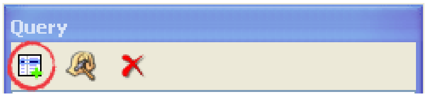
3) Click the **OK** button.

(2) Back in the **SQL Editor**, click the **OK** button to close the window.

v. Click the **OK** button to close out of the **Sub Query** window.

b. You will now see a new box, called 'Query', has been added to the table area of the **Report Query – Visual Design** tab.
NOTE: If you want to rename the new 'Query' box—which can come in handy when you start adding multiple subqueries—click on the ***Edit Query Properties*** *button (the hardhat and hammer) on the 'Query' box toolbar. In the bottom left hand corner is the* ***Rename*** *button. Click the button to rename the query box. Renaming the query box has no impact on the SQL, but it does make it easier to manage if you have large numbers of subqueries.*

i. Click the **Add to Conditional Fields Tree** button (the window icon with the green plus on the 'Query' box toolbar)

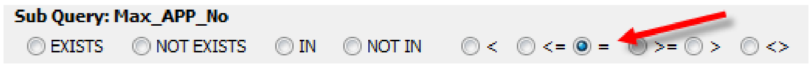


to add your correlated subquery to the **Conditional Fields (WHERE)** tab.

ii. On the **Conditional Fields (WHERE)** tab:

(1) In the conditional fields tree on the left, you'll see a new condition called '<query>'. Click to select it.

(2) Click the radio button for '=',


Sub Query: Max_APP_No
EXISTS NOT EXISTS IN NOT IN < <= [✓ =] >= > <>

which will display a new column of information about your new condition.

(3) Click in the **Field** row, and click the ellipsis button to open the **SQL Editor**:

(a) Click the **Insert Field** button (the green neutron on the toolbar) to open the **Pick a Field** window:

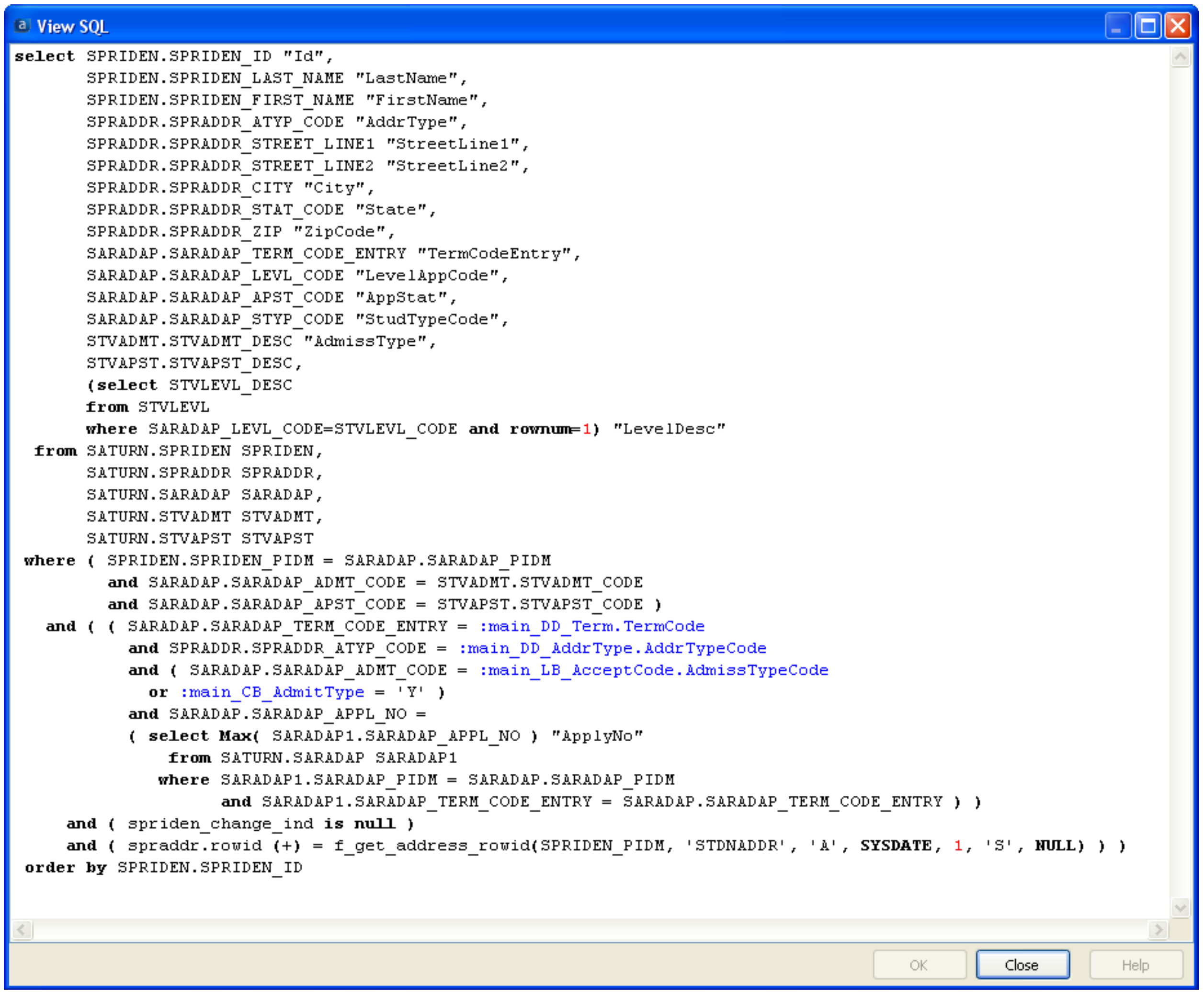
1) Click the **+** button to expand out 'SARADAP'

2) Click 'SARADAP_APPL_NO' to select it.

3) Click the **OK** button.

(b) Back in the **SQL Editor**, click the **OK** button to close the window.

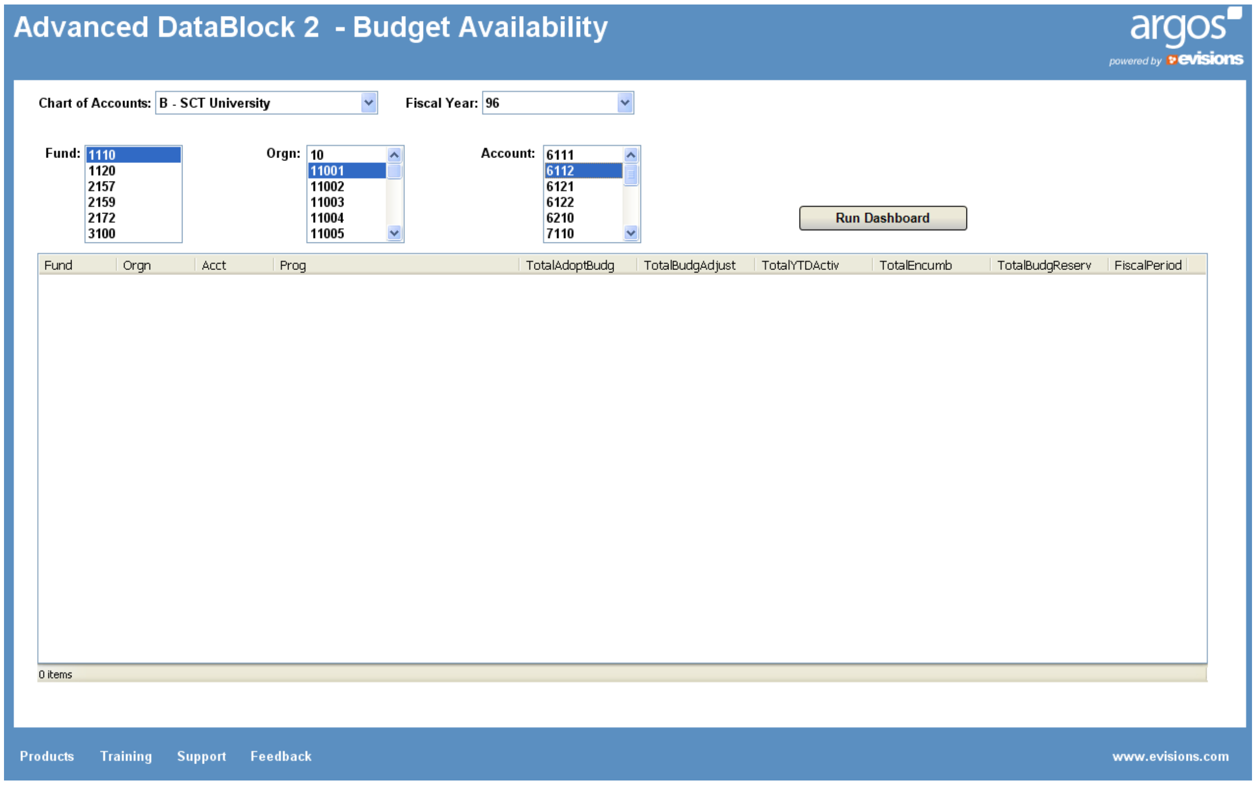
21. Click the **View SQL** button and review your finished Report Query. It should look like this:



```
select SPRIDEN.SPRIDEN_ID "Id",
       SPRIDEN.SPRIDEN_LAST_NAME "LastName",
       SPRIDEN.SPRIDEN_FIRST_NAME "FirstName",
       SPRADDR.SPRADDR_ATYP_CODE "AddrType",
       SPRADDR.SPRADDR_STREET_LINE1 "StreetLine1",
       SPRADDR.SPRADDR_STREET_LINE2 "StreetLine2",
       SPRADDR.SPRADDR_CITY "City",
       SPRADDR.SPRADDR_STAT_CODE "State",
       SPRADDR.SPRADDR_ZIP "ZipCode",
       SARADAP.SARADAP_TERM_CODE_ENTRY "TermCodeEntry",
       SARADAP.SARADAP_LEVL_CODE "LevelAppCode",
       SARADAP.SARADAP_APST_CODE "AppStat",
       SARADAP.SARADAP_STYP_CODE "StudTypeCode",
       STVADMT.STVADMT_DESC "AdmissType",
       STVAPST.STVAPST_DESC,
       (select STVLEVL_DESC
       from STVLEVL
       where SARADAP_LEVL_CODE=STVLEVL_CODE and rownum=1) "LevelDesc"
  from SATURN.SPRIDEN SPRIDEN,
       SATURN.SPRADDR SPRADDR,
       SATURN.SARADAP SARADAP,
       SATURN.STVADMT STVADMT,
       SATURN.STVAPST STVAPST
 where ( SPRIDEN.SPRIDEN_PIDM = SARADAP.SARADAP_PIDM
         and SARADAP.SARADAP_ADMT_CODE = STVADMT.STVADMT_CODE
         and SARADAP.SARADAP_APST_CODE = STVAPST.STVAPST_CODE )
   and ( ( SARADAP.SARADAP_TERM_CODE_ENTRY = :main_DD_Term.TermCode
           and SPRADDR.SPRADDR_ATYP_CODE = :main_DD_AddrType.AddrTypeCode
           and ( SARADAP.SARADAP_ADMT_CODE = :main_LB_AcceptCode.AdmissTypeCode
             or :main_CB_AdmitType = 'Y' )
           and SARADAP.SARADAP_APPL_NO =
           ( select Max( SARADAP1.SARADAP_APPL_NO ) "ApplyNo"
               from SATURN.SARADAP SARADAP1
              where SARADAP1.SARADAP_PIDM = SARADAP.SARADAP_PIDM
                    and SARADAP1.SARADAP_TERM_CODE_ENTRY = SARADAP.SARADAP_TERM_CODE_ENTRY ) )
     and ( spriden_change_ind is null )
     and ( spraddr.rowid (+) = f_get_address_rowid(SPRIDEN_PIDM, 'STDNADDR', 'A', SYSDATE, 1, 'S', NULL) ) )
 order by SPRIDEN.SPRIDEN_ID
```

22. Commit and test your changes.
    NOTE: At this point, your results should no longer include duplicate students.
23. Your first Advanced DataBlock is complete. Click the **Close** button to exit the DataBlock Designer.

Fifth Exercise: Advanced DataBlock 2 – Budget Availability


Advanced DataBlock 2 - Budget Availability
argos
powered by evisions
Chart of Accounts: B - SCT University
Fiscal Year: 96
Fund: 1110 1120 2157 2159 2172 3100
Orgn: 10 11001 11002 11003 11004 11005
Account: 6111 6112 6121 6122 6210 7110
Run Dashboard
Fund Orgn Acct Prog TotalAdoptBudg TotalBudgAdjust TotalYTDActiv TotalEncumb TotalBudgReserv FiscalPeriod
0 items
Products Training Support Feedback
www.evisions.com

Exercise Description

In this exercise we are going to copy the DataBlock we created in the second exercise and modify it to make the 'Run Dashboard' button's visibility dependent on the 'Account' list box variable. That way, the 'Run Dashboard' button will appear only once the end user has chosen a value for all five variables.

Instructions:

Copy the 'Budget Availability' DataBlock

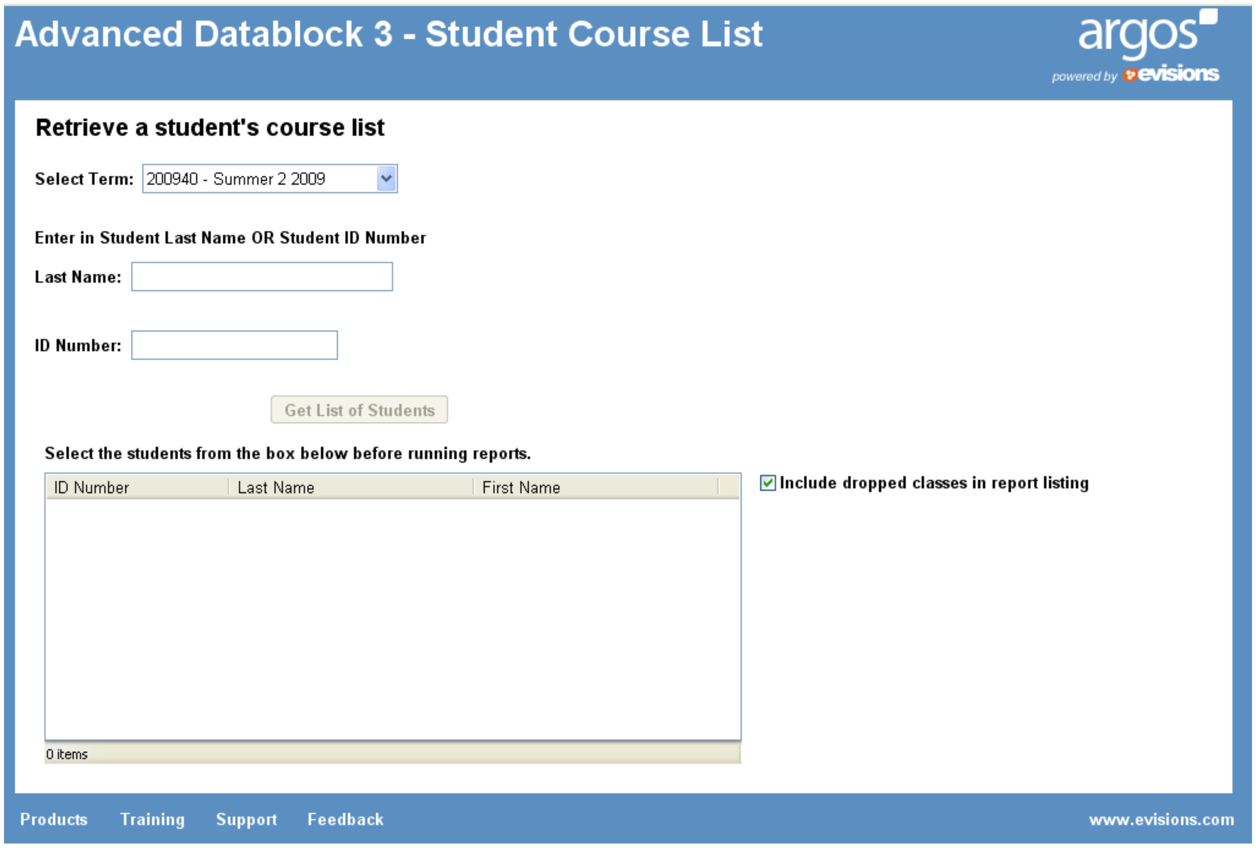
24. Find the DataBlock you created in the second exercise, 'Budget Availability'.
    NOTE: If you did not complete Exercise 2, you can request a copy of the DataBlock from the instructor.
25. Right-click on the DataBlock in the **Explorer** tab, and choose **Copy** from the menu.

26. Click on your user folder to highlight it.
27. Right-click on your folder, and choose **Paste** from the menu.
28. Click the **Paste** button in the Paste window that appears.
29. In the **Explorer** tab, click the name of the new DataBlock and rename it: 'Advanced Budget Availability'
30. Click the **Edit** button to open the DataBlock Designer.

Make the visibility of the 'Run Dashboard' button dependent on the 'Account' list box

31. Click the 'Run Dashboard' button to select it.
32. Click the **Properties** tab:
    a. Next to **Visible**, select 'main_LB_Acct.AccountCode'
33. Commit and test your changes.
    NOTE: Choose a value for the five required variables to make sure the 'Run Dashboard' button becomes visible.
34. Your second Advanced DataBlock is complete. Click the **Close** button to exit the DataBlock Designer.

Sixth Exercise: Advanced DataBlock 3 – Student Course List


Advanced Datablock 3 - Student Course List
argos
powered by evisions
Retrieve a student's course list
Select Term: 200940 - Summer 2 2009
Enter in Student Last Name OR Student ID Number
Last Name:
ID Number:
Get List of Students
Select the students from the box below before running reports.
ID Number
Last Name
First Name
0 items
Include dropped classes in report listing
Products
Training
Support
Feedback
www.evisions.com

Exercise Description

In this exercise we are going to copy the DataBlock we created in the third exercise and modify it so that the 'Get List of Students' button is only enabled once the end user has entered a value in the 'Last Name' or 'ID Number' text box. You'll add a SQL Variable to control whether the button is enabled or not.

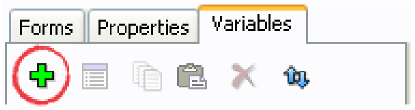
Instructions:

Copy the 'Student Course List' DataBlock

35. Find the DataBlock you created in the third exercise, Student Course List.
    NOTE: If you did not complete Exercise 3, you can request a copy of the DataBlock from the instructor.
36. Right-click on the DataBlock in the **Explorer** tab, and choose **Copy** from the menu.
37. Click on your user folder to highlight it.
38. Right-click on your folder, and choose **Paste** from the menu.
39. Click the **Paste** button in the Paste window that appears.
40. In the **Explorer** tab, click the name of the new DataBlock and rename it: 'Advanced Student Course List'
41. Click the **Edit** button to open the DataBlock Designer.

Add a SQL variable to set the enable property for the 'Get List of Students' button

42. Click the **Variables** tab.
43. Click the **Add Variable** button (the green plus sign on the **Variable** tab toolbar).



    a. In the **Choose Variable Type** window, select 'SQL Statement' and click the **OK** button.
    b. Open the Visual Designer:
        i. Add Table: 'Dual'
        ii. Click the **Visible Fields (SELECT)** tab:
            (1) Click in the **Table** row and select '<calculated>'
            (2) Click in the **Field** row and type: 'True'
            NOTE: Make sure you include the single quotes.
            (3) Click in the **As** row and type: TrueValue
        iii. Click the **Conditional Fields (WHERE)** tab
            (1) Click in the **Table** row, select '<calculated>'
            (2) Click in the **Field** row and click the ellipsis button to open the **SQL Editor**:
                (a) Click the **User-Defined Variable** button (the ABC button on the toolbar) to bring up a list of DataBlock variables:
                    (i) Expand out 'main_DD_Term'
                    (ii) Click on ' STVTERM_CODE' to select it.
                    (iii) Click the **OK** button.
                (b) Back in the **SQL Editor** window, click the **OK** button again.
            (3) Click in the **Condition** row and type: is not null
            (4) In the lower left-hand corner of the tab, click the **+** button to create a conditional group.
            (5) You now have a new conditional group under your root query. Click '<group>' to select it:
                (a) Click in the **Table** row of the new column, and select '<calculated>'
                (b) Click in the **Field** row and click the ellipsis button to open the **SQL Editor**:
                    (i) Click the **User-Defined Variable** button (the ABC button on the toolbar) to bring up a list of DataBlock variables:
                        1) Click on 'main_EB_IDNumber' to select it.

2) Click the **OK** button.

(ii) Back in the **SQL Editor** window, click the **OK** button again.

(c) Click in the **Condition** row and type: is not null

(d) Click in the **and/or** row of the next column, and select 'or'

(e) Click in the **Table** row of the next column, and select '<calculated>'

(f) Click in the **Field** row and click the ellipsis button to open the **SQL Editor**:

(i) Click the **User-Defined Variable** button (the ABC button on the toolbar) to bring up a list of DataBlock variables:

1) Click on 'main_EB_LastNm' to select it.

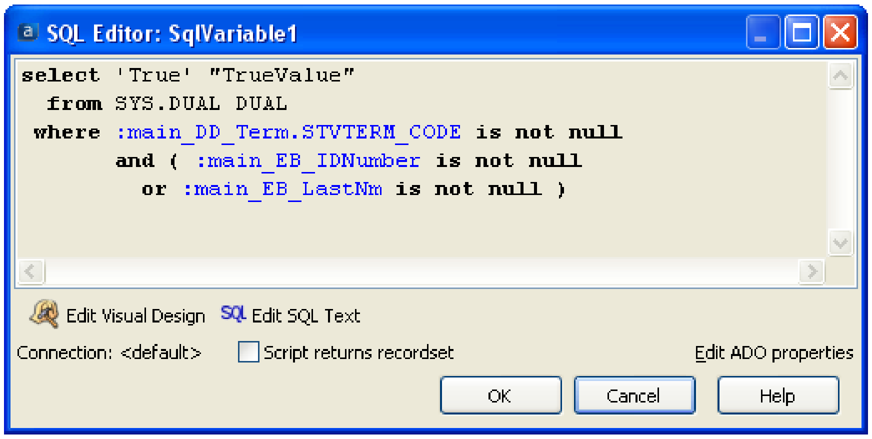
2) Click the **OK** button.

(ii) Back in the **SQL Editor** window, click the **OK** button again.

(g) Click in the **Condition** row and type: is not null

iv. Click the **OK** button to exit the Visual Designer.

c. Review your SQL query. It should look like this:



```
select 'True' "TrueValue"
  from SYS.DUAL DUAL
 where :main_DD_Term.STVTERM_CODE is not null
       and ( :main_EB_IDNumber is not null
         or :main_EB_LastNm is not null )
```

d. Click the **OK** button to and then click the **Yes** button to validate your new query.

e. Click the **OK** button, leaving the Test Values blank.

f. You should see a small window that says:
Results: TrueValue = <null>

g. Click the **OK** button.

44. In the **Variables** tab:

a. Right-click the your new variable, 'SqlVariable1', and choose **Rename Variable** from the menu.

b. Rename your variable by typing: SQL_ButtonEnabled

45. Click the 'Get List of Students' button to select it.

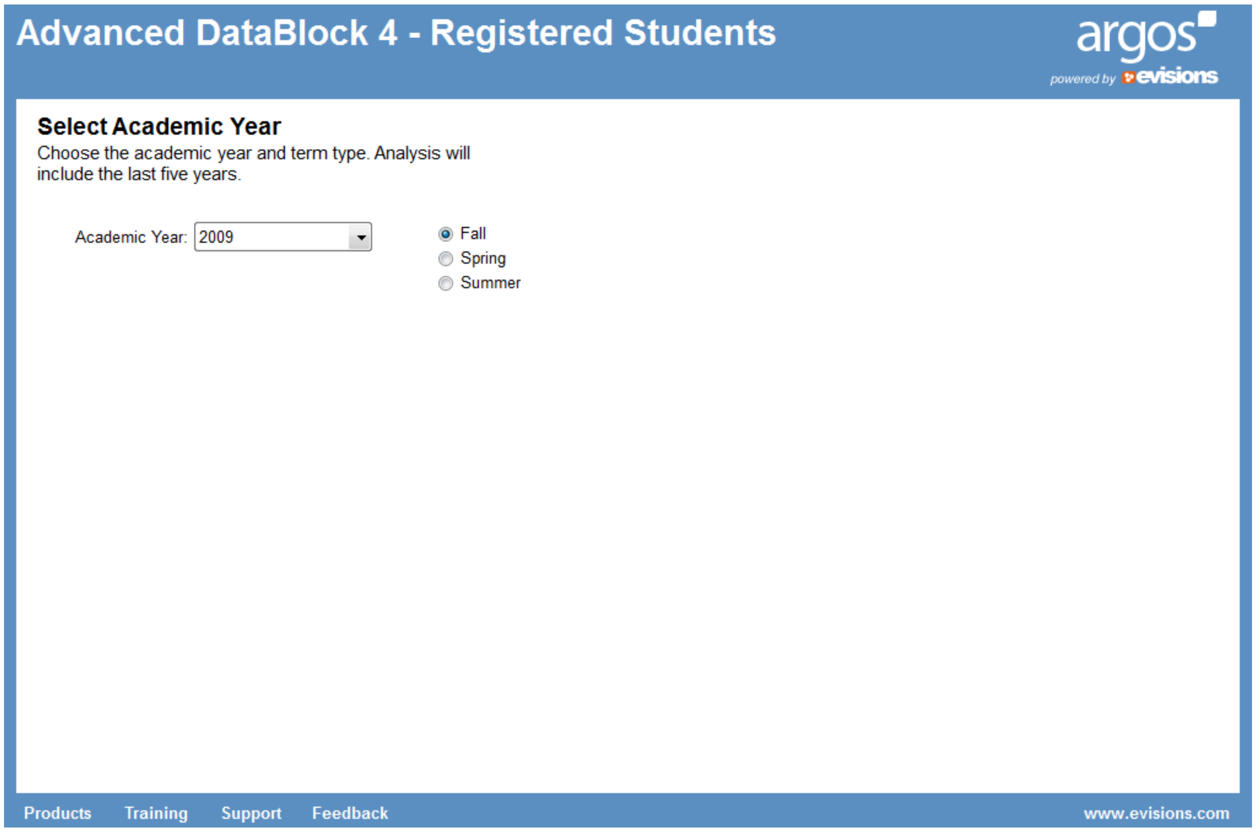
46. Click the **Properties** tab:

a. Next to **Enabled**, select 'SQL_ButtonEnabled.TrueValue'

47. Commit and test your changes.
NOTE: Enter a value for 'Last Name' or 'ID Number' and then click out of the text box to make sure the 'Get List of Students' button becomes enabled.

48. Your third Advanced DataBlock is complete. Click the **Close** button to exit the DataBlock Designer.

Seventh Exercise: Advanced DataBlock 4 – Registered Students


Advanced DataBlock 4 - Registered Students

Select Academic Year

Choose the academic year and term type. Analysis will include the last five years.

Academic Year: 2009

[✓ Fall] Spring Summer

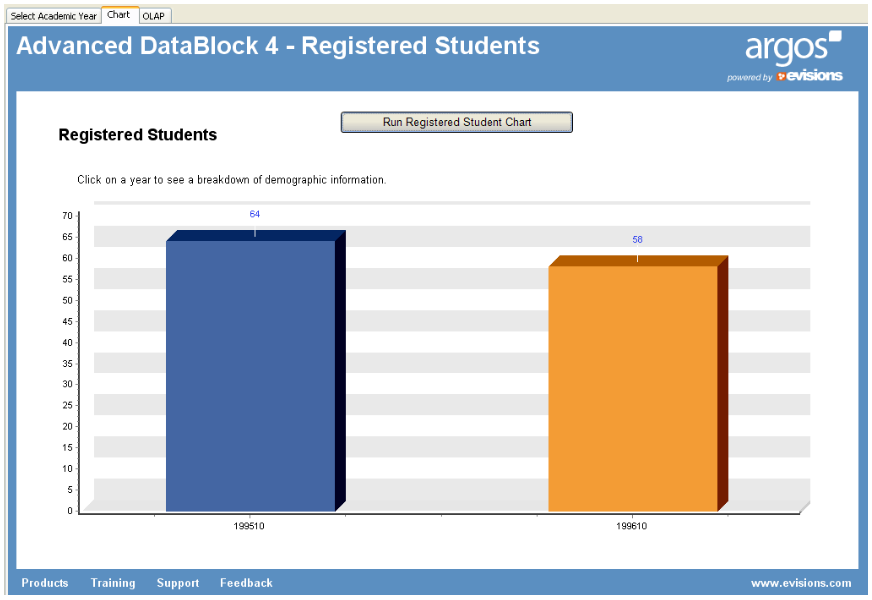
Exercise Description

This DataBlock uses a variety of advanced features to allow a Report Viewer to select a data set (in this case, registered students for a particular year and term type), and see it represented graphically in several different ways (a variety of charts and an OLAP cube).

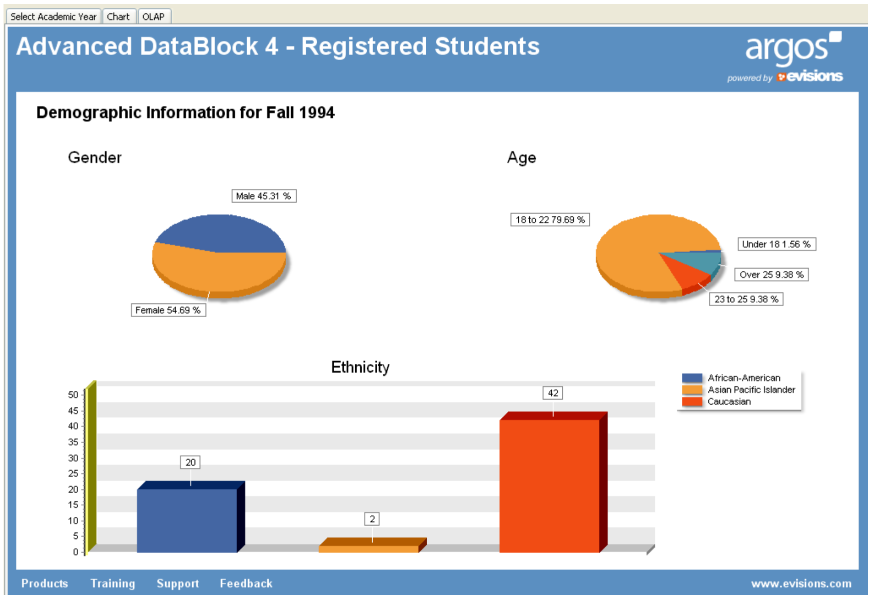
The dashboard for this DataBlock will be made up of four separate forms. On the first form (see example image above), Report Viewers will select an academic year and term type for which they'd like to see data on registered students. The second form in the dashboard will display a chart that shows the enrollment numbers based on the user's selections. From this chart, users will be able to drill down by selecting specific terms to view additional detail charts with demographic information on registered students for that term. The demographic information is displayed on a separate form. The last form will display an OLAP (On Line Analytical Processing) cube, which offers another way to view, interact with, and analyze the requested data set.

Below are examples of what the additional forms in the dashboard will look like:

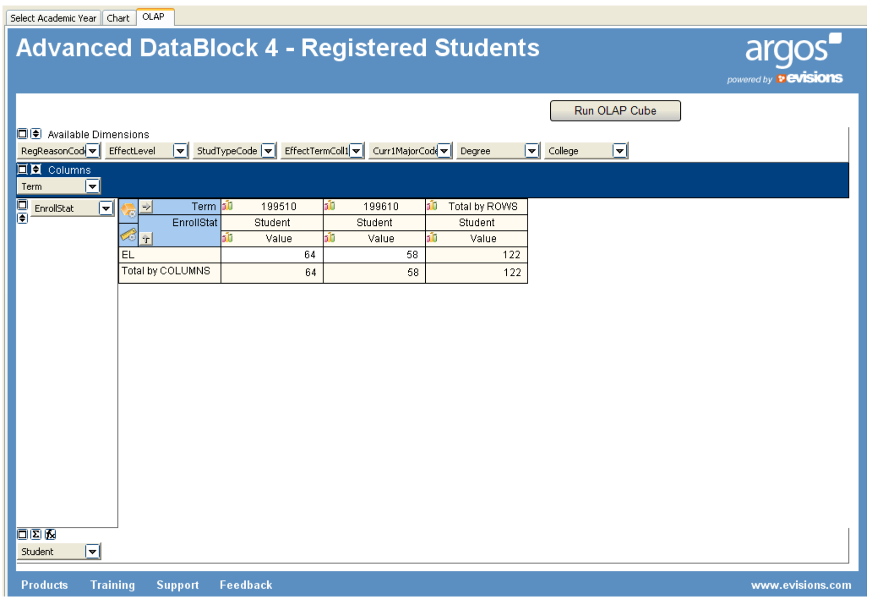
Example Chart Form



Example Chart Detail Form



Example OLAP Cube Form



Term	199510	199610	Total by ROWS
EnrollStat	Student	Student	Student
	Value	Value	Value
EL	64	58	122
Total by COLUMNS	64	58	122

Both the Chart and OLAP forms will automatically display data for the five years leading up to the academic year selected by the user on the first form. The 'Term Type' variable allows users to compare several terms of the same type (i.e. Fall terms from 2005-2010). The dashboard is designed so that the user only needs to make two selections (year and term type) to retrieve the last five years of data for that term type. Also, the variables restrict the end user from inadvertently comparing data from different term types. While users may sometimes want to compare Fall and Spring numbers against each other, that tends to be a rare occasion. It is much more common for users to want to compare like terms.

Here is a description of the different elements that you'll be creating for this DataBlock, and note on which database table(s) they draw from:

Academic Year – drop down variable that returns a list of academic years. For example: 2011, 2012, 2013. *Table used: STVTERM*

Term Type – radio button variable that returns a list of term types. For example Fall, Winter, Spring. *Manually created list, no table used.*

Registered Students – chart that displays a count of registered students grouped by term. This chart will also be used as a variable for other charts. *Table used: SFBETRM*

Age Range – chart that displays a count of registered students grouped by age range. Data displayed is based on a specific term selected in the 'Registered Students' chart. *Tables used: SFBETRM, SPBPERS*

Gender – chart that displays a count of registered students grouped by gender. Data displayed is based on a specific term selected in the 'Registered Students' chart. *Tables used: SFBETRM, SPBPERS*

Ethnicity – chart that displays a count of registered students grouped by ethnicity. Data displayed is based on a specific term selected in the 'Registered Students' chart. *Tables used: SFBETRM, SPBPERS*

Registered Students OLAP Cube – OLAP cube that displays registered student information. *Tables used: SFBETRM, SGBSTDN, STVCOLL*

Instructions

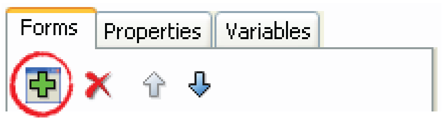
Create a new DataBlock and open it for editing

49. Find your user folder in the folder list on the left-hand side.
50. Right-click on your folder, and choose **New> DataBlock** from the menu.
51. Type in the name of your new DataBlock – 'Registered Students' – and hit the **Enter** key.
52. Click on your new DataBlock to highlight it.
53. On the right side of the screen, under the DataBlock Designer Actions heading, select the ADO being used for your training from the Associated Connection/Pool drop down.
54. Click the **Edit** button to open the DataBlock Designer.

Create additional forms

As with all new DataBlocks, when you open this one up for editing, you will see an empty form called 'Main'. Because our dashboard for this DataBlock includes more than one form, our first step is to create them. Although the end user will only navigate between three of them, we're actually going to want to create four forms total (three primary forms and one that will act as a sub-form).

55. To create a new form, go to the **Forms** toolbar on the left, and click the **Create New Form** button (the form icon with the green plus sign).



NOTE: You should now see a new form in the list called 'Form1'.

56. Click the **Create New Form** button two more times to create 'Form2' and 'Form3'.
57. Right-click on 'Form1' and select **Rename** from the menu.
58. Type 'Chart' to rename your form.
59. Repeat steps 60 & 61 to rename 'Form2' to 'ChartA' and 'Form3' to 'OLAP'.

Configure the form layout

The three main forms the users will be navigating are 'Main', 'Chart', and 'OLAP'. To make it easy to switch between them, we're going to create tabs for each of those forms. The form 'ChartA' will not be a tab, but the users will link to it from within the 'Chart' form to view additional details on information displayed in the chart.

60. On the **Forms** tab, click on the 'Main**'** form to select it.
61. On the **Properties** tab:

a. Next to **Is Tab**, select 'Yes'
b. Next to **Tab Name** type 'Select Academic Year'
c. And while you're here, next to **Color**, select your background color from the drop down.

62. Back on the **Forms** tab, select 'Chart'. Switch back to **Properties,** and repeat steps 64a-c., above, naming the tab 'Charts'.
63. Back on the **Forms** tab again, select 'OLAP'. Switch back to **Properties,** and repeat steps 64a-c., naming the tab 'OLAP'.
64. Head back to **Form** one last time and select 'ChartA'.
65. Since this form will not be a tab, simply switch back to **Properties,** and next to **Color**, select your background color from the drop down.

Add template objects to each form

66. On the **Forms** tab, click on 'Main' to select it.
67. Click on the **Add Objects from Library** button to open the **Choose an Object** window:
    a. Your user folder should be automatically selected. (If it's not, click on it to select it.)
    b. Click on 'Training DataBlock Template' to select it.
    c. Click the **OK** button.
68. Repeat steps 69 & 70 for 'Chart', 'ChartA', and 'OLAP'.
69. Commit and test your changes.
    NOTE: You should see an empty dashboard with 3 tabs.

Add instructions to the 'Main' form

70. On the **Forms** tab, click on 'Main' to select it.
71. Double-click on the 'Description/Instructions' label object and change the text to:
    'Select Academic Year'
72. Add a new label to the panel, double-click on it, and change the text to:
    'Choose an academic year and term type. Analysis will include the last five years.'
73. On the **Properties** tab:
    a. Next to **Font.Size**, type '11'
    b. Next to **Word Wrap**, select 'Yes'
74. Click and drag the new subhead to move it under the 'Select Academic Year' heading.
75. Resize to format the text as you see fit.

Add the 'Academic Year' variable to the 'Main' form

76. Add a label for the variable
    a. Add a new label to the panel.
    b. Double-click and change text to 'Academic Year:
    c. Click and drag label to move it beneath the instructions subhead.
77. Add drop down control object
    a. Click the **Add Drop Down** tool.
    b. Click on the panel to place the drop down.
    c. On the **Properties** tab:
        i. Next to **Variable Name**, type 'main_DD_AcadYr'

ii. Next to **Auto Select,** choose 'Yes'

d. Click and drag the drop down next to the 'Select Academic Year:' label.

78. Write a SQL query to supply the list of academic years.

a. Double click on the 'Academic Year' drop down.

i. Select 'SQL Statement' then click **Next.**

ii. Open the Visual Designer.

(1) Add Table: STVTERM

(2) Add 'STVTERM_ACYR_CODE' to the **Visible Fields (SELECT)** tab:

(a) Click in the **As** row and type 'AcadYr'.

(b) Click on the **Summing** button on the left-hand side of the tab.

(c) In the **Summing** row, select '<Group By>'.

(3) On the **Conditional Fields (WHERE)** tab:

(a) Add 'STVTERM_CODE'

(b) In the **Condition** row, type 'NOT IN ('000000', '999999')'

(c) Add 'STVTERM_START_DATE'

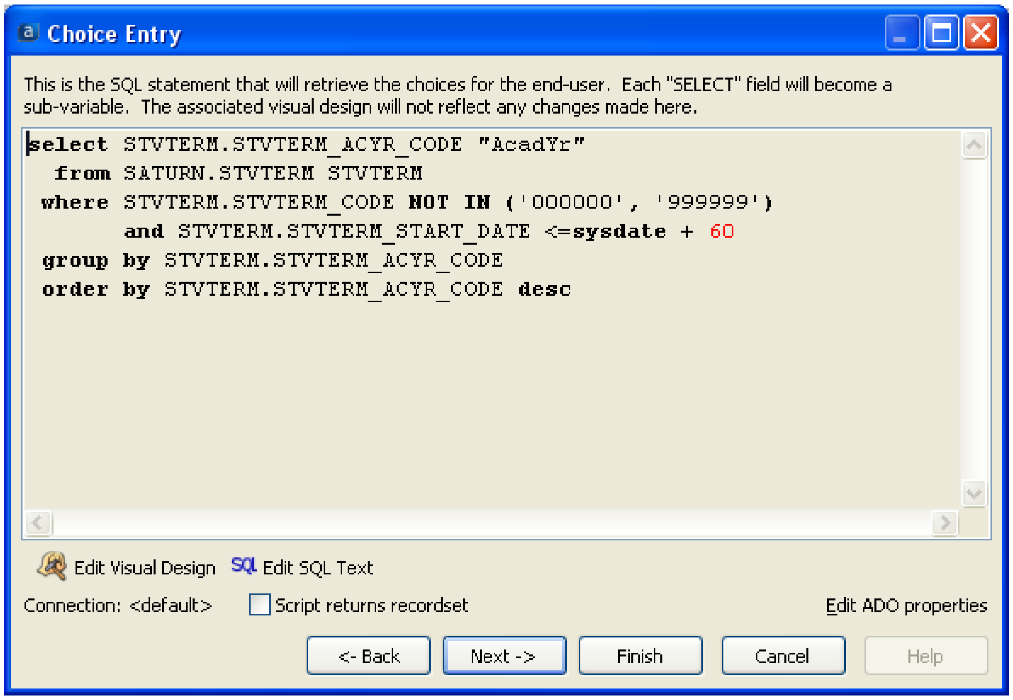
(d) In the **Condition** row, type '<=sysdate + 60'

(4) Add 'STVTERM_ACYR_CODE' to the **Ordering (ORDER BY)** tab:

(a) In the **Sort** row, select 'Descending'.

(5) Click the **OK** button to exit the Visual Designer.

iii. Review your SQL query. It should look like this:



This is the SQL statement that will retrieve the choices for the end-user. Each "SELECT" field will become a sub-variable. The associated visual design will not reflect any changes made here.

```
select STVTERM.STVTERM_ACYR_CODE "AcadYr"
  from SATURN.STVTERM STVTERM
 where STVTERM.STVTERM_CODE NOT IN ('000000', '999999')
       and STVTERM.STVTERM_START_DATE <=sysdate + 60
 group by STVTERM.STVTERM_ACYR_CODE
 order by STVTERM.STVTERM_ACYR_CODE desc
```

iv. Click the **Next** button.

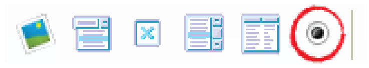
v. You should see a list of academic years. Click the **Next** button.

vi. On the next screen, make sure 'AcadYr' is selected, and then hit the **Finish** button.

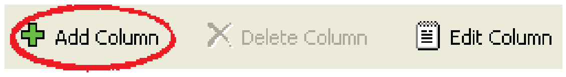
79. Commit and test your changes.

Add the 'Term Type' radio buttons to the 'Main' form

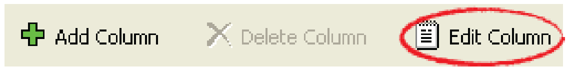
80. Add the radio button control object
    a. Click the **Add Radio Button Panel** button (the image of a radio button on the toolbar).



    b. Click on the panel to place the radio buttons.
    c. On the **Properties** tab:
        i. Next to **Variable Name**, type 'main_RB_TermType'
    d. Click and drag the radio button panel to the right of the 'Academic Year' drop down.
81. Create a list manually to display the term types.
    NOTE: Depending on the available term types at your institution the instructor may decide to create the list of term types manually or may opt to generate the list using a SQL query. The instructions for how to create a list manually follow.
    a. Double click on the 'TermType' radio button panel.
        i. Leave 'Manual Entries' selected and click the **Next** button.
        ii. You'll want two columns for your manual list—one for the term type code and one for the term type description. Click the **Add Column** button on the toolbar to add a new column.



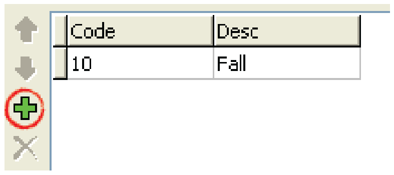
        iii. In the **Column Properties** window that pops up:
            (1) Click in the **Name** field, and type 'Desc'.
            (2) Click the **OK** button.
        iv. Click in the first row of the 'Main' column.
        v. Click the **Edit Column** button on the toolbar



        to open the **Column Properties** window:
            (1) Click in the **Name** field, and type 'Code'.
            (2) Click the **OK** button.

    NOTE: Your list now has two columns, one named 'Code' and one named 'Desc'. The 'Code' column will be a list of term type codes, i.e., 10, 20, 30. The 'Desc' column will have the corresponding term type descriptions, i.e. Fall, Spring, Summer.

        vi. In the first row of the 'Code' column, type the code for your school's first term type (e.g.,'10').
        vii. Click in the first row of the 'Desc' column and type the description for that term type (e.g., 'Fall').
        viii. Click the **Add Row** button (the green plus on the left side of the window)



Code	Desc
10	Fall

        to add a new row to your list.
        ix. Repeat steps vi.-viii. until you have added codes and descriptions for all of your institution's term types.

NOTE: You can rearrange the order of the term types by selecting a row, and clicking the blue up and down arrows to the left.

x. When your list is complete, click the **Next** button.

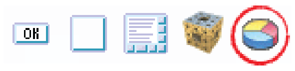
xi. Select 'Desc' to display the descriptions (not the term codes) to the users.

xii. Click the **Finish** button.

82. Commit and test your changes.

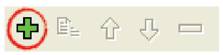
Add instructions, a run button, and the 'Registered Student' chart to the 'Chart' form

83. On the **Forms** tab, click on 'Chart' to select it.
84. Double-click on the 'Description/Instructions' label object and change the text to:
    'Registered Students'
85. Add a new label to the panel, double-click on it, and change the text to:
    'Click on a year to see a breakdown of demographic information.'
86. On the **Properties** tab:
    a. Next to **Font.Size**, type '11'
    b. Next to **Word Wrap**, select 'Yes'
87. Click and drag the new subhead to move it under the 'Registered Students' heading.
88. Click and drag to resize, as you see fit.
89. Click **Add Button** (the OK button on the toolbar) and click to add it to the panel.
90. On the **Properties** tab:
    a. Next to **Text** type 'Run Registered Student Chart'
    b. Next to **Variable Name**, type 'chart_BT_RunChart'
91. Resize the button until the full text is visible.
92. Click and drag the button to the upper part of the panel.
93. Click on the **Add Chart** button (the pie chart icon on the toolbar)



to add a new chart.

94. Click on the panel to place the chart.
95. Resize the chart so that it almost fills the panel.
96. On the **Properties** tab:
    a. Next to **Variable Name,** type 'chart_CT_RegStdnts'
97. Double click on the 'Registered Students' chart, to open the **Chart Wizard** window:
    a. On the welcome screen, click the **Next** button.
    b. Click the **Add Series** button (the green plus sign on the **Series Manager** toolbar)



to add a new series.

NOTE: A series is a set of data points that are displayed in a chart. Charts can have multiple series, but in this example we are displaying only one set of data.

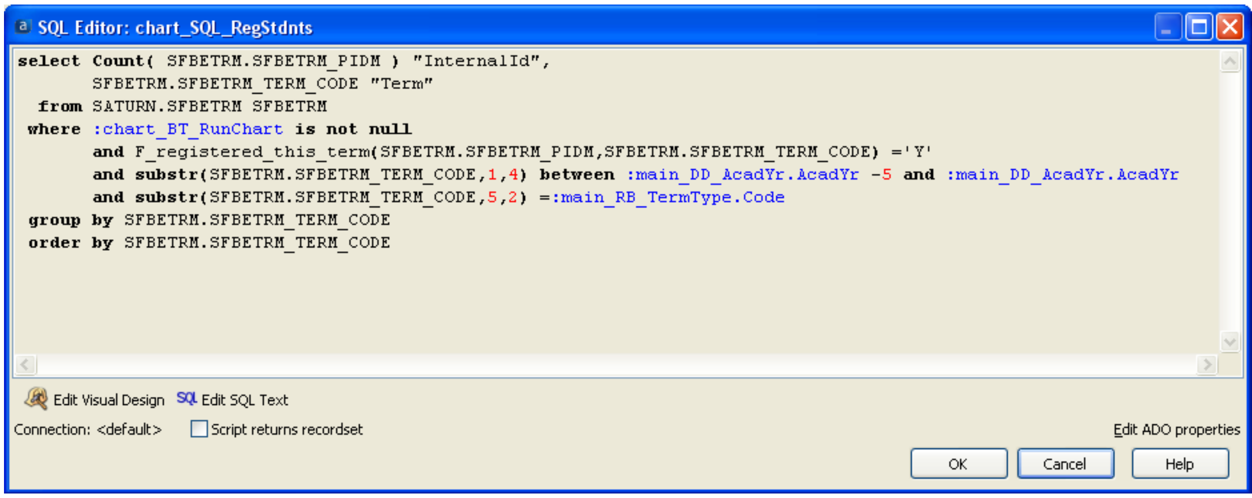
    c. On the **Data** tab:
        i. Next to **Series Name,** type 'RegStudents'.
    d. To the right of the **Dataset** drop down, click the hard hat icon.

i. In the **Dataset Name** window that pops up, type 'chart_SQL_RegStdnts' and click the **OK** button.
ii. Select 'SQL Statement' and click the **OK** button.
iii. Open the Visual Designer:
   (1) Add table: 'SFBETRM'
   (2) On the **Visible Fields (SELECT)** tab:
      (a) Add fields: 'SFBETRM_PIDM' and 'SFBETRM_TERM_CODE'
      (b) Click the **Summing** button.
      (c) In the **Summing** row for 'SFBETRM_PIDM' select 'Count'.
      (d) In the **Summing** row for 'SFBETRM_TERM_CODE' select '<Group By>'.
   (3) On the **Conditional Fields (WHERE)** tab:
      (a) Click in the **Table** row of the first column and select '<calculated>'.
      (b) Double-click in the **Field** row to open the **SQL Editor**:
         (i) Click the **User-Defined Variable** button (the ABC button):
            1) Click on 'chart_BT_RunChart' to select it.
            2) Click the **OK** button.
         (ii) Click the **OK** button.
      (c) Click in the **Condition** row and type 'is not null'.
   (4) In the next column:
      (a) Click in the **Table** row and select '<calculated>'.
      (b) Double-click in the **Field** row to open the **SQL Editor**:
         (i) Type 'f_registered_this_term('
         (ii) Click the **Insert a field** button (the green neutron)
         (iii) Click the plus sign to expand SFBETRM
         (iv) Select SFBETRM_PIDM, click OK
         (v) Type ','
         (vi) Click the **Insert a field** button (the green neutron)
         (vii) Click the plus sign to expand SFBETRM
         (viii) Select SFBETRM_TERM_CODE, click OK
         (ix) Type ')'
         (x) Click OK to exit the SQL Editor
      (c) Click in the **Condition** row and type '='Y' ' *(include single quotes around Y)*
      (d) Repeat steps (a-c) to create the following conditions

Table	**Field**	**Condition**
<calculated>	substr(SFBETRM.SFBETRM_TERM_CODE,1,4)	between :main_DD_AcadYr.AcadYr -5 and :main_DD_AcadYr.AcadYr
<calculated>	substr(SFBETRM.SFBETRM_TERM_CODE,5,2)	=:main_RB_TermType.Code

   (5) On the **Ordering (ORDER BY)** tab:
      (a) Add field: 'SFBETRM_TERM_CODE'
      (b) In the **Sort** row, make sure 'Ascending' is selected.
   (6) Click the **OK** button.

iv. Review your SQL query. It should look like this:



```
select Count( SFBETRM.SFBETRM_PIDM ) "InternalId",
       SFBETRM.SFBETRM_TERM_CODE "Term"
  from SATURN.SFBETRM SFBETRM
 where :chart_BT_RunChart is not null
       and F_registered_this_term(SFBETRM.SFBETRM_PIDM,SFBETRM.SFBETRM_TERM_CODE) ='Y'
       and substr(SFBETRM.SFBETRM_TERM_CODE,1,4) between :main_DD_AcadYr.AcadYr -5 and :main_DD_AcadYr.AcadYr
       and substr(SFBETRM.SFBETRM_TERM_CODE,5,2) =:main_RB_TermType.Code
 group by SFBETRM.SFBETRM_TERM_CODE
 order by SFBETRM.SFBETRM_TERM_CODE
```

Connection: <default> ☐ Script returns recordset

v. Click the **OK** button to exit the Visual Designer and then click the **Yes** button to validate your new query.

vi. Click the **OK** button, leaving the Test Values blank.

vii. You should see a small window that says:
Results:
InternalID = (null)
Term = (null)

viii. Click the **OK** button.

e. Back on the **Data** tab:

i. From the **Value** drop down, select 'InternalID'.

ii. From **Label** drop down, select 'Term'

f. Click the **Next** button.

g. The **Chart Theme and Panel** screen has a wide variety of formatting options to choose from. You can select a predefined Theme from the **Apply a Theme** list box, or a color palette from the **Select a Palette** list box. Below, you can modify the background on the **Format Background Panel** tab. Try out a few options, until you find one you like, and click the **Next** button.

h. On the **Legend Style and Position** tab:

i. Click the 'Legend Visible' checkbox to uncheck it.

i. Click the **Finish** button.

98. Commit and test your changes.

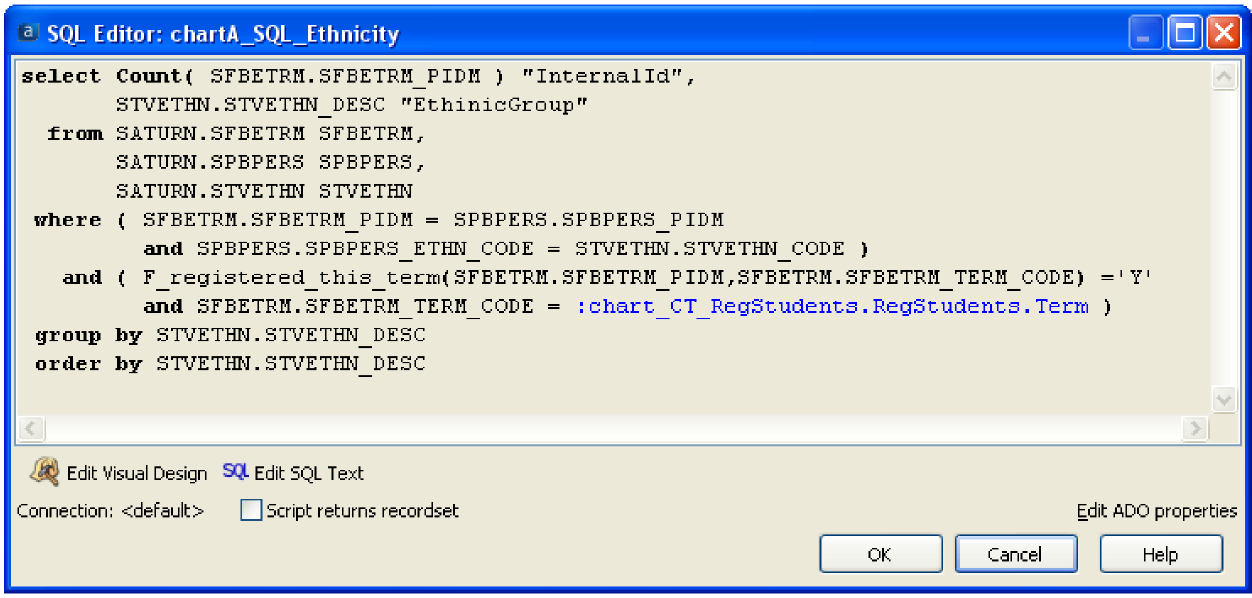
Add the 'Ethnicity' chart to the 'ChartA' form

From the 'Registered Students' chart we just created, users will be able to click on a term to see detailed demographic data for registered students in that term. The 'ChartA' form we created will house these detail charts, sometimes referred to as drill downs. Our next step will be to create the 'Ethnicity' drill down chart, which will display the ethnicity breakdown of all registered students for the selected term.

When creating drill down charts like this, you must recreate all of the same constraints you used in the main chart so that the drill downs are displaying information from the same data set. In this case, because the 'Registered Students' chart counts only registered students, the 'Ethnicity' chart must have the same constraints. The easiest way to ensure the constraints match exactly is to copy the SQL from the 'Registered Students' chart and simply modify it for the 'Ethnicity' chart (and any other drill downs).

99. Copy the SQL from the 'Registered Students' chart.
    a. On the **Variables** tab:
        i. Click 'Chart_SQL_RegStdnts' to select the SQL for the 'Registered Students' chart.
        ii. Right-click and select **Copy** from the menu.
        iii. Right-click anywhere on the **Variables** tab and select **Paste** from the menu.
        iv. Right-click on the new variable, called 'SqlVariable1', and rename it to 'ChartA_SQL_Ethnicity'.
100. Modify the 'Ethnicity' SQL to display ethnicity for registered students in a selected term.
    a. Double-click on 'ChartA_SQL_Ethnicity' and open the Visual Designer:
        i. Add tables: 'SPBPERS' and 'STVETHN'
        ii. Create table joins:
            (1) In the 'SFBETRM' table window, click on 'SFBETRM_PIDM' (row 2), drag it to the 'SPBPERS' table window, and drop it on 'SPBPERS_PIDM' (row 1).
            (2) In the 'SPBPERS' table window, click on 'SPBPERS_ETHN_CODE' (row 5), drag it to the 'STVETHN' table window, and drop it on 'STVETHN_CODE' (row 1).
        iii. On the **Visible Fields (SELECT)** tab:
            (1) Delete 'SFBETRM_TERM_CODE' by clicking the **X** button at the top of the column.
            (2) Add field: 'STVETHN_DESC'.
            (3) Make sure the **Summing** row still shows '<Group By>'.
        iv. On the **Conditional Fields (WHERE)** tab:
            (1) Delete the first, third, and fourth conditions by clicking the **X** button at the top of their columns.
            NOTE: This means that only the follow condition should remain:
            'F_registered_this_term(SFBETRM.SFBETRM_PIDM,SFBETRM.SFBETRM_TERM_CODE) = 'Y'
            (2) Add 'SFBETRM_TERM_CODE' from the 'SFBETRM' table window.
            (3) Double-click in the **Condition** row to open the **SQL Editor**:
                (a) Click the **Add User-Defined Variable** button (the ABC button):
                    (i) Expand out 'Chart_CT_RegStudents'.
                    (ii) Beneath that, expand out the 'RegStudents' series.
                    (iii) Click on 'Term' to select it.
                    (iv) Click the **OK** button.
                (b) Click the **OK** button.
        v. On the **Ordering (ORDER BY)** tab:
            (1) Delete 'SFBETRM_TERM_CODE' by clicking the **X** button at the top of the column.
            (2) Add 'STVETHN_DESC' from the 'STVETHN' table window.
        vi. Click the **OK** button.

vii. Review your SQL query. It should look like this:



```
select Count( SFBETRM.SFBETRM_PIDM ) "InternalId",
       STVETHN.STVETHN_DESC "EthinicGroup"
  from SATURN.SFBETRM SFBETRM,
       SATURN.SPBPERS SPBPERS,
       SATURN.STVETHN STVETHN
 where ( SFBETRM.SFBETRM_PIDM = SPBPERS.SPBPERS_PIDM
         and SPBPERS.SPBPERS_ETHN_CODE = STVETHN.STVETHN_CODE )
   and ( F_registered_this_term(SFBETRM.SFBETRM_PIDM,SFBETRM.SFBETRM_TERM_CODE) ='Y'
         and SFBETRM.SFBETRM_TERM_CODE = :chart_CT_RegStudents.RegStudents.Term )
 group by STVETHN.STVETHN_DESC
 order by STVETHN.STVETHN_DESC
```

viii. Click the **OK** button to and then click the **Yes** button to validate your new query.
ix. Click the **OK** button, leaving the test values blank.
x. You should see a small window that says:
Results:
InternalID = (null)
Term = (null)
i. Click the **OK** button.

101.Create the 'Ethnicity' chart

a. On the **Forms** tab:
   i. Click 'ChartA' to select it.
b. Click on the **Add Chart** button (the pie chart icon on the toolbar) to add a new chart.
c. Click on the panel to place the chart.
d. Click and drag it to the bottom of the panel.
e. Resize the chart so it takes up the full width and most of the lower half of the form.
   NOTE: You'll need to leave room for the other two charts you'll be adding to this form. See the example images at the beginning of this exercise for reference.
f. On the **Properties** tab:
   i. Next to **Variable Name,** type 'ChartA_CT_Ethnicity'.
g. Double click on the 'Ethnicity' chart, to open the **Chart Wizard** window:
   i. On the welcome screen, click the **Next** button.
   ii. Click the **Add Series** button (the green plus sign on the **Series Manager** toolbar) to add a new series.
   iii. On the **Data** tab:
      (1) Next to **Series Name,** type 'Ethnicity'.
      (2) From the **Dataset** drop down, select the variable we just created: 'ChartA_SQL_Ethnicity'.
      (3) From the **Value** drop down, select 'InternalID'.

(4) From the **Label** drop down, select 'EthnicGroup'.

iv. Click the **Next** button.

v. On the **Chart Theme and Panel** screen, select a theme and color palette for your chart.

vi. Click the **Next** button, which will take you to the **Chart Legends and Titles** screen.

vii. Click on 'Main Title' in the list to select it.

viii. On the **Title** tab:

(1) Type 'Ethnicity' in the **Enter Title Here** text box.

(2) From the **Alignment** drop down, select 'Center'.

(3) Click the **Font** button or the **Format** tab to adjust the title formatting as desired.

ix. Click the **Advanced** button in the lower left corner

(1) Click **Yes** to continue

(2) In the editing box, click on the arrow next to Axis to expand the selection

(3) Select Bottom Axis

(4) Uncheck the Visible box

x. Click the **Close** button.

h. Click Commit, but don't bother testing your changes yet. We have created a new chart but we haven't yet added a way to navigate to that chart.

Create navigation from the 'Chart' form to the 'ChartA' form

102.On the **Form** tab:

a. Click on 'Chart' to select it.

103.On the panel, click on the 'Registered Students' chart to select it.

104.On the **Properties** tab:

a. Next to **Cursor**, select 'HandPoint'.
NOTE: This will make the cursor turn into a pointing hand when it's over the 'Registered Students' chart, which will indicate to the end user that clicking on the chart does something.

b. Next to **On Click**, click the ellipsis button to open the **Edit Event** window:

i. From the **Available Events** list, select 'Activate Form'.

ii. Click the blue arrow button to move 'Activate Form' to the **Event List.**

iii. In the **Event List**, click 'Activate Form' to select it.

iv. From the **Event Properties** drop down, select 'ChartA'.

v. Click the **OK** button.

105.Commit and test your changes.
NOTE: To test that your On Click navigation works, you'll need to select an academic year from the drop down, go to the 'Chart' tab, run the 'Registered Student' chart and then click the bar of the term for which you want to see details. It should take you to the 'ChartA' form. Click the 'Charts' tab to get back to the 'Registered Student' chart.

Add the 'Gender' chart to the 'ChartA' form

Much like we did with the 'Ethnicity' chart, we're going to copy the SQL variable from 'Ethnicity' and use it as our base for the 'Gender' chart, which will also live on the 'ChartA' form.

106.Copy the SQL from the 'Ethnicity' chart.

a. On the **Variables** tab:

i. Click ‘ChartA_SQL_Ethnicity’ to select it.
ii. Right-click and select **Copy** from the menu.
iii. Right-click anywhere on the **Variables** tab and select **Paste** from the menu.
iv. Right-click on the new variable, called ‘SqlVariable1’, and rename it to ‘ChartA_SQL_Gender’.

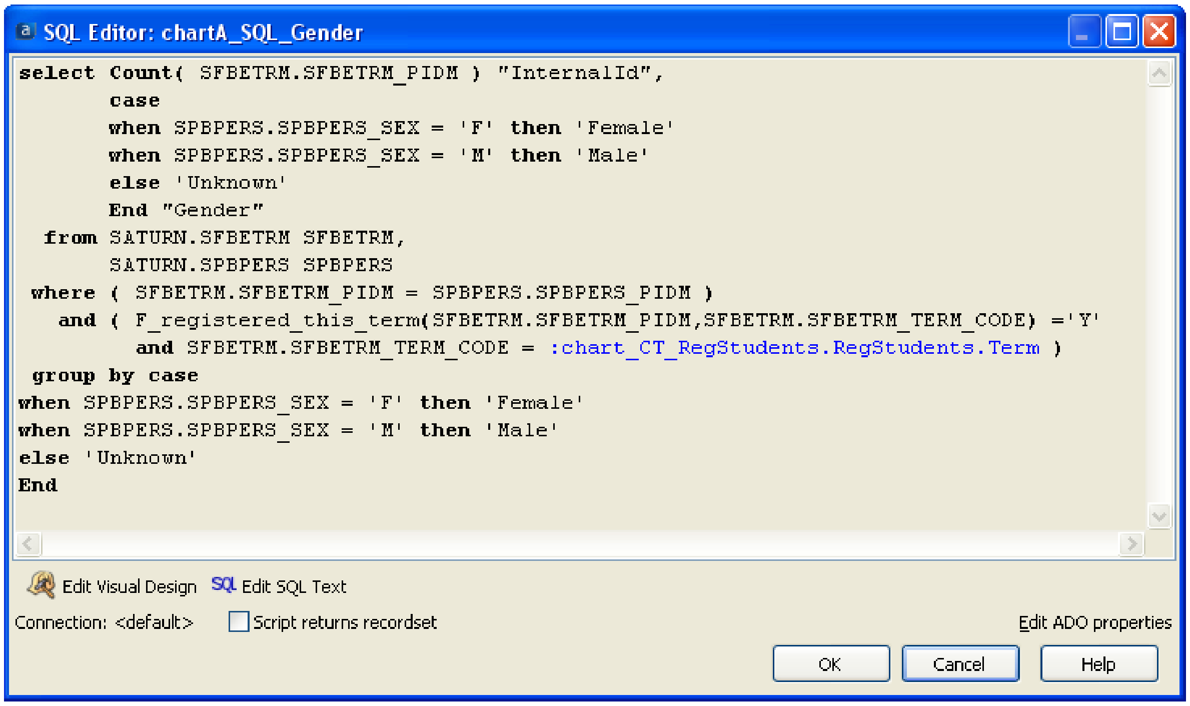
107. Modify the ‘Gender’ SQL to display gender for registered students in a selected term.

a. Double-click on ‘ChartA_SQL_Gender’ and open the Visual Designer:
   i. Remove the ‘STVETHN’ table.
   ii. On the **Visible Fields (SELECT)** tab:
      (1) Click in the **Table** row of the second column and select ‘<calculated>’.
      (2) Double-click in the **Field** row to open the **SQL Editor:**
         (a) Type the following to create a case statement to display the gender values:

```
case
    when SPBPERS.SPBPERS_SEX = 'F' then 'Female'
    when SPBPERS.SPBPERS_SEX = 'M' then 'Male'
    else 'Unknown'
end
```

         *NOTE: You can use the **Insert Field** button (green neutron on the toolbar) to find the ‘SPBPERS.SPBPERS_SEX’ field.*
         (b) Click the **OK** button.
      (3) In the **As** row, type ‘Gender’.
      (4) Confirm that the **Summing** row is set to ‘<Group By>’.
   iii. We don’t need to make any changes to the **Conditional Fields (WHERE)** or **Ordering (ORDER BY)** tabs, so click the **OK** button to close the Visual Designer.

b. Review your SQL query. It should look like this:



```
select Count( SFBETRM.SFBETRM_PIDM ) "InternalId",
       case
       when SPBPERS.SPBPERS_SEX = 'F' then 'Female'
       when SPBPERS.SPBPERS_SEX = 'M' then 'Male'
       else 'Unknown'
       End "Gender"
  from SATURN.SFBETRM SFBETRM,
       SATURN.SPBPERS SPBPERS
 where ( SFBETRM.SFBETRM_PIDM = SPBPERS.SPBPERS_PIDM )
   and ( F_registered_this_term(SFBETRM.SFBETRM_PIDM,SFBETRM.SFBETRM_TERM_CODE) ='Y'
         and SFBETRM.SFBETRM_TERM_CODE = :chart_CT_RegStudents.RegStudents.Term )
 group by case
when SPBPERS.SPBPERS_SEX = 'F' then 'Female'
when SPBPERS.SPBPERS_SEX = 'M' then 'Male'
else 'Unknown'
End
```

Connection: <default>

c. Click the **OK** button to and then click the **Yes** button to validate your new query.
d. Click the **OK** button, leaving the test values blank.
e. You should see a small window that says:
   Results:
   InternalID = (null)
   Gender = (null)
f. Click the **OK** button.

108.Create the 'Gender' chart.
a. On the **Forms** tab:
   i. Click 'ChartA' to select it.
b. Click on the **Add Chart** button (the pie chart icon on the toolbar) to add a new chart.
c. Click on the panel to place the chart.
d. Click and drag it to the upper left corner of the panel.
e. Resize the chart so it takes up roughly a quarter of the panel.
   NOTE: Remember to leave room for one more chart.
f. On the **Properties** tab:
   i. Next to **Variable Name**, type 'ChartA_CT_Gender'
g. Double click on the 'Gender' chart, to open the **Chart Wizard** window:
   i. On the welcome screen, click the **Next** button.
   ii. Click the **Add Series** button (the green plus sign on the **Series Manager** toolbar) to add a new series.
   iii. On the **Data** tab:
      (1) Next to **Series Name,** type 'Gender'.
      (2) From the **Dataset** drop down, select the 'ChartA_SQL_Gender' variable.
      (3) From the **Value** drop down, select 'Internal ID'.
      (4) From the **Label** drop down, select 'Gender'.
   iv. On the **Type** tab:
      (1) Under **Select Chart Type**, select 'Pie Chart'.
   v. On the **Labels** tab:
      (1) From the **Label Style** drop down, select 'Value'.
   vi. Click the **Next** button.
   vii. On the **Chart Theme and Panel** screen, select a theme and color palette for your chart.
   viii. Click the **Next** button, which will take you to the **Chart Legends and Titles** screen.
   ix. On the **Legend Style and Title** tab:
      (1) Uncheck the **Legend Visible** checkbox
   x. Click on 'Main Title' in the list to select it.
   xi. On the **Title** tab:
      (1) Type 'Gender' in the **Enter Title Here** text box.
      (2) From the **Alignment** drop down, select 'Center'.
      (3) Click the **Font** button or the **Format** tab to adjust the title formatting as desired.
   xii. Click the **Finish** button.

109.Commit and test your changes.

Add the 'Age Range' chart to the 'ChartA' form

The 'Age Range' chart will be the last chart on the 'ChartA' form. We'll put it together in much the same way, starting with the 'Gender' chart SQL variable.

110.Copy the SQL from the 'Gender' chart.

- a. On the **Variables** tab:
  - i. Click 'ChartA_SQL_Gender' to select it.
  - ii. Right-click and select **Copy** from the menu.
  - iii. Right-click anywhere on the **Variables** tab and select **Paste** from the menu.
  - iv. Right-click on the new variable, called 'SqlVariable1', and rename it to 'ChartA_SQL_Age'.

111.Modify the 'Age Range' SQL to display the age ranges for registered students in a selected term.

- a. Double-click on 'ChartA_SQL_Age' and open the Visual Designer:
  - i. On the **Visible Fields (SELECT)** tab:
    - (1) Delete the case statement from the second column by clicking the red **X** button at the top.
    - (2) Click in the **Table** row of the second column and select '<calculated>'.
    - (3) Double-click in the **Field** row to open the **SQL Editor:**
      - (a) Type the following to create a case statement to display age groups:

```
case
   when SPBPERS.SPBPERS_BIRTH_DATE is null then 'Unknown'
   when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char
(SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 18 then 'Under 18'
   when  (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char
(SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY')) < 23 then '18 to 22'
   when  (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char
(SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY')) < 26 then '23 to 25'
   else 'Over 25'
End
```

      NOTE: Again, remember to use the ***Insert Field*** *button. Copying and pasting will speed this up too—the 3 'when' lines are very similar and can be easily copied and edited for each age group.*

      - (b) Click the **OK** button.
    - (4) In the **As** row, type 'AgeGroup'.
    - (5) Confirm that the **Summing** row is set to '<Group By>'.
    - (6) Click in the **Table** row of the third column, and select '<calculated>'.
    - (7) Right-click the **Field** row of the *second* column and choose **Copy** from the menu to copy the case statement you just wrote.
    - (8) Right-click in the **Field** row of the third column and choose **Paste** from the menu.
    - (9) Double-click in the **Field** row of the third column to open the **SQL Editor**:
      - (a) Edit the case statement as follows—replacing the age group descriptions with the numbers 1-5—to create an order by which the 'Age Range' chart will sort the data:

case

        when SPBPERS.SPBPERS_BIRTH_DATE is null then **1** when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 18 then **2**

when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY')) < 23 then **3**

when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY')) < 26 then **4**

else **5**

End

(b) Click the **OK** button.

(10) In the **As** row, type 'SortMe'.

(11) Confirm that the **Summing** row is set to '<Group By>'.

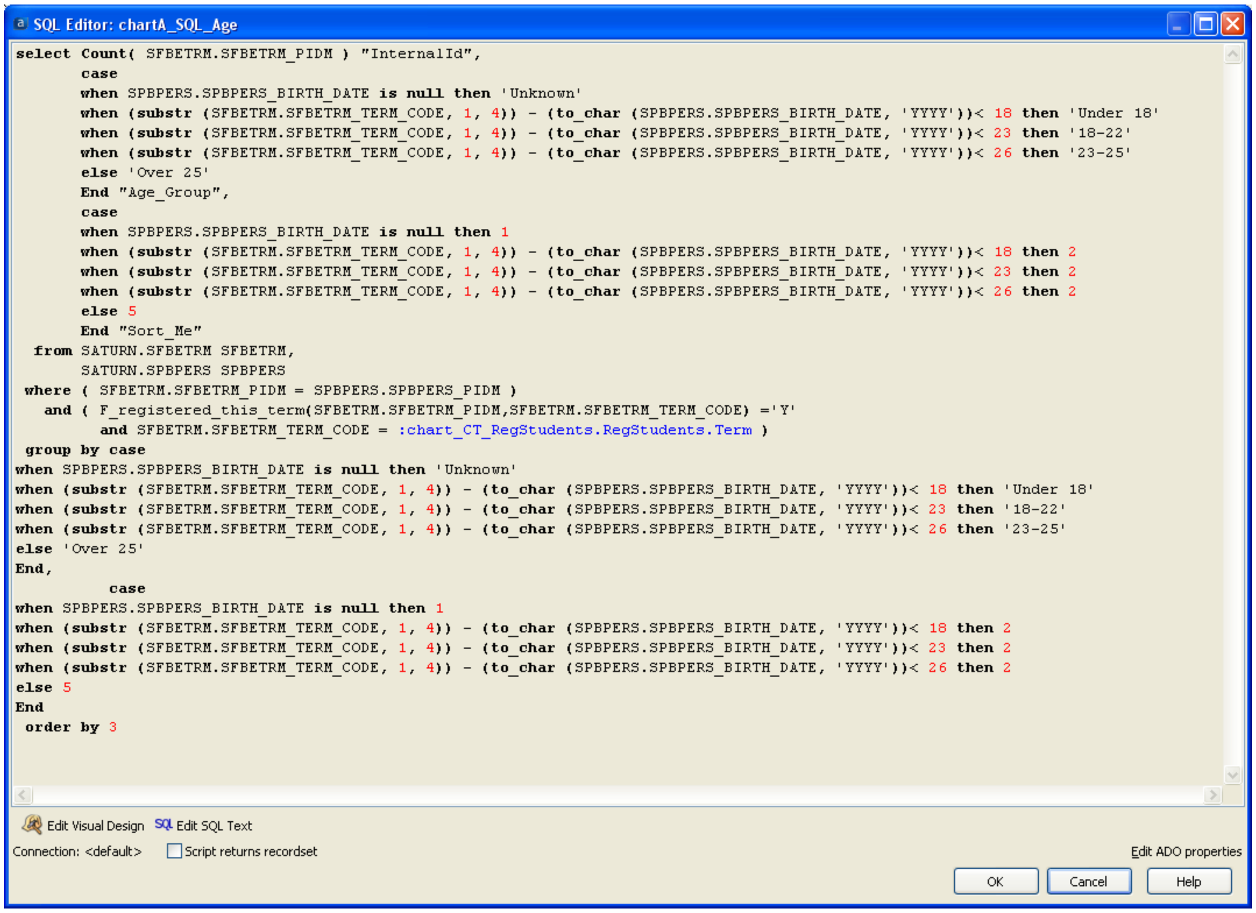
ii. On the **Ordering (ORDER BY)** tab:

(1) In the **Table** row, select '<calculated>'.

(2) In the **Field** row, type '3'.

iii. Click the **OK** button to close the Visual Designer.

b. Review your SQL query. It should look like this:



SQL Editor: chartA_SQL_Age

```
select Count( SFBETRM.SFBETRM_PIDM ) "InternalId",
       case
       when SPBPERS.SPBPERS_BIRTH_DATE is null then 'Unknown'
       when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 18 then 'Under 18'
       when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 23 then '18-22'
       when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 26 then '23-25'
       else 'Over 25'
       End "Age_Group",
       case
       when SPBPERS.SPBPERS_BIRTH_DATE is null then 1
       when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 18 then 2
       when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 23 then 2
       when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 26 then 2
       else 5
       End "Sort_Me"
  from SATURN.SFBETRM SFBETRM,
       SATURN.SPBPERS SPBPERS
 where ( SFBETRM.SFBETRM_PIDM = SPBPERS.SPBPERS_PIDM )
   and ( F_registered_this_term(SFBETRM.SFBETRM_PIDM,SFBETRM.SFBETRM_TERM_CODE) ='Y'
         and SFBETRM.SFBETRM_TERM_CODE = :chart_CT_RegStudents.RegStudents.Term )
 group by case
when SPBPERS.SPBPERS_BIRTH_DATE is null then 'Unknown'
when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 18 then 'Under 18'
when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 23 then '18-22'
when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 26 then '23-25'
else 'Over 25'
End,
          case
when SPBPERS.SPBPERS_BIRTH_DATE is null then 1
when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 18 then 2
when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 23 then 2
when (substr (SFBETRM.SFBETRM_TERM_CODE, 1, 4)) - (to_char (SPBPERS.SPBPERS_BIRTH_DATE, 'YYYY'))< 26 then 2
else 5
End
 order by 3
```

c. Click the **OK** button to and then click the **Yes** button to validate your new query.

d. Click the **OK** button, leaving the test values blank.

e. You should see a small window that says:
   Results:
   InternalID = (null)
   AgeGroup = (null)
   SortMe = (null)
f. Click the **OK** button.

112. Create the 'Age Range' chart.
   a. On the **Forms** tab:
      i. Click 'ChartA' to select it.
   b. Click on the **Add Chart** button (the pie chart icon on the toolbar) to add a new chart.
   c. Click on the panel to place the chart.
   d. Click and drag it to the upper right corner of the panel.
   e. Resize the chart so it takes up the remaining quarter of the panel.
   f. On the **Properties** tab:
      i. Next to **Variable Name**, type 'ChartA_CT_Age'
   g. Double click on the 'Age Range' chart, to open the **Chart Wizard** window:
      i. On the welcome screen, click the **Next** button.
      ii. Click the **Add Series** button (the green plus sign on the **Series Manager** toolbar) to add a new series.
      iii. On the **Data** tab:
         (1) Next to **Series Name,** type 'Age'.
         (2) From the **Dataset** drop down, select the 'ChartA_SQL_Age' variable.
         (3) From the **Value** drop down, select 'Internal ID'.
         (4) From the **Label** drop down, select 'AgeGroup'.
      iv. On the **Type** tab:
         (1) Under **Select Chart Type**, select 'Pie Chart'.
      v. On the **Labels** tab:
         (1) From the **Label Style** drop down, select 'Percent'.
      vi. Click the **Next** button.
      vii. On the **Chart Theme and Panel** screen, select a theme and color palette for your chart.
      viii. Click the **Next** button, which will take you to the **Chart Legends and Titles** screen.
      ix. On the **Legend Style and Title** tab:
         (1) Uncheck the **Legend Visible** checkbox
      x. Click on 'Main Title' in the list to select it.
      xi. On the **Title** tab:
         (1) Type 'Age Range' in the **Enter Title Here** text box.
         (2) From the **Alignment** drop down, select 'Center'.
         (3) Click the **Font** button or the **Format** tab to adjust the title formatting as desired.
      xii. Click the **Finish** button.

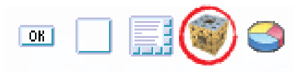
113. Commit and test your changes.

Add a run button and an OLAP cube to the 'OLAP' form

114. Create run button for the OLAP cube.
   a. On the **Forms** tab, click on 'OLAP' to select it.

   b. Click **Add Button** (the OK button on the toolbar) and click to add it to the panel.
   c. On the **Properties** tab:
      i. Next to **Text** type 'Run OLAP Cube'
      ii. Next to **Variable Name**, type 'OLAP_BT_RunOLAP'
   d. Resize the button until the full text is visible.
   e. Click and drag the button to the upper right corner of the panel.

115. Copy the SQL from the 'Registered Students' chart.
   NOTE: Unlike for the other charts in this exercise, we can't just duplicate the SQL variable and modify it. Instead, we're going to go into the Visual Designer, copy the SQL directly, and paste it into an OLAP object.
   a. On the **Variables** tab:
      i. Double-click 'Chart_SQL_RegStdnts', and open the Visual Designer:
         (1) Click the **Copy** button on the toolbar.
         (2) Click the **Close** button to exit the Visual Designer.
      ii. Click the **Close** button again.

116. Click the **Add OLAP Cube** button (the cube icon on the toolbar)


   to add a new OLAP object.

117. Click on the panel to place the cube.
   NOTE: For most DataBlock elements, we would assign a variable name at this point. However, since OLAP cubes cannot be referenced as variables by other SQL queries, we don't need to assign them a variable name.

118. Resize the OLAP cube so that it fills the entire panel.

119. Double-click on the cube to open the **Build OLAP Data Cube** window:
   a. On the **Main Properties** screen, leave everything in its default setting, and hit the **Next** button.
   b. On the **Build Statements** screen, click the hard hat icon at the bottom to open the Visual Designer:
      i. Click the **Paste** button in the toolbar to paste in the SQL from the 'Registered Students' chart.
      ii. Add table: 'SGBSTDN'
      iii. Create a table join:
         (1) In the 'SFBETRM' table window, click on 'SFBETRM_PIDM' (row 2), drag it to the 'SGBSTDN' table window, and drop it on 'SGBSTDN_PIDM' (row 1).
         (2) Right-click on the join and select **Edit Join** from the menu.
         (3) In the **Edit Join** window, select 'Outer Left Join'.
         NOTE: Make sure that 'SFBETRM' is the left table and 'SGBSTDN' is the right table.
         (4) Click the **OK** button.
      iv. On the **Visible Fields (SELECT)** tab:
         (1) Delete both fields by clicking the **X** button at the top of the columns.
         (2) Click the **Summing** button to turn summing off.
         (3) Add the following fields:
            SFBETRM.SFBETRM_PIDM
            SFBETRM.SFBETRM_TERM_CODE
            SFBETRM.SFBETRM_ESTS_CODE
            SFBETRM.SFBETRM_RGRE_CODE
            SGBSTDN.SGBSTDN_LEVL_CODE
            SGBSTDN.SGBSTDN_STYP_CODE

SGBSTDN.SGBSTDN_COLL_CODE_1
SGBSTDN.SGBSTDN_MAJR_CODE_1

NOTE: With the exception of SFBETRM_PIDM, these will be the dimensions for your OLAP cube.

(4) Click in the **Table** row of the next column and select '<calculated>'.

(5) Double-click in the **Field** row to open the **SQL Editor**:

(a) Type the following to create a scalar sub query that will include the degree descriptions in the OLAP cube:

```
(select stvdegc_desc
from stvdegc
where sgbstdn_degc_code_1 = stvdegc_code
and rownum = 1)
```

(b) Click the **OK** button.

(c) In the **As** row, type 'Degree'.

(d) Click the **Table** row of the next column and select '<calculated>'.

(e) Double-click in the **Field** row to open the **SQL Editor**:

(i) Type the following, a Banner function, to retrieve the college description:

```
f_get_desc_fnc('STVCOLL',SGBSTDN.SGBSTDN_COLL_CODE_1,30)
```

(f) In the **As** row, type 'College'.

v. On the **Conditional Fields (WHERE)** tab:

(1) Find the condition that points to the Chart button.
NOTE: This should be the first condition. The Field row will read ':Chart_BT_RunChart'.

(2) Double-click in the **Field** row to open the **SQL Editor**:

(a) Click the **Add User-Defined Variable** button (the ABC button):

(i) Click on 'OLAP_BT_RunOLAP' to select it

(ii) Click the **OK** button.

(b) Click the **OK** button

vi. On the **Ordering (ORDER BY)** tab:

(1) Delete 'SFBETRM_TERM_CODE' by clicking the **X** button at the top of the column.
NOTE: The OLAP cube will order the data automatically.

vii. The last step is to add a correlated sub query to evaluate for the latest term. Click the **Sub Query** button on the toolbar to open a new visual design window:

(1) Add table: 'SGBSTDN'
NOTE: The table is automatically added under the alias 'SGBSTDN1'.

(2) On the **Visible Fields (SELECT)** tab:

(a) Add field: 'SGBSTDN_TERM_CODE_EFF'

(b) Click the **Summing** button.

(c) In the **Summing** row, select 'Max'.

(3) On the **Conditional Fields (WHERE)** tab:

(a) Add field: 'SGBSTDN_PIDM'

(b) Double-click in the **Condition** row to open the **SQL Editor**:

(i) Click the **Insert Field** button (green neutron on the toolbar):

1) Expand out 'SGBSTDN'.

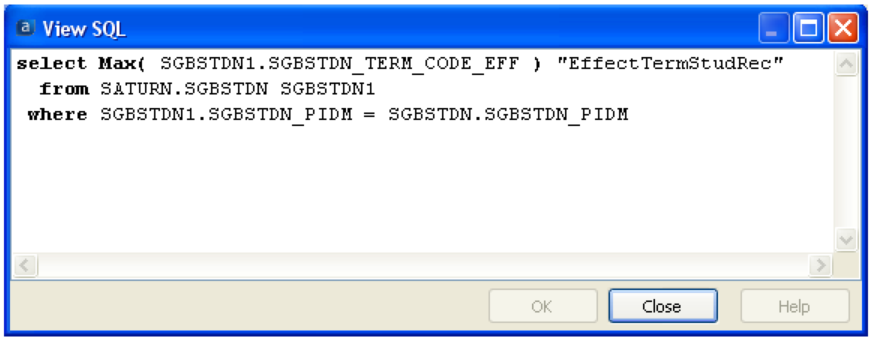
NOTE: Make sure you select 'SGBSTDN' and NOT 'SGBSTDN1'.

2) Click on 'SGBSTDN_PIDM' to select it.

3) Click the **OK** button.

(ii) Click the **OK** button.

(4) Click the **View SQL** button on the toolbar, and review your sub-query. It should look like this:



```
select Max( SGBSTDN1.SGBSTDN_TERM_CODE_EFF ) "EffectTermStudRec"
  from SATURN.SGBSTDN SGBSTDN1
 where SGBSTDN1.SGBSTDN_PIDM = SGBSTDN.SGBSTDN_PIDM
```

(a) Click the **Close** button.

(5) Click the **OK** button to return to your original visual design window.

viii. You'll see that you have a new box—called 'Query'—in your visual design.

*NOTE: Remember, if you want to rename the 'Query' box, click on the **Edit Query Properties** button (the hardhat and hammer) on the 'Query' box toolbar.*

ix. Click the **Add to Conditional Fields Tree** button in the 'Query' box (the window icon with the green plus on the 'Query) to add your correlated subquery to the **Conditional Fields (WHERE)** tab.

x. On the **Conditional Fields (WHERE)** tab:

(1) In the conditional fields tree on the left, you'll see a new condition called '<query>'. Click to select it.

(2) Click the radio button for '=', which will display a new column of information about your new condition.

(3) Double-click in the **Field** row to open the **SQL Editor**:

(a) Click the **Insert Field** button (the green neutron on the toolbar):

(i) Expand out 'SGBSTDN'.

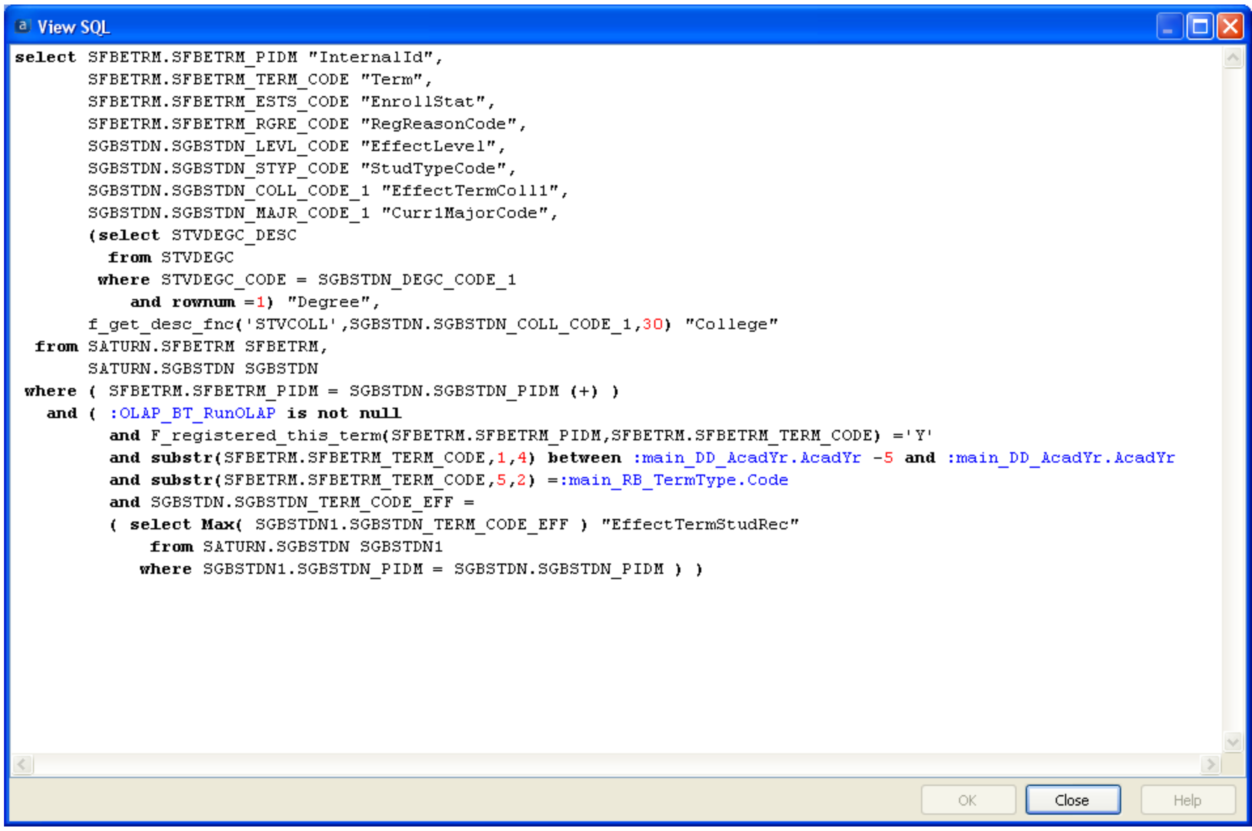
(ii) Click on 'SGBSTDN_TERM_CODE_EFF' to select it.

(iii) Click the **OK** button.

(b) Click the **OK** button.

NOTE: Depending on your Banner configuration of the 'SGBSTDN' table, your instructor may add in additional conditions.

xi. Click the **View SQL** button on the toolbar, and review your sub-query. It should look like this:



```
select SFBETRM.SFBETRM_PIDM "InternalId",
       SFBETRM.SFBETRM_TERM_CODE "Term",
       SFBETRM.SFBETRM_ESTS_CODE "EnrollStat",
       SFBETRM.SFBETRM_RGRE_CODE "RegReasonCode",
       SGBSTDN.SGBSTDN_LEVL_CODE "EffectLevel",
       SGBSTDN.SGBSTDN_STYP_CODE "StudTypeCode",
       SGBSTDN.SGBSTDN_COLL_CODE_1 "EffectTermColl1",
       SGBSTDN.SGBSTDN_MAJR_CODE_1 "Curr1MajorCode",
       (select STVDEGC_DESC
         from STVDEGC
        where STVDEGC_CODE = SGBSTDN_DEGC_CODE_1
           and rownum =1) "Degree",
       f_get_desc_fnc('STVCOLL',SGBSTDN.SGBSTDN_COLL_CODE_1,30) "College"
  from SATURN.SFBETRM SFBETRM,
       SATURN.SGBSTDN SGBSTDN
 where ( SFBETRM.SFBETRM_PIDM = SGBSTDN.SGBSTDN_PIDM (+) )
   and ( :OLAP_BT_RunOLAP is not null
         and F_registered_this_term(SFBETRM.SFBETRM_PIDM,SFBETRM.SFBETRM_TERM_CODE) ='Y'
         and substr(SFBETRM.SFBETRM_TERM_CODE,1,4) between :main_DD_AcadYr.AcadYr -5 and :main_DD_AcadYr.AcadYr
         and substr(SFBETRM.SFBETRM_TERM_CODE,5,2) =:main_RB_TermType.Code
         and SGBSTDN.SGBSTDN_TERM_CODE_EFF =
         ( select Max( SGBSTDN1.SGBSTDN_TERM_CODE_EFF ) "EffectTermStudRec"
             from SATURN.SGBSTDN SGBSTDN1
            where SGBSTDN1.SGBSTDN_PIDM = SGBSTDN.SGBSTDN_PIDM ) )
```

(1) Click the **Close** button.

xii. Click the **OK** button to exit the Visual Designer.

c. Click the **Next** button.

d. The **Test Values** window will pop up. Leave it blank and click the **OK** button.

e. On the **Configure Measures** screen, double-click on 'InternalID' in the **Available Fields** list to add it to the **Measures** list and activate it.

f. Under **Display Name**, type 'Student'.

g. Under **Method of Calculation**, select 'Count'.

h. Under **Display Format**, type '0' (with no decimal places).

*NOTE: You can also click the ellipsis button to open the **Edit Format String** window. This will display a list of format options you can choose from.*

i. Click the **Next** button.

j. On the **Configure Dimensions** page, double-click the fields in the **Fields Available** list box to add them to the **Available** tab.

NOTE: Do not add 'InternalID'. Start with the field immediately after it—if you followed the order outlined above, it should be 'Term'—and work your way down.

k. Customize the **Display Name** for each dimension, if you like.

l. Click and drag 'Term' from the **Available** tab to the **Column** tab.

m. Click and drag 'EnrollStat' from the **Available** tab to the **Row** tab.

n. Click the **Finish** button.

120. Commit and test your changes.
NOTE: You should now be able to select a year and term type on the 'Main' form, and use the run buttons on 'Chart' and 'OLAP' to generate charts and an OLAP cube for that data.
121. Congratulations! Your fourth Advanced DataBlock is complete. Click the **Close** button to exit the DataBlock Designer.