

Paper 110-2007

A “SAS® Programmer’s” Guide to SAS® Enterprise Guide®
 Marje Fecht, Prowerk Consulting LLC, Cape Coral, FL
 Rupinder Dhillon, Dhillon Consulting Inc., Toronto, ON, Canada

ABSTRACT

You have been programming in SAS for a while..... You have been told “we are moving to Enterprise Guide” and removing Display Manager”. You say *I can program everything just fine myself, thank – you!* OR What *am I supposed to do with all these windows?*

This presentation will demonstrate how SAS programmers can use SAS Enterprise Guide as their primary interface to the SAS system while maintaining the flexibility of writing their own customized code. We will look at

- how to navigate the views and menus
- how SAS Enterprise Guide can be used as your primary SAS Editor
- how you can leverage the more complex built-in capabilities available in SAS Enterprise Guide to further enhance the information you deliver.

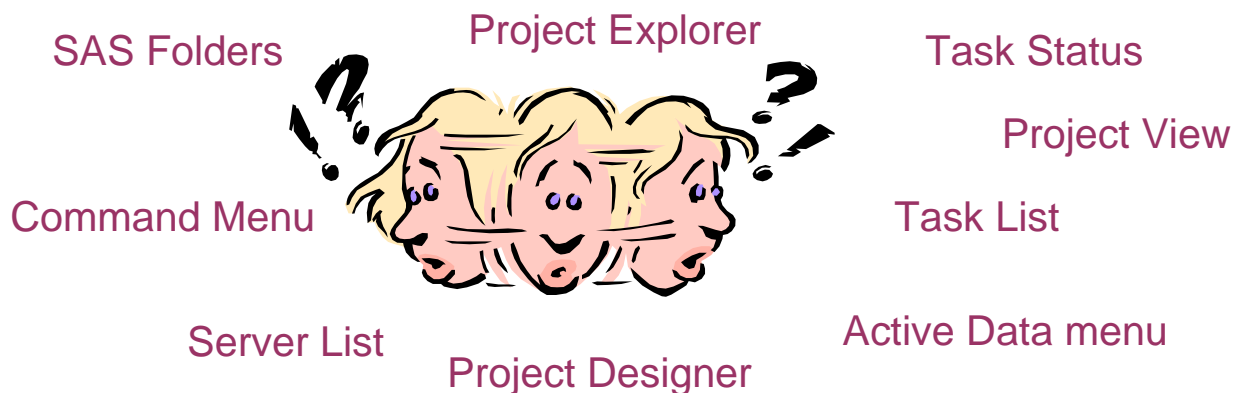
Enterprise Guide version 4.1 will be used.

INTRODUCTION

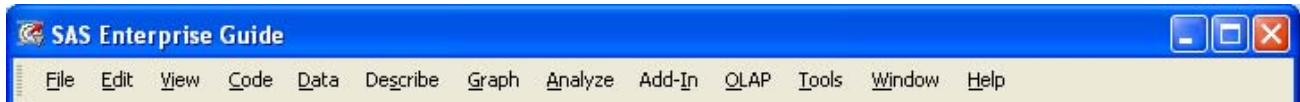
This presentation is designed as a Hands-On Workshop, where the participants follow along and step through the usage of SAS Enterprise Guide. This Proceedings paper follows a similar format.

I KNOW HOW TO PROGRAM IN SAS – HOW DO I NAVIGATE SAS ENTERPRISE GUIDE?

With all the windows and views and menus available in Enterprise Guide, it is hard to know where to start! For many “long –time” SAS programmers, the initial view can send them running right back to their programming environment. But, trust us, the benefits of Enterprise Guide are worth the time it takes to get used to all the windows.

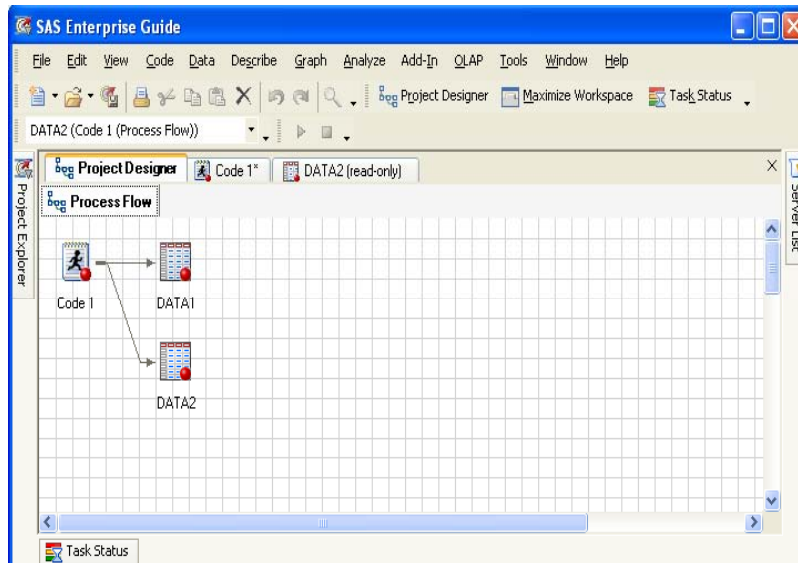
**GETTING STARTED**

The first time you open Enterprise Guide, you will see the Welcome to Enterprise Guide Screen. This screen gives you the option of launching the Enterprise Guide tutorial or starting a new project. If you do not want this to be your default screen when you open Enterprise Guide, you can check the box labeled ‘Do not show this screen again,’ after which you will be taken to the default view with the MAIN MENU displayed at the top of your screen. The main menu allows you to access the various tasks and views that are available to you. Although you may also access the tasks and views through shortcut buttons, the main menu organizes them into logical categories, making them easy to find.




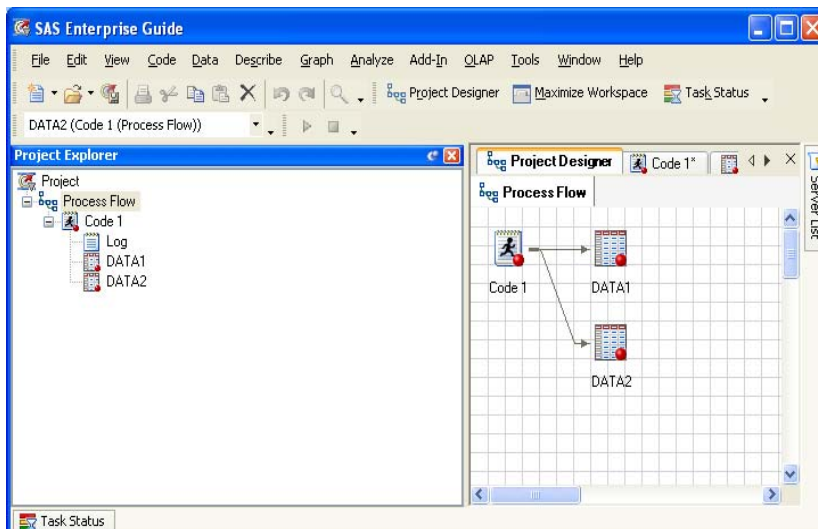
As SAS Programmers, you all know that the tasks you are expected to perform are seldom simple. More often than not, there are a series of SAS programs that you run in a sequence, with different types of output directed to different locations, along with all the intermediate and final SAS datasets that are generated and need to be saved.

Enterprise Guide provides a mechanism to organize your work by allowing you to group sequences of tasks, data items and results into **Projects**. All of the code and tasks that you run within an Enterprise Guide session are part of your active Project and are each displayed as separate **nodes**.



These nodes can then be linked and organized in a logical sequence to form a **Process Flow**. You can save your Process Flow as a part of a Project; then anytime the project is opened the same sequence of tasks can be easily executed. This helps you and your organization distribute and share information easily. The **Project Designer view** is used to display all of the **task, data and code nodes** that make up your Process Flow.

There are a number of ways that you can access your Project Designer view. From the main menu, select **View → Project Designer**. You can also select **F4** or click the  **Project Designer** shortcut button. Notice that the **Project Designer** view also displays the connections between your work components. Again, this allows you to see and visually inspect the organization of tasks, code and data associated with a Project.




You may prefer the **Project Explorer** view, which provides a more traditional "Windows Explorer" Hierarchical view of your projects. Select **View → Project Explorer** from the main menu to open the Project Explorer view. Note that the Project Explorer view does not display the flow (connection) within your projects.

BUT, I AM A PROGRAMMER – HOW DO I RUN MY FAVORITE PROGRAM IN ENTERPRISE GUIDE?

Enterprise Guide enables you to easily run existing SAS programs, using similar steps to what you would follow using the SAS Display Manager windows. Let's run a simple program and review the log and results using Enterprise Guide. Throughout this workshop, we will be using the Shoes dataset from the SASHELP library.

To run an existing program using Enterprise Guide (See Appendix – Program_1 for example program),

- from the **Main Menu**, select **File → Open → Code → Local Computer**
- type the location and name of your program into the **File Name** box, and then click **Open**
- for this example, open Program_1
- the actual code will open in a new window tab
- notice that a code node, Program_1 is added to your Process Flow workspace
- run the program by selecting **Run On SASMain** from the POP MENU in the code node, OR by clicking  .

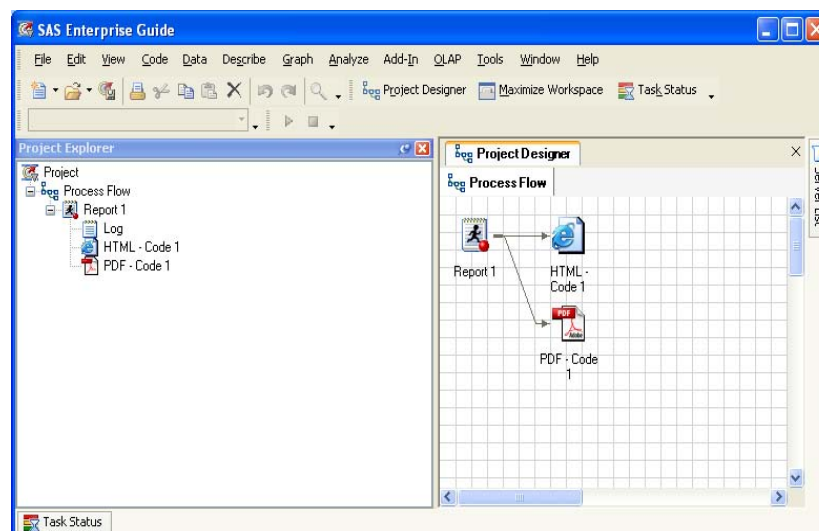
VIEWING LOGS AND RESULTS

To review the log messages, double click the **Log** icon in your **Project Explorer** view. The SAS log will open up in a new window tab.



Helpful Hint: If you would like your SAS Log to open automatically when you have an error, check the **Display SAS Log when Errors Occur** option in your **Results** settings.

Since our example program produces an Excel Spreadsheet report as output, you would use your normal process for opening and reviewing the results (such as Windows Explorer). If the results were **standard SAS output** you would double click on the generated results node in either your Project Designer view or your Project Explorer view and the results will open in a new tab window within Enterprise Guide. Leveraging the functionality of ODS, you can prompt Enterprise Guide to produce different types of output. You can easily produce PDF results along with your standard SAS output by simply changing your Results Options settings. Each type of generated result will display as its own result node in your Project Designer or Project Explorer view.



Helpful Hint: To change the type of Results generated by a specific code node or task, **right click on the node** and select **properties**. Under the **Results**, click the checkbox beside the type of output you want to generate.

CUSTOMIZING THE WINDOWING ENVIRONMENT

Everyone has different preferences about how Enterprise Guide windows should work, and Enterprise Guide provides the flexibility for you to create a work space that works best for you. For example, Enterprise Guide windows can be

- docked and undocked
- resized to make best use of the real estate
- hidden so that you only display what you need and use
- opened as new horizontal or vertical tabs
- easily reopened using the main menu or shortcut buttons.



Helpful Hint: Unhappy with your window changes? Click the **Reset your Docking Windows** button in your Options window.

THAT WAS PAINLESS, BUT I NEED TO BE ABLE TO CHANGE CODE?

Suppose that you need to enhance existing code, rather than just run it. Navigate to the Program_1 code node and double click to open it in a new window tab if it isn't already open; or activate the tab if the code is already open. Once the code is open, you can make changes; you will find the editor commands are identical to the SAS Editor. Then simply rerun the code and review your results. You can save your changes from the main menu by selecting **File → Save**. Remember, each time you save a code node, you are actually saving a **.sas** program file.

If you have selected 'Open the LOG when errors occur' in your Options, then your log will now automatically appear if there are issues.



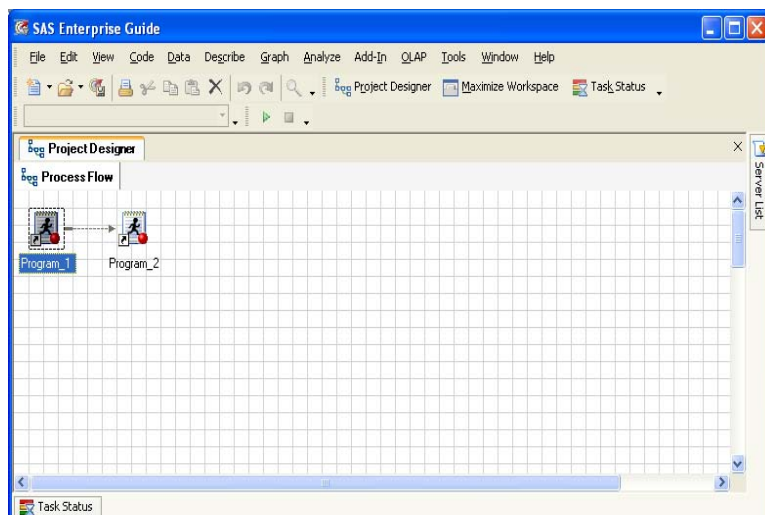
Helpful Hint: If you would like to run just a section of the code, highlight and submit just like you do in the SAS Display Manager Editors.

SPLITTING YOUR PROGRAM INTO LOGICAL STEPS

During program development and testing, do you often work with just one set of steps at a time, confirming results before you move on? If you do, that same logic can easily be applied to your code in Enterprise Guide by defining your program as a series of code nodes. Code nodes can be linked together in the Process flow window, thus providing a clear picture of the steps and intermediate results.

To create a second code node that links to the first code node,

- open an existing program by selecting **File → Open → Code → Local Computer**, from the main menu
- supply the location and name of your program in the **File Name** box, and then click **Open**. For this example, use Program_2 (see the appendix for code stored in Program_2).
- notice that a Program_2 code node is added to your Process Flow workspace.
- link the new code node to the Program_1 code node



- right click on the Program_1 code node in your Project Designer window
- select **Link Program_1 to . . .**
- select **Program_2** and click **OK**.
- You now see the link between the two nodes and have the option of running either a single code node or running the branch that starts with Program_1.

To run both code nodes, right click on your Program_1 code node and select **Run Branch from Program_1**. Once the code in Program_1 has run, Enterprise Guide will automatically run Program_2, generating and displaying all intermediate results along the way.



Helpful Hint: To remove a link between tasks or Code nodes, right click anywhere on the joining arrow and select **Delete**. This will only delete the linkage, not the code nodes.

You'll notice that a separate log node is generated for each code node. By turning on **Project Logging**, you can easily tell Enterprise Guide that you'd like a single SAS log to be generated for all of the tasks and code nodes in your Project. This single Project Log will be created in addition to the individual logs created for each task or code node.



Helpful Hint: If Project Logging is turned on, the log represents a running log of the entire project. To turn on the Project Logging, right click on the **Project** node in the **Project Explorer View** and select **Properties**. Under the **Project Log** tab, click the checkbox labeled **Maintain Project Log**.

WOW – THAT'S EASY – ARE GRAPHICS EASY TOO?

You have now submitted and modified existing code in Enterprise Guide, so let's work on a task from scratch. A picture is worth a thousand words, so it is time to see how Enterprise Guide does with graphics. Using the same sales data used in the first example program (Program_1), you will produce a vertical bar chart of Sales by Region.

- Select **Graph → Bar Chart** from the main menu. This opens the Bar Chart Wizard.
- In the **Bar Chart** screen, select **Simple Vertical Bar**
- In the **Task Roles** screen
 - Drag **Region** to **Column to Chart**
 - Drag **Sales** to **Sum of**.
- In the **Appearance → Options Screen**, change the **Shape** to **Cylinder** from drop down list
- In the **Titles** screen
 - Click on **Graph**
 - Uncheck **Use Default Text**. Enter your own title for your graph.
- To see what code was generated, click on **Preview Code**.
- Close the **Code Preview Window**
- Click on **Run** to create the Bar Chart
- The bar chart opens automatically in a new Tab in your Enterprise Guide session.

LEVERAGING CODE THAT ENTERPRISE GUIDE BUILDS

Behind the scenes, Enterprise Guide created “correct” code! Like most SAS programmers, you would probably like to re-use the code elsewhere.

To access and then save the generated code,

- In the **Project Window**, double click on the **Bar Chart** task node
- Highlight the **GOPTIONS** statements and the **PROC GCHART** statements
- Right click anywhere on the highlighted code, and select **Copy**
- **Close** the **Code Preview** Window
- **Close** the **Bar Chart Task** Window.

To insert the copied code into your program, double click on the appropriate code node and **paste** the code in the desired location. To save the expanded code node, select **File → Save Code As → Local Computer**, and provide the name Program_3.

The expanded code can now be run in the usual fashion. Note that you could also have created a separate code node from this copied code, as discussed earlier. You can make any additional updates and customizations to code

that you've just pasted, such as changing the appearance of the Legend or adding a Where clause to subset the data being graphed.



Helpful Hint: The **Task Status** Window shows you the status of the code or task you are running. If you are running large SAS programs, the Task Status will tell you which task or Procedure is currently running. You can open your Task status view from your Main menu. Select **View → Task Status**.

Each of the built-in tasks will generate SAS code as you step through the Task Panes. This serves as a great way to learn how to code new SAS Procedures since the syntax is available to view and edit.

At this point, save the Project so it is available for another day.

- Select **File → Save Project As → Local Computer**
- Navigate to the file directory where you would like to save your project.
- Enter a name for your Project.
- Click ok.

The next time you open your saved project; you will see the code nodes and links that you defined in this session. Note that any temporary work datasets that you may have created in your session will not be saved with your project. You can easily recreate the work datasets by re-running the project the next time you open it.

CUSTOMIZING THE GRAPH FORMAT

By default, graphs are generated as interactive ActiveX graphs, which are *cool* but can use up a lot of system resources. You may prefer or require a more static format such as HTML or JPEG. You can easily change the default in the Graph Results settings of Options.

Select **Tools → Options → Results → Graphs** and select a different format from the drop down list.

OK – ENTERPRISE GUIDE REALLY WILL HANDLE MY NEEDS BUT HOW CAN I OFFLOAD SOME OF MY WORK (OR DO MORE WORK IN LESS TIME)?

For most SAS programmers, it is a rare occurrence to write a program that is never used again. More typically, you write programs that either run in production, or that are rerun with different input (parameters) on a regular basis. Furthermore, YOU are not always the person that runs the job after it is written. Someone else may be responsible for providing input parameters (changes) and running your code when results are needed. With this in mind, we explore how you can easily set up your code so that others can access and get the results they need. You will do this by creating a Stored Process.

A **Stored Process** is a SAS program that is typically located on a central server (ie: available to multiple users across multiple clients) so that users can run the SAS Code but can't change the SAS code. Designed properly, the Stored Process enables user input to maximize the applicability of the code. For example, you

- can embed business rules and logic so that everyone is creating consistent results
- have change control around any coding updates
- can make your code available to more people
- can provide the users with input parameter choices to take care of minor tweaks in the requirements.

CREATING A STORED PROCESS

Earlier we said that a Stored Process is a SAS Program. That is only half the story; the Stored Process also consists of Metadata that describes the mechanics. So, before we get started, you must have a Metadata Server configured and running in your work environment. Assuming this is complete, create a simple stored process in your project:

- Right click on your **Program_1 code node**

- Select **Create Stored Process**, which will launch the **Create New SAS Stored Process Wizard**

Create New SAS Stored Process Wizard

1 of 7 General Information

Name:
Stored Process for Program_1

Description:

Keywords (comma separated):

Specify the keywords that can be used to describe the stored process. Separate multiple keywords with a comma. Keywords help search utilities identify the stored process. Keywords are optional. [More \(F1\)...](#)

< Back **Next >** Finish Cancel

- In the **General Information** screen, enter a name for your Stored Process and click **Next**

- The second screen will show you the **SAS Code** that will be used in your Stored Process (in this example, Program_1 code). No changes are required so click **Next**

Create New SAS Stored Process Wizard

2 of 7 SAS Code

```

data shoes;
  set sashelp.shoes;
  AverageSalesPerStore = sales / stores;
  label AverageSalesPerStore = 'Average Sales Per Store';
  format AverageSalesPerStore dollar12.2;
ods listing close; ods noresults;
ods tagsets.ExcelXP path = 'c:\workshop\ws110'
  file='HOW_1_Basic_Report.xls' style=statdoc
  options (sheet_name = 'Example 1'
          frozen_headers = 'Yes' autofilter = 'All' );
title "Simple SAS Code Example 1";
proc print data=shoes noobs label ; run;
ods tagsets.ExcelXP close;
ods listing;

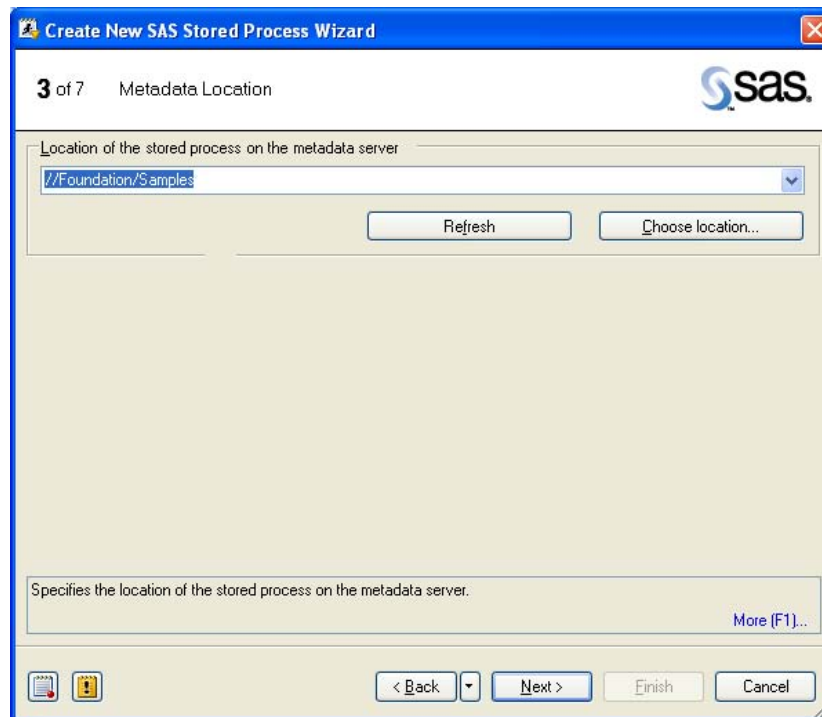
```

Replace with code Include code for Clear code Reset code

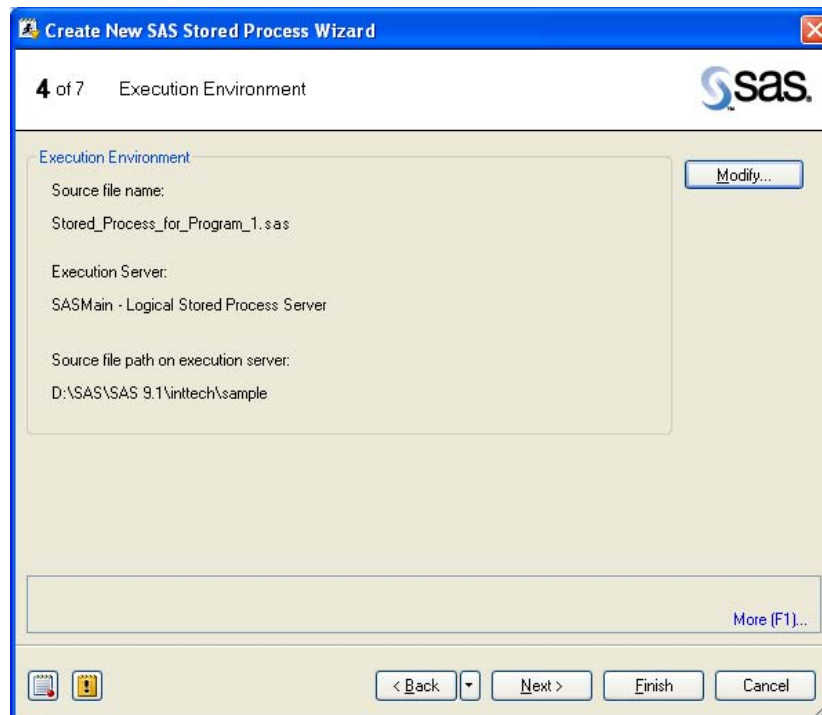
Moves to the next step in the wizard. [More \(F1\)...](#)

< Back **Next >** Finish Cancel

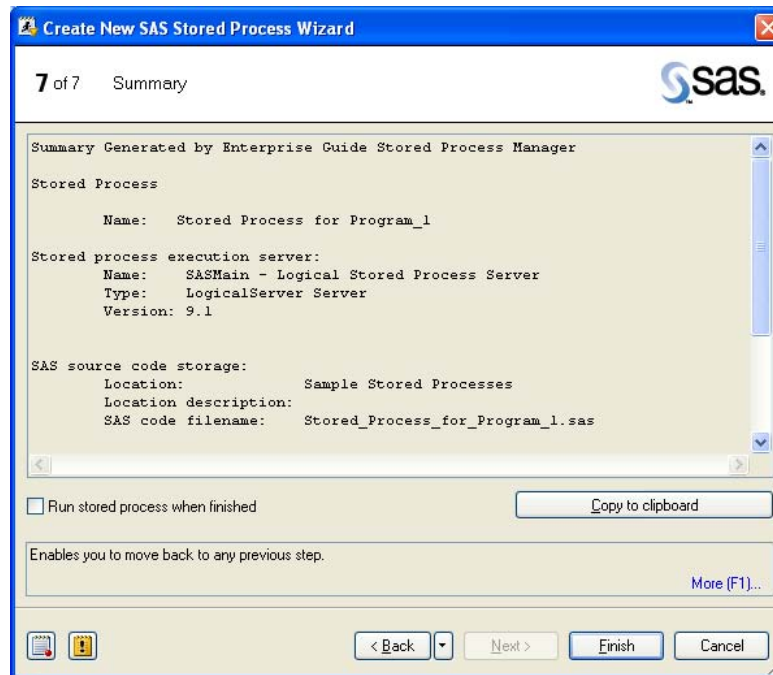
- in the third screen, you need to specify the **Metadata Location** where the metadata for your Stored Process will be saved, then click **Next**. Your administrator can help you determine the location for your Stored Processes. In this example, store the Stored Process metadata in the Samples directory.



- Screen 4 defines the **execution environment**. Here you can modify the source file (where your SAS code will be saved), the Execution server (Workspace server or Stored Process server). Since there are no changes needed in this example, click **Next**

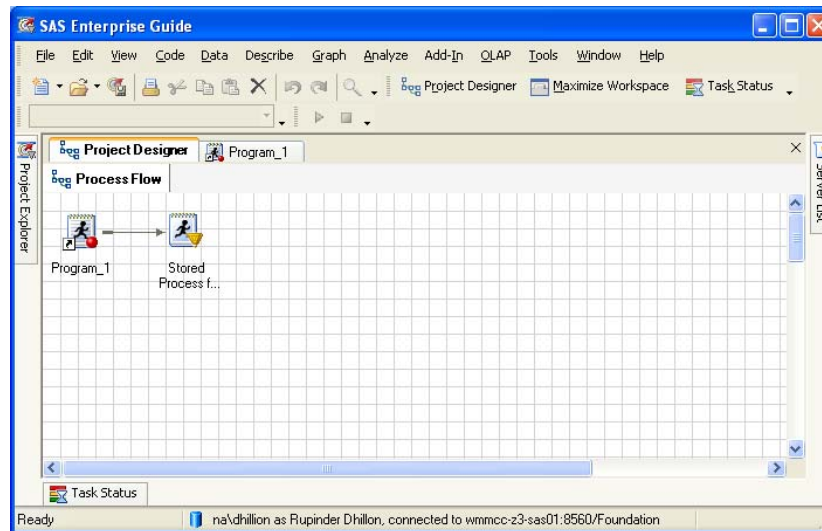


- Finally, screen 7 is a **Summary** of your selections for this stored process. Click **Finish**.



Note: An additional screen would be included if **librefs** needed to be defined.

The stored process is now available in the Project Designer View.



To **run** the Stored Process, right click on the node and select **Run Stored Process Name**. To view the resulting log, right click on the node and select **Open Log**. You may have noticed that we skipped over the parameters screen in this workshop. Parameters are a useful way of providing the user with some flexibility when running a Stored Process. Parameter values that the user enters are passed as macro variables to the SAS code and can be used as such throughout your coding logic. For more information on Parameters, refer to the recommended reading at the end of this paper.

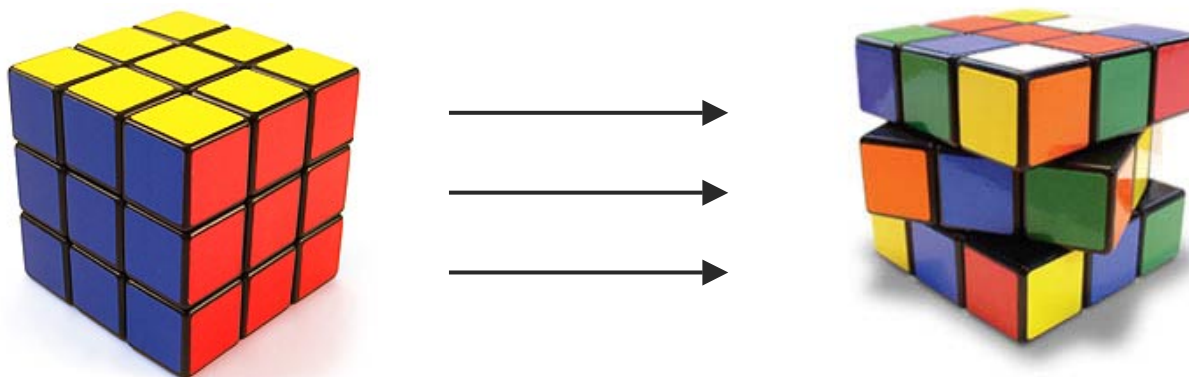


Helpful Hint: To get easy access to your Stored Processes, use the SAS Folders View. Select **View** → **SAS Folders** from the main menu.

NOW – I NEED OLAP!

Many user-applications require immediate results based on huge volumes of data. Most users do NOT want to submit a job that churns away for hours before providing their results. With that in mind, if an application can be sourced from summarized data, then OLAP can help! OLAP, or Online Analytical Processing, allows a user to access and analyze summarized data (an OLAP cube) including the ability to quickly switch to different views of the aggregated business data (***Slice and Dice***). What is the benefit for you – the SAS Programmer?

- Data in an OLAP cube is aggregated, so half the work is already done for you
- Querying OLAP cubes through ***passthru to OLAP*** is much faster than accessing raw (detail) data
- You can create the same reports, but the jobs run much quicker
- ***Drill Down*** (investigating underlying data) is possible.



OLAP TERMINOLOGY

In the cube above, the colors represent different groupings of data into logical categories, called **dimensions**. **Dimensions** might include Product, Geography, Job Roles, Time, etc. The order in which you get information from the summary down to the detailed data is specified by a **hierarchy**. For example, the Geography dimension might be ordered as continent→country→province→city. The data values that are summarized and used for analysis are **measures**.

The different groups of data that are summarized for easy and fast access are referred to as **aggregations**. Cubes can be created using a single source of underlying data or can be created by using a series of tables connected by various keys.

When creating a cube from a single table, you source the data from a **Detail Table** which contains the measures and levels (for defining the hierarchies). When using a series of tables, the underlying data structure is a **Star Schema**, which uses a Fact table along with a set of Dimension tables to define the cube.

CREATING AND VIEWING OLAP CUBES

There is more than one way to create a SAS cube. If you prefer to use a GUI based tool, you can use the Olap Cube Wizard (available in SAS Olap Cube Studio and Data Integration Studio). If you are a coder at heart, you can create, delete and update a SAS cube using PROC OLAP code. For those of you who are somewhere in between, you can use the OLAP Cube Wizard to generate the PROC OLAP code and then tweak and customize it yourself.

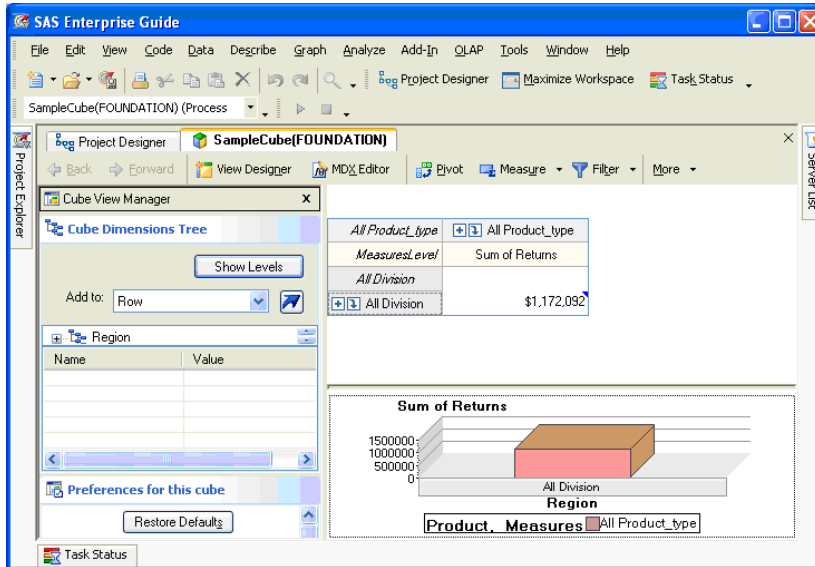
This example uses PROC OLAP to create a cube based on the Shoes detail Table used in previous examples.

- Open the next example Code provided. Select **File → Open Code → Open code from Local Computer**
- Select **Program_4**, click **Open**
- This code will register the metadata for your underlying Detail table, it will create the OLAP cube and register the cube details in the metadata server.
- Submit the code.

Enterprise Guide has an OLAP Cube Viewer that allows you to navigate and drill through an OLAP cube.

- To open the Olap Cube in Enterprise Guide , click **File → Open → Olap Cube**

- Dependent upon your setup, enter the following into the Open OLAP Window (place information specific to your setup where the *** appear) :
 - Olap Server Name: *****
 - Provider: SAS OLAP Data Provider 9.1 (from drop down list)
 - User Name: *****
 - Password: *****
- Click **Connect**
- Check the box next to your newly created Cube – click **Open**.

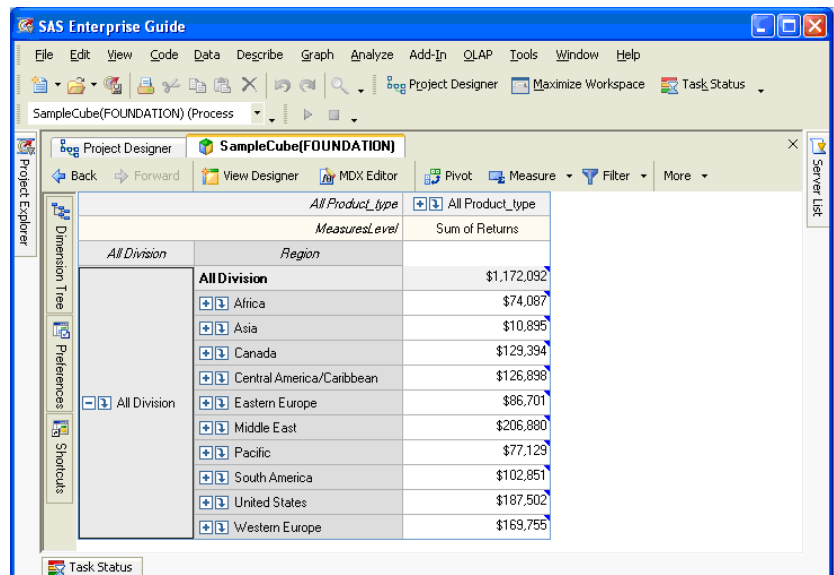


The OLAP Viewer automatically opens displaying the default view of your cube. In this example, we specified the Division Hierarchy as our default (see Program_4 code), so this is the first view we see. Notice that the view is also aggregated to the highest level. You'll also notice that the OLAP Viewer includes a Graphical representation of your current view. As you drill to different dimensions and levels, the graph automatically changes to reflect the data in the current view. For now, remove the graph from the view to maximize the area available for viewing the data.

Helpful Hint: To remove the graph from your Cube Viewer, Click on More in your Cube Viewer Task bar, click Graph.

Let's explore this cube.

- Drill Down on the Product Type – click the downward facing arrow on the 'All Product Type' column.
- Expand the list of Divisions – click the plus sign on the 'All Division' Row.



- Change the displayed measure. Right click on the 'MeasureLevel.' and select 'Edit Measures'
 - Select 'Average Sales' and 'Sum of Stores'
 - Click OK.

SAS Enterprise Guide - SampleCube(FOUNDATION) (Process)

Project Designer | SampleCube(FOUNDATION) | View Designer | MDX Editor | Pivot | Measure | Filter | More

All Product_type		All Product_type		
MeasureLevel		Sum of Returns	Average Sales	Sum of Sales
All Division	Region			
All Division	All Division	\$1,172,092	\$85,700	\$33,851,566
	Africa	\$74,087	\$41,832	\$2,342,588
	Asia	\$10,895	\$32,874	\$460,231
	Canada	\$129,394	\$115,019	\$4,255,712
	Central America/Caribbean	\$126,898	\$114,305	\$3,657,753
	Eastern Europe	\$86,701	\$77,256	\$2,394,940
	Middle East	\$206,880	\$234,657	\$5,631,779
	Pacific	\$77,129	\$51,040	\$2,296,794
	South America	\$102,851	\$45,089	\$2,434,783
	United States	\$187,502	\$137,600	\$5,503,986
	Western Europe	\$169,755	\$78,597	\$4,873,000

- Take a closer look at Canada
 - Right click anywhere on the **Canada** cell. Click 'Isolate Canada'

SAS Enterprise Guide - SampleCube(FOUNDATION) (Process)

Project Designer | SampleCube(FOUNDATION) | View Designer | MDX Editor | Pivot | Measure | Filter | More

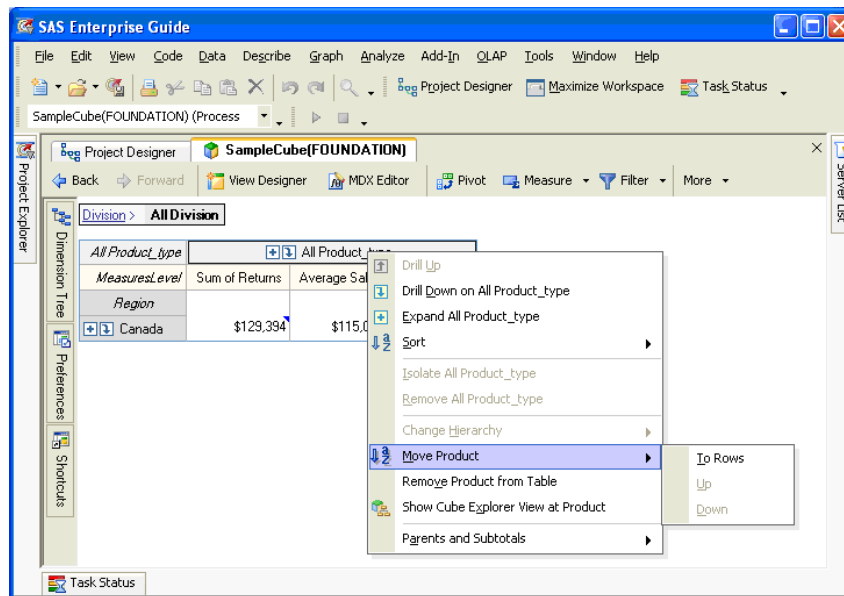
All Product_type		All Product_type		
MeasureLevel		Sum of Returns	Average Sales	Sum of Sales
All Division	Region			
All Division	All Division	\$1,172,092	\$85,700	\$33,851,566
	Africa	\$74,087	\$41,832	\$2,342,588
	Asia	\$10,895	\$32,874	\$460,231
	Canada	\$129,394	\$115,019	\$4,255,712
	Central America/Caribbean	\$126,898	\$114,305	\$3,657,753
	Eastern Europe	\$86,701	\$77,256	\$2,394,940
	Middle East	\$206,880	\$234,657	\$5,631,779
	Pacific	\$77,129	\$51,040	\$2,296,794
	South America	\$102,851	\$45,089	\$2,434,783
	United States	\$187,502	\$137,600	\$5,503,986
	Western Europe	\$169,755	\$78,597	\$4,873,000

Context Menu (over Canada cell):

- Drill Up to All Division
- Drill Down on Canada
- Expand Canada
- Sort
- Isolate Canada**
- Remove Canada
- Change Hierarchy
- Move Region
- Remove Region from Table

- Move the Products dimension from the columns to the rows. Right click on any of the Product

Column headings, click **Move Product** → **To rows**.



- Drill down on the Region to see the subsidiaries. Click the downward facing arrow on the **Canada** column heading.

INCORPORATING OLAP INTO YOUR SAS PROGRAMS

Now that the OLAP cube is created and we have a better idea of the data available in the cube, we want to be able to use the aggregated data to create some reports.

Insert the OLAP query into the existing code in Program_5 (see appendix).

- from the **Main Menu**, select **File** → **Open** → **Code** → **Local Computer**
- Select Program_5 and click open.

Using PROC SQL, you can connect to your cube and query the data like you would a SAS or Database table. The query language that you use is called MDX and while it uses the familiar SQL keywords (select, from, where), that's where the similarities end. So how do you connect to a cube through your code? First, let's look at the familiar syntax:

```
Connect to OLAP (<options>)
Create table MYDATA as select * from connection to OLAP
( select <MDX query here>)
Disconnect from OLAP;
Quit;
```

Building MDX select code is a *little* trickier:

```
SELECT
    {axisSpecification} ON COLUMNS,
    {axisSpecification} ON ROWS
FROM cubeName
WHERE (slicerItems)
```

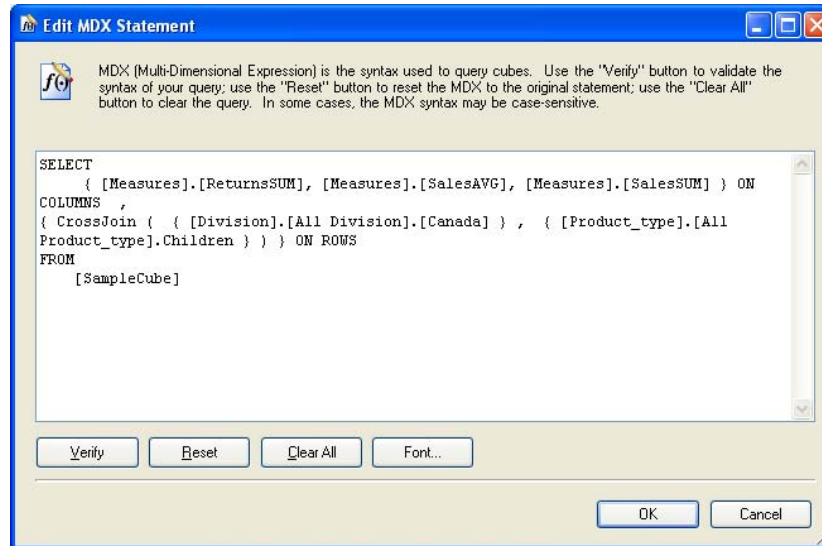
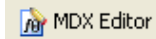
Since Enterprise Guide has a built-in cheat sheet within its OLAP Cube Viewer, you don't need to know too much!



Helpful Hint: You can use the MDX Editor in the Cube Viewer to build and tweak your query code. Simply navigate to the desired Cube view and copy the MDX query code that is built for you.

Lets go back to our Olap Cube viewer and get our MDX Query code.

- To view the MDX code from the Main Menu Select **OLAP** → **View MDX** or click on in your Cube Viewer Window



- Copy the MDX Query. Highlight the text, right click and select **Copy**.
- Paste** the MDX query into your PROC SQL Code. Place your cursor where you want to insert the code, right click and hit **Paste**.
- Submit your code.



Helpful Hint: By creating a 'slice' from your Cube view, you can preview the SAS dataset that will be generated by the MDX query. To create a slice of the view you are looking at, return to your Process Flow in the Project Designer View, right click on the Cube Node and select, Create Slice. This 'slice' is essentially a Work SAS dataset.

WHAT'S NEXT

Now that you know the basics of using Enterprise Guide, there are lots of bells and whistles and additional features to explore. First, check out the use of parameters with Stored Processes (a separate Hands-On Workshop at SAS Global Forum focuses on this topic). Then, begin looking at the other many features that Enterprise Guide and the SAS BI Suite offer.

CONCLUSION

Enterprise Guide 4.1 provides a powerful environment and toolset for end-to-end information delivery. As a SAS programmer, you can easily begin taking advantage of Enterprise Guide to provide a flexible reporting and analytics environment to meet your organization's business needs.

RECOMMENDED READING

To gain a more complete understanding of Enterprise Guide 4.1, the authors recommend the following web link.

- Getting Started with SAS Enterprise Guide – free learning tutorial available from http://www.sas.com/apps/elearning/elearning_details.jsp?pubcode=59271

ACKNOWLEDGMENTS

The authors appreciate the feedback and suggestions provided by Andrew Kuligowski and Peter Eberhardt.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Marje Fecht
Prowerk Consulting
marje.fecht@prowerk.com
www.prowerk.com

Rupinder Dhillon
Dhillon Consulting
Rupinder@dhillonconsulting.com
www.dhillonconsulting.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

APPENDIX – PROGRAMS**PROGRAM_1**

```

data shoes;
  set sashelp.shoes;
  AverageSalesPerStore = sales / stores;
  label AverageSalesPerStore = 'Average Sales Per Store';
  format AverageSalesPerStore dollar12.2;
ods listing close;
ods noresults;
ods tagsets.ExcelXP path = 'c:\workshop\ws110'
  file='HOW_1_Basic_Report.xls' style=statdoc
  options (sheet_name = 'Example 1'
    frozen_headers = 'Yes' autofilter = 'All' );
title "Simple SAS Code Example 1";
proc print data=shoes noobs label ;
run;
ods tagsets.ExcelXP close;
ods listing;
ods results;

```

PROGRAM_2

```

ods listing close;
ods noresults;
ods pdf file = 'c:\workshop\ws110\HOW_1_Shoes_Report.pdf';

proc report data = shoes;

  column ("Location" (Region Subsidiary Stores))
         Product
         ("Sales" (Sales AverageSalesPerStore))
         ;
  define region / order "Region";
  define Subsidiary / order "Subsidiary";
  define Stores / display "Number of Stores";
  define Product / display "Product";
  define Sales / Sum "Sales";
  define AverageSalesPerStore / Sum "Avg Sales per Store";
  break after region / summarize;
  rbreak after / summarize;
run;

ods pdf close;
ods listing;
ods results;

```

PROGRAM_4

```

/* Macro Variables for use in Samples code */
%let SERVER = server.company.com; /* Metadata Server Name or IP Address */
%let PORT = 8561; /* Metadata Port Number */
%let PROTOCOL = Bridge; /* Metadata connection protocol */
%let USER = sasdmo; /* User ID with ReadMetadata and WriteMetadata permissions */
%let PASSWORD = sasdmo1; /* Password for the User Id */
%let REPOSITORY = Foundation; /* Repository in which to build data and cube */
%let LIBNAME = sample; /* Libref which references permanent data location */
%let LIBPATH = c:\sasdata; /* Directory in which to store data */
%let SCHEMA = SASMain - OLAP Schema; /* OLAP Schema in which to store cube */
%let CUBEPATH = \Data; /* Physical location for OLAP Cube files */
%let CUBE = SampleCube ;

```

```

/* Options for Metadata Server connections */

libname &libname "&libpath";

option metaserver="&SERVER"
    metaport=&PORT
    metaprotocol="&PROTOCOL"
    metarepository="&REPOSITORY"
    metauser="&USER"
    metapass="&PASSWORD";

data &libname..shoes;
    set sashelp.shoes;
run;

/* Register tables using PROC METALIB */
proc metalib;
    omr (library="&libname" metarepository="&repository");
    update_rule (delete); * Overwrite dups regardless;
    select (shoes);
report;
run;

/* Delete any cube that exists with the cube name */
PROC OLAP cube=&cube delete;
    METASVR repository="&repository" olap_schema="&schema";
RUN;

/* Create the OLAP cube */
PROC OLAP
    Data=&libname..Shoes
    DrillThrough_Table=&libname..shoes
    cube=&cube
    Path="&cubePath"
    Description="Shoes Cube for HOW Examples"
    ;

METASVR repository="&repository" olap_schema="&schema";

DIMENSION Division hierarchies=(Division )
    CAPTION='Region'
    SORT_ORDER=ASCENDING ;

    HIERARCHY Division ALL_MEMBER='All Division'
    levels=( Region Subsidiary )
    CAPTION='Division'
    DEFAULT
    ;

    LEVEL Subsidiary
    CAPTION='Subsidiary'
    SORT_ORDER=ASCENDING
    ;

    LEVEL Region
    CAPTION='Region'
    SORT_ORDER=ASCENDING
    ;

DIMENSION Product_type hierarchies=(Product_type )

```

```

    CAPTION='Product'
    SORT_ORDER=ASCENDING ;

    HIERARCHY Product_type ALL_MEMBER='All Product_type'
    levels=( Product )
    CAPTION='Product'
    DEFAULT
    ;

    LEVEL Product
    CAPTION='Product'
    SORT_ORDER=ASCENDING
    ;

    MEASURE ReturnsSUM
    STAT=SUM
    COLUMN>Returns
    CAPTION='Sum of Returns'
    FORMAT=DOLLAR12.
    DEFAULT
    ;

    MEASURE InventorySUM
    STAT=SUM
    COLUMN=Inventory
    CAPTION='Sum of Inventory'
    FORMAT=DOLLAR12.
    ;

    MEASURE SalesAVG
    STAT=AVG
    COLUMN=Sales
    CAPTION='Average Sales'
    FORMAT=DOLLAR12.
    ;

    MEASURE SalesSUM
    STAT=SUM
    COLUMN=Sales
    CAPTION='Sum of Sales'
    FORMAT=DOLLAR12.
    ;

    MEASURE StoresSUM
    STAT=SUM
    COLUMN=Stores
    CAPTION='Sum of Stores'
    FORMAT=12.
    ;

    AGGREGATION Region
    Subsidiary
    Product
    / NAME='DEFAULT'
    ;

    DEFINE
    MEMBER '[Shoes].[Measures].[Sales per Store]' AS
    '([Measures].[SalesSUM] / [Measures].[StoresSUM])', FORMAT_STRING = "dollar30.";
    RUN;

```

PROGRAM_5

```

/* Macro Variables for use in Samples code */
%let SERVER = server.company.com; /* Metadata Server Name or IP Address */
%let PORT = 8561; /* Metadata Port Number */
%let PROTOCOL = Bridge; /* Metadata connection protocol */
%let USER = sasdemo; /* User ID with ReadMetadata and WriteMetadata permissions */
%let PASSWORD = sasdemo1; /* Password for the User Id */
%let REPOSITORY = Foundation; /* Repository in which to build data and cube */
%let LIBNAME = sample; /* Libref which references permanent data location */
%let LIBPATH = c:\sasdata; /* Directory in which to store data */
%let SCHEMA = SASMain - OLAP Schema; /* OLAP Schema in which to store cube */
%let CUBEPATH = \Data; /* Physical location for OLAP Cube files */
%let CUBE = SampleCube ;
%let Olap_Port = 5451;

```

Proc SQL;

```

Connect to OLAP (host="&server" port=&olap_port protocol=&protocol user="&user"
                pass="&password"
                repository="&repository"
                olap_schema="&schema");

```

```

Create table CanadaData as select * from connection to OLAP
( insert MDX query here );
Disconnect from OLAP;
Quit;

```

```

Ods pdf file = 'c:\workshop\ws110\Sales in Canada';
Proc report data = CanadaData;
Title1 j=c "Shoe Sales in Canada";
Columns subsidiary StoresSUM SalesAVG;
Define subsidiary / order "subsidiary";
Define StoresSUM / display "Number of stores";
Define SalesAVG / sum "Average Sales";
break after subsidiary / summarize;
rbreak after / summarize;
Run;
ODS PDF close;

```