

Paper 036-2010

Interactive and Efficient Macro Programming with Prompts in SAS® Enterprise Guide® 4.2

Kenneth Sucher, SAS Institute Inc., Washington, DC

ABSTRACT

SAS Enterprise Guide 4.2 is the newest version of the front-end Windows client that provides a point-and-click interface to access and analyze your data in SAS. Longtime users of SAS, especially those who already take advantage of macro processing, might shy away from SAS Enterprise Guide, seeing it as a tool for novice users or useful only for creating a stored process. This paper focuses on macro processing use (called “prompts” in version 4.2) inside the SAS Enterprise Guide environment. I intend to demonstrate some advanced techniques that are not only very practical, but only possible in the SAS Enterprise Guide environment. Topics covered include: creating and using prompts in tasks and queries, multiple value and range prompts, running tasks conditionally, and adding run-time prompts to a user’s own code. This paper is especially useful for current SAS programmers who are interested in increasing the functionality of their macro programming skills, as well as non-macro users looking for a less programmatic introduction to macro processing within the SAS Enterprise Guide environment.

INTRODUCTION

SAS Enterprise Guide 4.2 provides a point-and-click interface that generates code that you can use to manage data and create reports. In addition, it includes a full programming interface that can be used to write, edit, and submit SAS code. SAS Enterprise Guide also includes a feature called “prompts.” Prompts work by requesting input from the user when a task, query, or SAS program is run. Prompts create SAS macro variables whose values are assigned at run time based on user input. Prompts act like place holders in the code that is generated by SAS Enterprise Guide. These place holders are given values by a user within a prompt window upon execution. As we’ll see, prompts can be used in task, queries, and SAS programs.

This paper will discuss using prompts in point-and-click tasks and queries, as well as your own SAS programs submitted in SAS Enterprise Guide. Many SAS users might be unfamiliar with prompts in SAS Enterprise Guide, or they might shy away from SAS Enterprise Guide thinking of it as merely a tool for beginners or non-programmers. Or they might think that any practical use requires Business Intelligence (BI) tools and stored processes. This paper intends to show these types of SAS users the advantage and power of SAS Enterprise Guide prompts. We will see how prompts can be used to create interactive projects with more informative results. We will also see ways to enhance the code generated by SAS Enterprise Guide to make it dynamic, efficient, and useful. And for those programmers who are already familiar with the macro language, we’ll see ways to make a user’s own macro code user-friendly and interactive by adding prompts.

Throughout this paper, we’ll work through a SAS Enterprise Guide project with data from a fictitious company called Orion Star Sports & Outdoors. Specifically, we’ll use information about customer orders with an overall goal of determining what our top five products sold are. This paper will first cover the basics of prompting in tasks and queries by working through an example that uses a prompt to substitute a variable in a task and another prompt to substitute a variable’s value in a query filter. Next, we’ll discuss more advanced prompting topics such as having a prompt represent multiple values and ranges of values in a query filter. We’ll then discuss ways to edit query code to include multiple values and ranged information in such places as titles and footnotes. Finally, we’ll learn how to add prompts to user-generated code and create conditional processing prompts to apply to our tasks, queries, and programs so that they run only if a condition is met.

PROMPTING BASICS IN A TASK

Let’s first discuss some prompting basics and include one in a task within the SAS Enterprise Guide 4.2 environment. The **Prompt Manager** is used to create, edit, and delete prompts. Once a prompt has been created and defined, it is stored as part of your project and can be used in tasks, queries, and SAS programs. Prompts connected to tasks are typically used to represent variable names. This enables a user to select a variable from a list at run time to be used in a specific task role. When you are creating the prompt, a variable list can be populated with variable names from a data source. Let’s consider this example.

Suppose we've been tasked to find out the top five products sold by our Orion Star Sports & Outdoors company. The data available includes a table named *Orders* that contains order information from 2003 to 2007. Using SAS Enterprise Guide, we plan to run a Summary Statistics task based on the *Orders* table. This task generates a **PROC MEANS** step that calculates summary statistics based on groups within the data. The intent is to create a report that groups together the products that were ordered and sums numeric information about these products (such as Profit, Cost, and Quantity). Because we're not quite sure what variable or variables we'd like to use in our final report, we'll run this task multiple times on different variables to help us decide.

To simplify this process, let's create a prompt by specifying some general properties, and then populate the prompt with a list of possible analysis variables we'd like to select from when running the task that generates the necessary report. To start, we first need to add the *Orders* table into a new SAS Enterprise Guide project, and then we'll go into the **Prompt Manager** to create the needed prompt.

To-Do:

1. Inside a new SAS Enterprise Guide project, select **File** ⇒ **Open** and navigate to the location where you stored the example data. Highlight both *Orders.sas7bdat* and *Products.sas7bdat* and select **Open** to add the tables to the project. We'll use just the *Orders* table now and *Products* a bit later.
2. Select **View** ⇒ **Prompt Manager**, and then click **Add** in the Prompt Manager window located in the resources pane at the bottom left of the screen.
3. In the Add New Prompt window, select the **General** tab to create the name of our prompt. Type `StatVar` in the **Name** field. This will be the name of the macro variable that will appear in the generated code once it is used. Type `Select an analysis variable:` in the **Displayed Text** field. This is what the user will see when prompted for a value. Select the **Requires a non-blank value** check box.¹
4. Select the **Prompt Type and Values** tab to specify the type of prompt and its values. Because we're going to use this prompt to represent a variable in our task, select **Variable** from the **Prompt Type** menu.
We now have the option of typing variable names or loading variable names from a selected data set. Let's load some names in from our Orders table.
5. Select **Numeric** in the Variable Type window, and then select **Load Values**. This will load all the numeric variable names from the selected data source. Select **Project**, and then select the *Orders* table and click **Open**.
Notice we now have a list of every numeric variable from the Orders table. We don't need all of them. Let's leave just a few variables that we think might be useful in our analysis.
6. Hold down the **CTRL** key and select variables to highlight and delete (by hitting the X) leaving only Quantity, Total_Retail_Price, and Profit.² Your window should look like Figure 1. Click **OK** to create the prompt and exit the window.

¹ Requiring a non-blank value helps to avoid possible syntax errors and other undesirable results.

² If preferred, we also have the option of allowing a user to specify additional values by checking the box at the bottom of this window and/or specifying a default value by checking the **Default value: box** and entering a value.

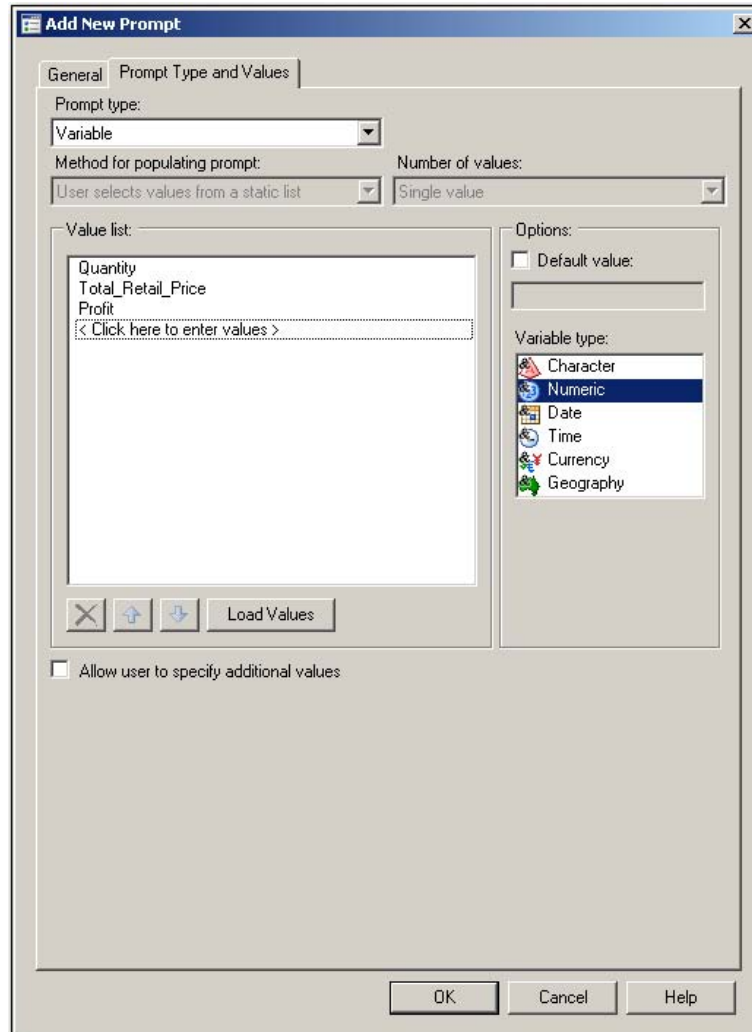


Figure 1. Add New Prompt

We now have a prompt called StatVar that we can use as a substitute for variables in our project. We'll use StatVar to replace a variable in the Summary Statistics task. This task generates standard descriptive statistics. We can use this task to produce a summary report of a given analysis variable grouped by the different products sold. If we use the StatVar prompt as a replacement for the analysis variable, we'll have a few options for the report. For example, we could have a report showing the sum of profit for each product or one showing a total quantity ordered for each product. We'll do that in the following steps.

To-Do:

1. Double-click on the *Orders* table in your project and select **Describe** ⇒ **Summary Statistics** from the menu on top of the data grid. Make sure you open the Summary Statistics *task* and *not* the Summary Statistics wizard.
2. With **Data** highlighted in the selection pane, drag **Product_ID** to the CLASSIFICATION variable role.
Product_ID contains numeric product identifiers. It's the only variable in the *Orders* table that allows us to uniquely identify the different products sold.
3. The StatVar prompt that we created in the **Prompt Manager** should be located at the bottom of the list of variables to assign. Drag it to the ANALYSIS variable role.

4. Highlight **Basic** under **Statistics** in the selection pane. Select the sum statistic and deselect everything else. Use the **Maximum decimal places** menu option to select **2**. This option limits the results to two decimal places in the report.

Let's also add the selected StatVar prompt value to the title of our report.

5. Select **Titles** in the selection pane. With **Analysis** highlighted, deselect **Use Default Text** and edit the title to read *Total & StatVar by Product*.
6. Click **Run**. Notice the prompt window that appears. It contains all the information that we specified in the **Prompt Manager**. Select **Quantity** in the pull-down list. Click **Run**.

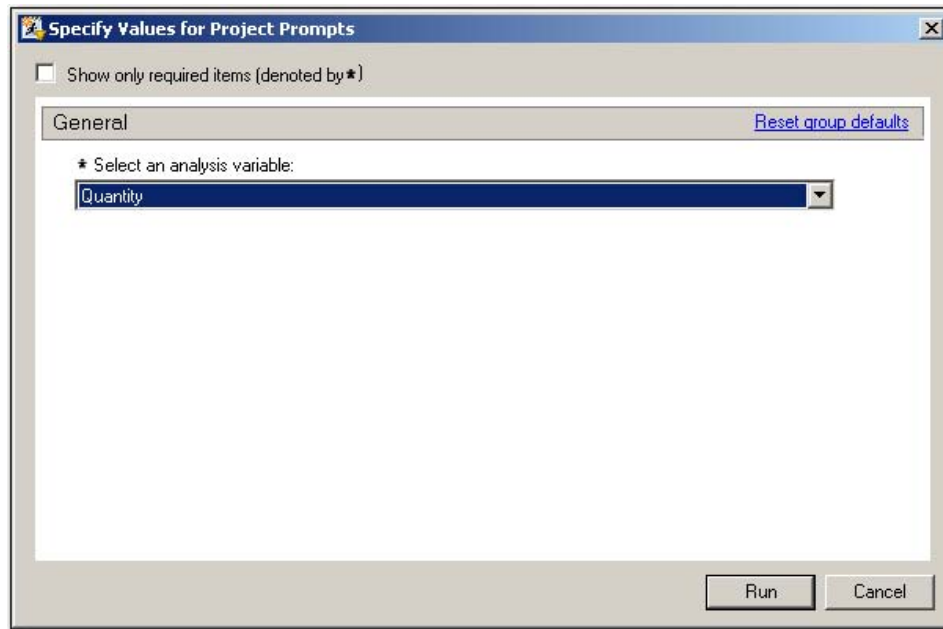


Figure 2. Prompt Window

The report that was generated includes total products ordered by Quantity. It also includes the selected analysis variable *Quantity* in the report title. Before we discuss this report further, let's refresh the results and generate a similar one with Profit as our analysis variable.

To-Do:

1. Click the **Refresh** button located above the report.
Refresh causes the code to be resubmitted behind the scenes and the prompt window to reappear.
2. Select **Profit** in the pull-down list and click **Run** to generate the report shown below.

Analysis Variable : Profit	
Product ID	Sum
210200100009	38.40
210200100017	21.65
210200200022	21.90
210200200023	11.55
210200300006	6.60
210200300007	25.10
210200300052	21.70
210200400020	18.90
210200400070	20.70
210200500002	20.70
210200500006	24.10
210200500007	43.20
210200500016	30.25
210200600056	27.65
210200600067	152.40
210200600085	41.20
210200600112	12.55
210200700016	14.30
210200900004	59.90
210200900033	31.00
210200900038	33.00
210201000050	21.10
210201000067	33.60
210201000126	4.20
210201000198	66.60
210201000199	67.65
210201100004	27.00
210201100018	13.75

Figure 3. Summary Statistics Task Results

Notice the new table in Figure 3 is similar to the previous results except now we have a table with total profits by product. As you can see, this report is too long and ineffective with just product ID numbers and no names. A better option might be to create a report with just the top five products based on sum of profit. The **Query Builder** is better equipped for this type of table. We'll see how to create it in the next section. Before we do, let's examine the generated code for this Summary Statistics task. You can view this code by selecting the **Code** tab above the generated report in the workspace area and scrolling down to the **PROC MEANS** step.

```
TITLE;
TITLE1 "Total &StatVar by Product";
FOOTNOTE;
FOOTNOTE1 "Generated by the SAS System (&SASSERVERNAME, &SYSSCPL) on
%TRIM(%QSYSFUNC(DATE()), NLDATE20.) at %TRIM(%SYSFUNC(TIME()), NLTIMAP20.)";

PROC MEANS DATA=WORK.SORTTempTableSorted
  FW=12
  PRINTALLTYPES
  CHARTYPE
  MAXDEC=2
  NWAY SUM NONOBS      ;
  VAR "&StatVar"n;
  CLASS Product_ID /  ORDER=UNFORMATTED ASCENDING;

RUN;
```

This is the **PROC MEANS** step that was generated by the task. Notice the TITLE and FOOTNOTE statements inserted above it. We can edit this information in the Titles selection window. SAS Enterprise Guide automatically includes the footnote shown. The footnote includes automatic macro variables and macro functions to generate information such as the SAS release, current date and time, and so on. Users with macro programming knowledge

can easily edit this footnote in the same location we edited the title. In addition, notice the &StatVar macro reference in TITLE1. This macro resolves to the value that was selected in the prompt window upon execution. Because we used the StatVar prompt as our Analysis variable in the task, &StatVar is included in the VAR statement in the PROC MEANS code. It's surrounded by double quotation marks and followed by an 'n' so that any value entered in the prompt window upon execution will be accepted as a valid variable name when SAS checks the syntax. Whether that variable exists or not depends on the data. Users with programming knowledge can easily edit this code to further enhance the report. Code modifications can be done in both the Preview Code window within the task or by creating and editing a copy of the generated code. However, instead of making any changes here, let's use the **Query Builder** to more easily control the content and size of our report.

PROMPTING BASICS IN THE QUERY BUILDER

The **Query Builder** allows us to do a variety of things with our data. We can select rows and columns, sort, compute new columns, join tables, group, summarize, and modify column attributes. We can also use prompts in our queries. Text, numeric, and date type prompts are used to represent variable values for a filter in a query. We can even create the prompt by accessing the **Prompt Manager** from inside the **Query Builder**.

Let's use the **Query Builder** to create a report of the **Top 5 Products by Total Profit**. We can do this by joining the *Orders* table to the *Products* table by the column Product_ID. The *Products* table contains information about all of our products including their names. We'll group the results by the various product names while summing the profits made within the groups. We're also going to subset the results by order type. This will allow us to choose orders that are either from Internet, catalog, or retail sales. To do this, we'll create a prompt based on the values of the column Order_Type. Order_Type, however, is a numeric variable with coded information. For example, Internet orders are coded as 3. To make this prompt and the report more user-friendly and informative, let's create a user-defined format called ORDTYPFMT.

To-Do:

1. Select **File** ⇒ **New** ⇒ **Program** and then enter the following **PROC FORMAT** step³ in the program window:

```
PROC FORMAT LIB=SASUSER;
    VALUE OrdTypFmt
        1 = "Retail Sales"
        2 = "Catalog Sales"
        3 = "Internet Sales";
RUN;
```

2. Click **Run**. Verify the following message in the log:

```
NOTE: Format ORDTYPFMT has been written to SASUSER.FORMATS.
```

*We'll use the ORDTYPFMT format to help create a prompt and enhance the report. Let's jump into the **Query Builder**.*

3. Right-click on the *Orders* table in your project and select **Query Builder**. Name the query **Top 5 Products**. We do not need to name the output table since we're only using the query to create a report.
4. We need product names from the *Products* table. Select **Add Tables** and double-click on the *Products* table in your project. This automatically joins *Products* to *Orders* by the matching column-name Product_ID.
5. Verify that the **Select Data** tab is active and drag the Product_Name and Profit columns over. Change the Summary pull-down option to **SUM** for Profit. This automatically groups products and creates a summary of profit for each product name. See Figure 4 below.

³ The Format task can also be used here if preferred.

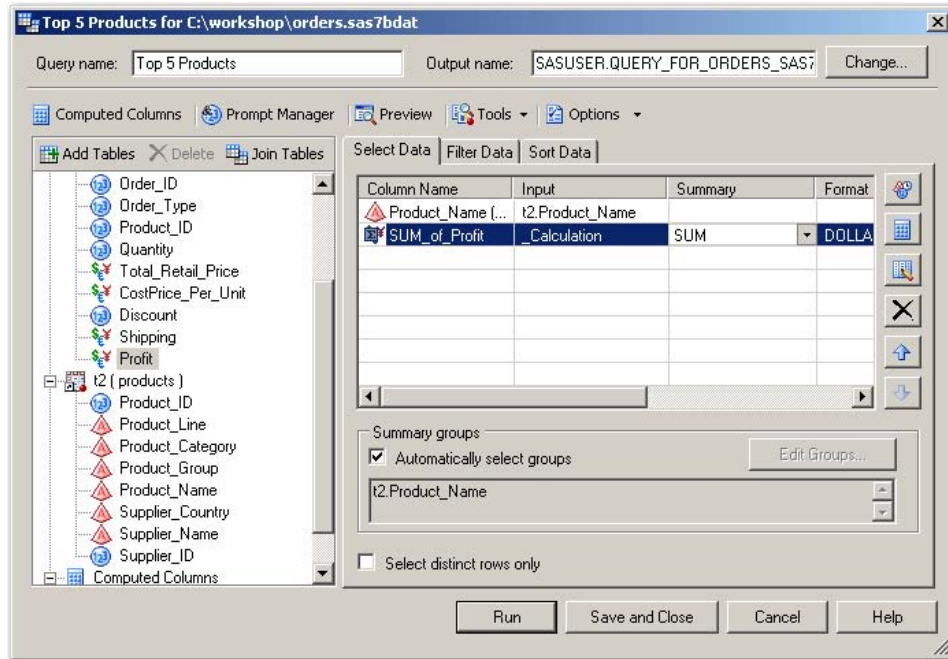


Figure 4. Query Builder

6. Select **Prompt Manager** within the **Query Builder** and click **Add** to create the **OrderType** prompt.
7. Name your prompt **OrderType** and type *Choose an order type:* in the **Displayed Text** field. Select the **Requires a non-blank value** check box.
8. In the **Prompt Type and Values** tab, select **Numeric**.
Numeric is used because the prompt we're creating will contain a selection of numeric order type values.
9. Select **User selects value from a static list** in the **Method for populating prompt** menu. Verify that **Single value** is selected in the **Number of Values** field. Select the **Allow only integer values** check box.⁴
10. Select **Get Values...** located near the bottom right of the window to populate the prompt with the values of the column **Order_Type**. See Figure 5 below.

⁴ Since we're populating this prompt with a selection of values, the integer only value option is used so only integer values are displayed in the prompt. For example, the prompt will display a 2 instead of 2.0. Also notice the option to specify a minimum and maximum value; this isn't necessary for our prompt since users will select from a list.

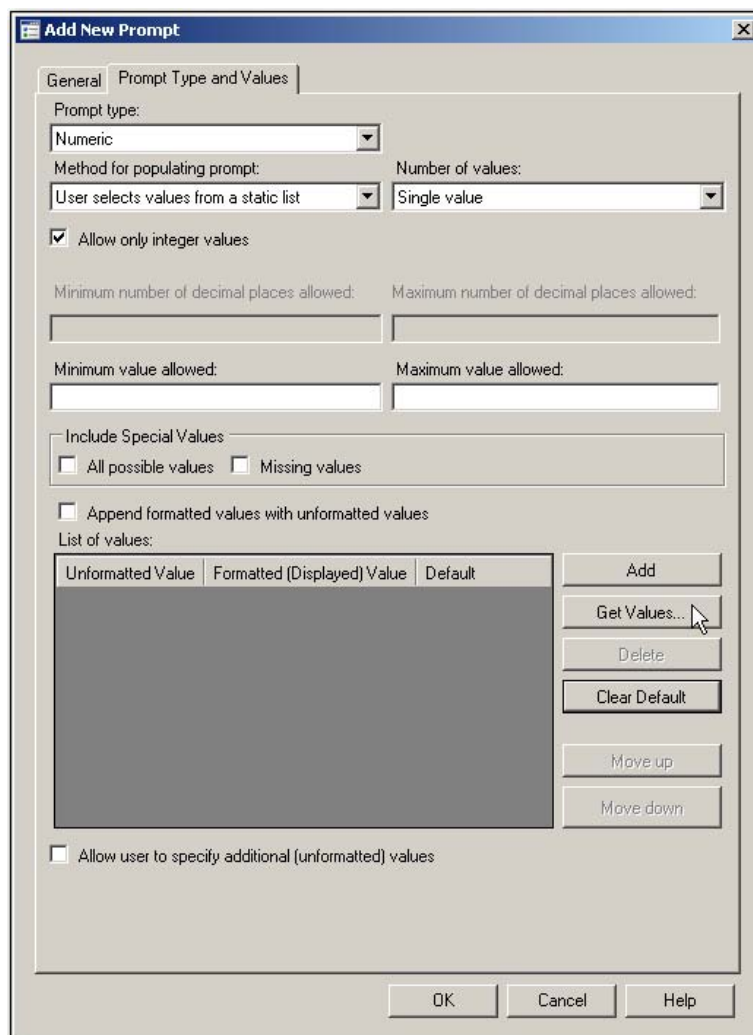


Figure 5. Add New Prompt Window: Numeric

11. In the Get Values window, select **Browse**. Click the **Project** icon to the left. This populates the window with a list of tables from the current project. Highlight *Orders* and select **Open**.
12. Use the **Column** menu under **Unformatted Values** to select *Order_Type*. Also select *Order_Type* in the menu under **Formatted (Displayed) Values** and click **Select** to apply a format.
13. In the Formats window, select **User Defined** and then highlight *ORDTYPFMT*. Click **OK**.
14. Select **Get Values...** to populate the Available values window. Select the double arrow with the plus mark to send all the values over to the Selected values window to the right. Use the up and down arrows on the right to order the list numerically.

Notice that by adding ORDTYPFMT we were able to format the prompt values with more descriptive labels. When a user is prompted, they will see the formatted values⁵ but in the code the unformatted values will get inserted. For example, if a user selects Internet as the value for the OrderType prompt, the OrderType macro variable in the code will resolve to 3.
15. Your window should now look like Figure 6. Click **OK**, then **OK** again, and **Close** to exit and create the prompt.

⁵Select the **Append Unformatted Values with Formatted Values** option in the **Add New Prompt** window if you'd like users to see both. And if you need to allow users to specify values in addition to those in the list select the check box at the bottom of the window.

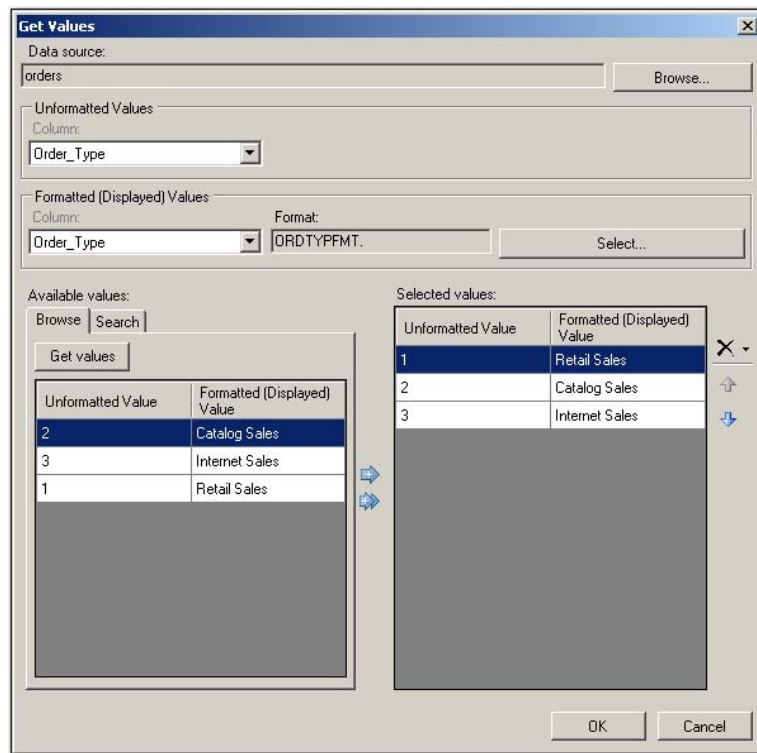


Figure 6. Get Values

Now that we have the OrderType prompt created, let's use it in our query filter.

16. Select the **Filter Data** tab and drag and drop the Order_Type column from the left into the **Filter the raw data** section on the right.
17. In the New Filter window, verify that the selected operator is **Equal to**. Use the pull-down arrow in the **Value** field. Select the **Prompts** tab and highlight &OrderType. For now, deselect the **Generate filter for a prompt value option**.

Notice the generated WHERE statement in the preview window⁶: `t1.Order_Type = &OrderType`. &OrderType is simply a macro reference that will resolve to the unformatted value of the prompt selection.

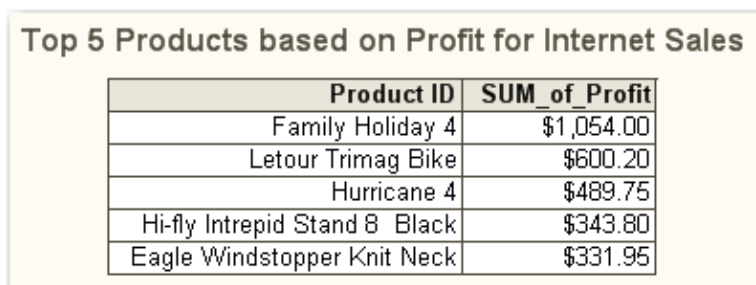
18. Click **Finish** to exit the New Filter window.
19. Select the **Sort Data** Tab. Drag and drop the Sum_of_Profit column we created earlier (labeled _Calculation) to the window to the right. Change the **Sort direction** to **Descending**.
The report will now be in descending order based on total profits. If we limit the number of output rows to five, we'll have the top 5 profit earning products. We can do this in the Query Options window, along with changing the output type for this query to Report, and creating a title.
20. Select the **Options** pull-down list and choose **Options for this query**.
21. In the Query Options window, select the **Override the corresponding default settings in Tools->Options** check box and select **Report** underneath.
22. Select the **Limit the number of rows to save in output** check box and enter **5** in the field to the right.
23. Lastly, select **Titles** in the selection pane. Select the **Override the corresponding default settings in Tools ⇒ Options** check box for both titles and footnotes. Delete the contents in the footnote text field at the bottom and add the following for the title text field at the top:

⁶ T1 is an alias for the ORDERS table.

Top 5 Products based on Profit for %sysfunc(putn(&OrderType,ORDTYPFMT.))

If we were to simply include the title: *Top 5 Products based on Profit for &OrderType*, we'd only see the unformatted value of the order type selected in the report title. Using the macro function %sysfunc, we can take advantage of the PUTN function to apply our user-defined format to the prompt value.⁷

24. Click **OK** and then **Run** to generate the report. Select **Internet Sales** when prompted and click **Run** to create the report shown in Figure 7.



Product ID	SUM_of Profit
Family Holiday 4	\$1,054.00
Letour Trimag Bike	\$600.20
Hurricane 4	\$489.75
Hi-fly Intrepid Stand 8 Black	\$343.80
Eagle Windstopper Knit Neck	\$331.95

Figure 7. Results of Single Value Query

We now have a report showing the top five products by profit for Internet sales. And if we refresh this query, we can change the value of order type to Catalog or Retail sales. By creating a prompt based on the values of a variable, and using a user-defined format to apply to the values of the variable, we were able to create a descriptive list of items for a user to choose from. The unformatted values were inserted into the generated code where needed. You can also see how using macro functions can help create things like more informative titles (and footnotes) by grabbing the formatted values of the created prompt. We can still do more. At this point, the data included in our report spans multiple years. We can also use prompts to filter for things like a range of order dates we'd like to include in the analysis. These types of prompts are appropriately called range prompts and we'll talk more about them in the next section.

RANGE PROMPTS

Range prompts allow lower and upper boundaries to be entered in the prompt window. You can build ranges for a variety of different values including numbers, text, and dates. For example, if we needed to query a table with payroll information we could look at employees who made salaries between \$25,000 and \$50,000. For our scenario, we'd like the ability to look at orders that were placed during a certain period. To do so, we need to create a new prompt representing a date range and add it to the filter for the **Top 5 Products** query.

To-Do:

1. Modify the **Top 5 Products** query by selecting **Modify Task** above the recently created report in the workspace area.
2. Select **Prompt Manager** and click **Add** to create the new range prompt.
3. Name your new prompt **DateRange** and type *Select an order date range:* in the **Displayed Text** field. Select the **Requires a non-blank value** check box.
4. In the **Prompt Type and Values** tab, select **Date Range**. Under **Date type**, you have the option of choosing day, week, month, quarter, or year to define your boundary specifications. Select **Day** as your **Date type**.
5. Enter **01JAN2003**⁸ into the **Minimum value allowed** field since the *Orders* table doesn't contain orders before 2003.

⁷ The PUTN function is used to apply numeric formats to a value. A PUTC function is used to apply character formats.

⁸ Dates can be entered or selected by using the calendar option. This will also be true when selecting dates in the prompt window.

6. To set up a default custom date range, select **Custom** from the **Range type** menu. Enter 01JAN2003 in the **From:** field and select **Today:** in the **To** field. This creates a default prompt value that represents all orders from the first of January 2003 until the current date.
7. Click **OK**, and **Close** to exit and create the prompt.
8. Back in the **Query Builder**, select the **Filter Data** tab and drag and drop the Order_Date column from the left into the **Filter the raw data** section on the right.
9. In the New Filter window, verify that the selected operator is **In a range**. Use the pull-down arrow in the **Value** field; select the **Prompts** tab and then highlight &DateRange.

*The **In a Range** operator must be used with range prompts. It uses a special macro call that is generated by SAS Enterprise Guide and is seen in the filter window (in Figure 8). This %_eg_WhereParam macro calls a hidden program during the execution of the query code. It works by determining if a column's value is in the range specified at run time. The **Between** operator is used behind the scenes here but can't be specified with the range prompt name since it requires two values. However, the **Between** operator can be used with the macro variables that are created behind the scenes. We'll see an example of using these macro variables soon.*

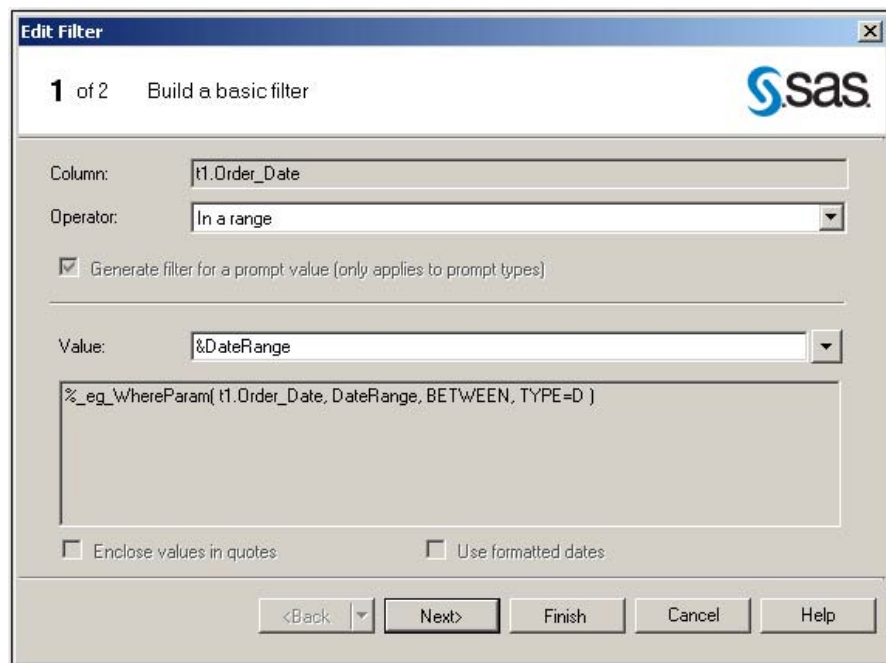


Figure 8. Edit Filter Window (In a Range)

10. Click **Finish** to close the filter window. Click **Run** to execute the query and **Yes** to replace the results from the previous run.
11. In the prompt window, select **Internet Sales** for the order type and enter **01JAN2006** in the **From:** field. Verify that **Today** is selected in the **To:** field. See Figure 9 for reference. The report will show the top 5 products by profit for Internet sales between 01JAN2006 and the current date.

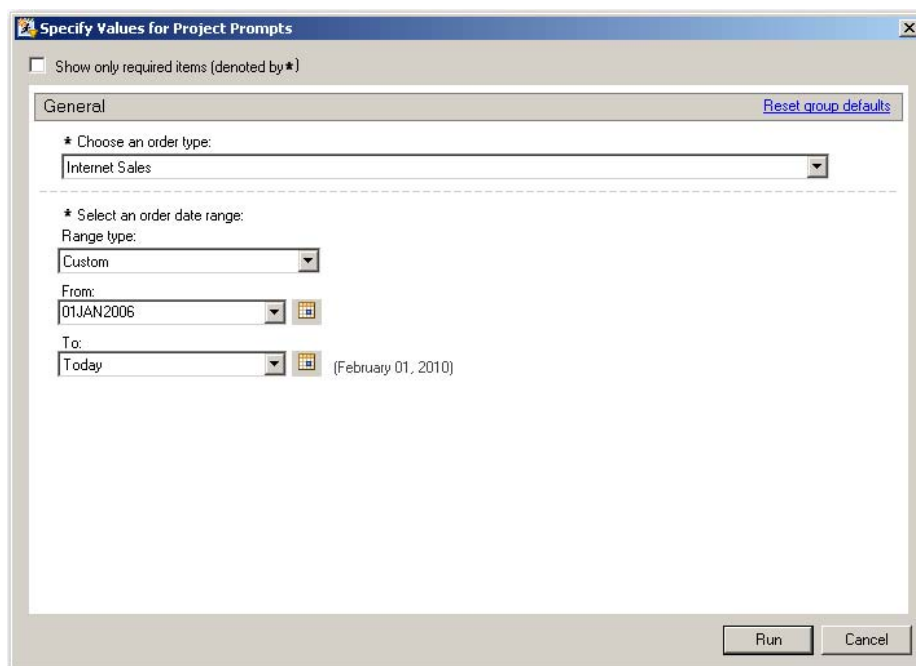


Figure 9. Prompt Window for Date Range

The report generated includes the correct information but doesn't have a very informative title since it doesn't include the date range information selected in the prompt. Range prompts work by creating multiple macro variables behind the scenes that represent the various prompt information entered. You can reference most range prompt information by using the prompt name followed by `_MIN` for the start range and `_MAX` for the end range.⁹ You can see the values of the macro variables created for the **Top 5 Products** query in the log by selecting the **Log** tab above the report and scrolling down to the `%LET` statements.¹⁰

```
%LET DateRange_max = 01Feb2010;
%LET DateRange_min = 01Jan2006;
```

Let's add the `DateRange_Min` and `DateRange_Max` macro variables to the title.

To-Do:

1. Select **Modify Task** to modify the **Top 5 Products** query.
2. Select the **Options** pull-down button and choose **Options for this query**.
3. Select **Titles** in the selection pane. Edit the title to read: **Top 5 Products based on Profit for %sysfunc(putn(&OrderType,ORDTYPFMT.)) between &DateRange_Min and &DateRange_Max**.
4. Click **OK** to close the window. Click **Run** to execute the query and **Yes** to replace the results from the previous run. Click **Run** in the prompt window to accept the previously entered prompt values and create the report. Select the **Results** tab to see the report shown in Figure 10.

⁹ Exceptions include date range prompts with an interval other than Day. Month, for example, would include macro variables representing the first and last day of the start month and macro variables representing the first and last day of the end month in the range.

¹⁰ The `DateRange_max` should be equal to the date when the report was created. The reports shown in this paper were generated on 01FEB2010.

Top 5 Products based on Profit for Internet Sales between 01Jan2006 and 01Feb2010

Product ID	SUM_of_Profit
Hi-fly Intrepid Stand 8 Black	\$343.80
Hi-fly Intimidator Ti R80/10	\$283.80
Jaguar 50-75 Liter Blue Women's Backpack	\$243.30
Force Micro Men's Suit	\$235.35
Shirt Termir	\$230.80

Figure 10. Report with Order Type and Date Range

By including range prompts here we've made our project even more interactive and functional. In addition, we made our report more informative by including in our title selected special macro variables that SAS Enterprise Guide uses behind the scenes to store prompt information. In the next section, we'll edit the OrderType prompt to allow users to select multiple values. In addition, we'll use some advanced macro techniques to include the multiple values in our ever-increasing report title.

MULTIPLE VALUE PROMPTS

Prompts can also allow for a user to select multiple values in the prompt window. Specifically, we'll edit the OrderType prompt to allow for multiple values and edit the query to accept this new prompt. That way, we'll be able to select multiple order types for the report. We'll also see how to take advantage of the code SAS Enterprise Guide generates to display these multiple values in the title.

To-Do:

1. Modify the **Top 5 Products** query by selecting **Modify Task** above the recently created report.
2. Select **Prompt Manager**. Highlight OrderType and click **Edit**. Click **Yes** in the Change Prompt warning window.¹¹ Select the **Prompt Type and Values** tab in the Edit Prompt window.
3. Change the **Number of Values** field to **Multiple values**. This will allow for multiple values to be selected in the prompt window. Click **OK** and **Close** to save the changes, and exit the **Prompt Manager**.
4. Before submitting the query, we must also edit the filter to allow for multiple values. Select the **Filter Data** tab in the **Query Builder** and double-click on the **t1.Order_Type = &OrderType** filter to edit it.
5. In the Edit Filter window, change the **Operator** to **In a list**. This is the appropriate operator for the possibility of multiple returned values from the prompt. Add **&OrderType** to the value field if it's not already there. Select the **Generate filter for a prompt value** check box. This is necessary in order for all selected prompt values to be used. See Figure 11 for reference.

¹¹ The Change Prompt warning window states that since the current version of the prompt is being used in a query any change to it could affect the results.

Figure 11. Edit Filter Window (Multiple Values)

Notice the following syntax in the filter window: **%_eg_WhereParam(t1.Order_Type, OrderType, IN, TYPE=N)**. This is the same macro call as before but with different parameters. It works by searching in the column *Order_Type* for the multiple values selected in the *OrderType* prompt. The *TYPE=N* argument refers to the type of the prompt value (numeric in this case). We can even use this same syntax in our own code to grab the selected prompt values. We'll see an example of this soon.

6. Click **Finish** to close the filter window and **Run** to execute the query. Click **Yes** to replace the results from the previous run.
7. Let's just look at non-retail sales. In the prompt window, highlight both **Internet Sales** and **Catalog Sales** in the list of available values and select the arrow to move them into the selected area. Leave the date range information the same. Click **Run** to generate the report. See Figure 12 below.

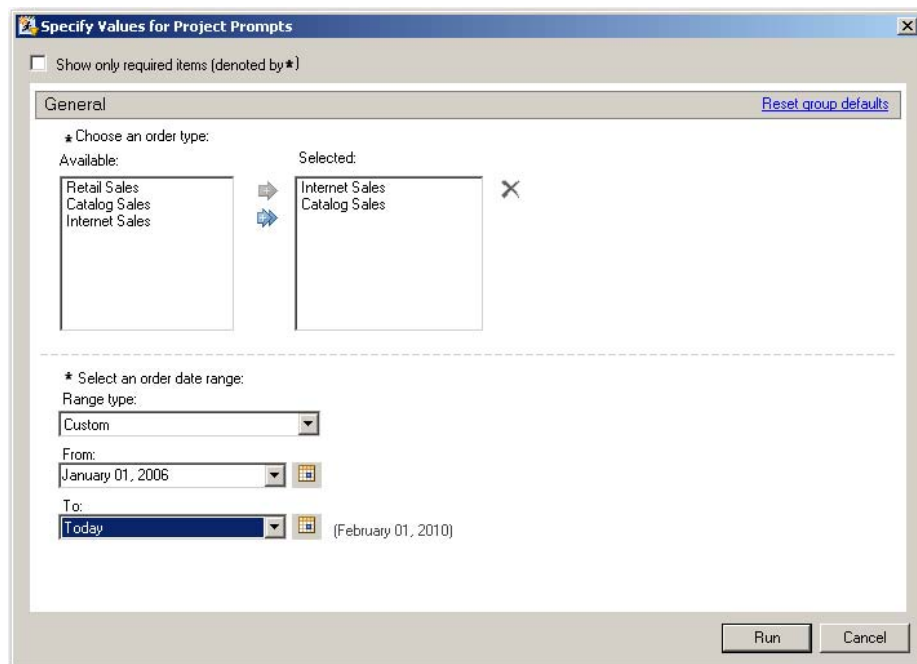


Figure 12. Multiple Value and Range Prompt Window

The results generated in the report now include both Internet and Catalog sales but the title only references the Internet Sales prompt value. This is because the OrderType prompt stores only the value of the first selection. Additional macro variables are created behind the scenes to store the additional values. These macro variables are resolved inside the special `%_EG_WhereParam` macro call we saw earlier. Programmers with some macro knowledge can edit the code generated from the **Query Builder** to create a macro with all the prompt selections to use in the report title. One possible method is PROC SQL with an INTO clause. Using the distinct option on a SELECT clause in SQL, we can create a list of all possible values of a column, such as Order_Type, and then use the `%_EG_WhereParam` macro call on a WHERE clause to pull out the values selected by the user in the prompt window. Adding an INTO clause we can create a single macro variable that lists every selected value separated by a delimiter, such as a comma or a space. We can even format the selected values by using a FORMAT= option on the SELECT clause. We can then resolve this macro variable anywhere, including the report title. Let's see an example.

To-Do:

1. Select the **Code** table above the recently generated report.
2. Attempt to edit and when prompted click **Yes** to create a modifiable copy of the code. Note that this new program is automatically named "Code for Top 5 Products".
3. Edit the LIBNAME statement(s) at the top of the code to read:


```
libname ORION "sas-data-library";
```
4. Add the following code at the top of the program directly below the LIBNAME statement:

```
PROC SQL noprint;
  SELECT DISTINCT ORDER_TYPE FORMAT=ORDTYPFMT. INTO :ORD_TYPES_SELECTED
  separated by ', '
  FROM ORION.orders AS t1
  WHERE %_eg_WhereParam( t1.Order_Type, OrderType, IN, TYPE=N );
```

*Notice much of this code is similar to the SELECT statement generated by the **Query Builder**. A NOPRINT option is added to the PROC SQL statement to prevent this code from generating output since we only need it to create a macro variable. The new SELECT clause is used here to grab the DISTINCT values of the ORDER_TYPE column, formatting them with the user defined format we created earlier. The WHERE clause includes the special `%_EG_WhereParam` (identical to the one in the query code), which selects only*

the order types selected by the user in the prompt window. The `INTO SEPARATED BY` clause then extracts the selected formatted order type values and puts them into a single macro variable, called `ORD_TYPES_SELECTED`, separated by a comma and a space. Once this code is submitted we can reference this macro anywhere, including a title.

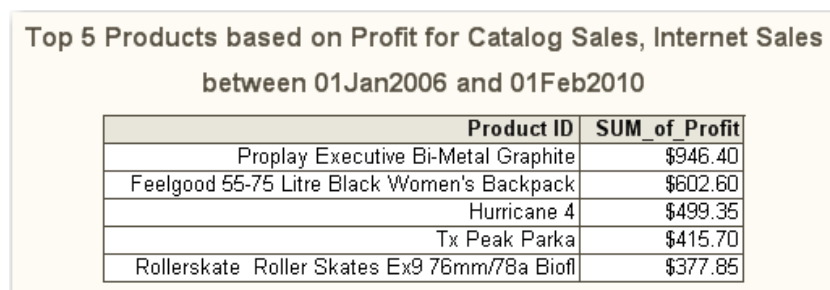
5. Edit the `TITLE1` statement in the code to read:

```
TITLE1 "Top 5 Products based on Profit for &ORD_TYPES_SELECTED";
```

6. Add a `TITLE2` statement directly below it that reads:

```
TITLE2 "between &DateRange_Min and &DateRange_Max";
```

7. To generate the report shown in Figure 13, click **Run** above the program and **Run** in the prompt window to accept the previously entered prompt values.



Product ID	SUM_of_Profit
Proplay Executive Bi-Metal Graphite	\$946.40
Feelgood 55-75 Litre Black Women's Backpack	\$602.60
Hurricane 4	\$499.35
Tx Peak Parka	\$415.70
Rollerskate Roller Skates Ex9 76mm/78a Biofl	\$377.85

Figure 13. Results of a Multiple and Range Value Query

We now have an interactive program with code generated by SAS Enterprise Guide and user enhancements that help to create a more flexible and informative report. We do, of course, have to submit the report in SAS Enterprise Guide if we want to be prompted for the values of our macro variables in our program. You probably noticed that although the `ORD_TYPES_SELECTED` macro variable we created with the `INTO` clause resolved to the correct values, we weren't prompted for any information for it. That's because the macro variable wasn't created in the **Prompt Manager** and added to the properties of the program. The other macro variables in the program were created in the **Prompt Manager** and got automatically added to the properties of our program when we used a copy of the **Query Builder** generated code as a starting point. Of course, we didn't need to enter any information for the `ORD_TYPES_SELECTED` macro variable since it got everything from the first SQL query—but there are times when we will want SAS Enterprise Guide to prompt us for macro variables we include in our code. We'll see an example of this in the next section.

ADDING RUN-TIME PROMPTS TO CODE

Even if you're a coding purist and wouldn't even consider using SAS Enterprise Guide to generate code you can still take advantage of prompts by adding them to your own code. Running your SAS macro programs inside SAS Enterprise Guide gives you the option of assigning values to macro variables using a prompt window—just like with tasks and queries. The trick is to create the prompts in the **Prompt Manager** and then assign these prompts in the properties of the code within SAS Enterprise Guide. We'll see an example of adding some pre-written code to our current project. We will use the `Products` table to generate a user-defined format to display product names rather than `Product_ID` numbers. Applying this user-defined format to the `Product_ID` column in our query will eliminate the need to join the two tables. Although this example would work without using macro variables and prompts, doing so will create a dynamic program that will allow us to generate formats from any set of variables from a data set. Let's take a look.

To-Do:

1. Select **File** ⇒ **New** ⇒ **Program** to open a new node into the current project. Include the following code:

```
libname LIBRARY "&location";
```



```

data fmts/view=fmts;
    set LIBRARY.&data (rename=(&start=start &label=label));
    retain fmtname "&fmtname" type "&type" ;
run;

proc format cntlin=fmts library=sasuser;
run;

proc sql;
    drop view fmts;
run;

```

This program creates an input control lookup table¹² that is read into a **PROC FORMAT** to construct a user-defined format. More information about this method can be found in the SAS Online Documentation.¹³ We're using macro references here to allow a user complete control of this process. Before running the program, a user must give values to the following macro variables: *Data* (the name of the data set that contains the variables needed to create the format), *Location* (the location of the data), *Start* (the name of the variable that contains the values we need to replace), *Label* (the name of the variable that contains the labels we want to replace the start values with), *fmtname* (the name of the user-defined format), and *Type* (the type of the format that corresponds to the type of the variable we plan to apply this to). Inside SAS Enterprise Guide, we can create prompts to give values to these macro variables. To do so, these prompts have to be created in the **Prompt Manager** and assigned to the properties of that program. Let's see how:

- After including the code in the programming window, right-click on the program name in the project tree and select **Rename**. Type *Create Format from Data*.
- Select **Properties** above the program in the workspace area to create the prompts.
- In the Properties for *Create Format from Data* window, select **Prompts** in the selection pane. Select **Prompt Manager**, and then select **Add** in the **Prompt Manager** to create the first prompt.
- In the Add New Prompt window under the **General** tab, name your prompt *Location* to correspond with the *Location* macro variable in the *Create Format from Data* program. Type *Where is the table located?* in the **Displayed Text** field. Do **not** select the **Requires a non-blank value** check box. This will be important in a later example.
- Select the **Prompt Types and Values** tab to verify that **Text** is selected as the **Prompt type** and **User enters values** is the **Method for populating prompt**. The **Number of values:** should be set to: **Single value**. Click **OK** to exit and create the prompt.
- Repeat this process to create the same type of text entry prompts¹⁴ for *Data*, *Start*, *Label*, and *FmtName* using the information in the table below. Remember: Do **not** select the **Requires a non-blank value** check box for any of these prompts. You shouldn't have to make changes to the **Prompt Type and Values** tab; it should look just like it did for the *Location* prompt.

Prompt Name	Displayed Text
Data	<i>Enter input control data table name:</i>
Start	<i>Enter Start values variable name:</i>
Label	<i>Enter Label values variable name:</i>
FmtName	<i>Enter name of format:</i>

¹² A view is used here to avoid creating an intermediate data set. The view is then deleted at the end of the program since it is no longer needed in the project.

¹³ <http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/a002473477.htm#a002473479>

¹⁴ Default values and lists could be used for these prompts if desired but text prompts are used here to simplify the directions.

8. Create a final prompt for Type. Name this prompt Type and enter *What's the format type?* in the displayed text field and do **not** select the **Requires a non-blank value** check box.

Since the only possible values are "C" for Character and "N" for Numeric let's set this prompt up as a list allowing a user to select a value.

9. In the **Prompt Types and Values** tab, select **Text** as the **Prompt type**, **User selects values from a static list** as the **Method for populating prompt**, and allow for a **Single value**.
10. Select **Add** to create the first selection. Type **C** in the **Unformatted Value** field and **Character format** in the **Formatted (Displayed) Value** field.

*By giving a formatted displayed value here, a user can select **Character format** in the prompt window to populate the Type macro variable with the value: **C**. We can similarly do this with the numeric selection.*

11. Select **Add** again to create the second selection. Type **N** in the **Unformatted Value** field and **Numeric format** in the **Formatted (Displayed) Value** field. See Figure 14 for reference. Click **OK** to finish and create the prompt. Close the Prompt Manager window.

The screenshot shows the 'Add New Prompt' dialog box with the following configuration:

- General** tab selected.
- Prompt type:** Text
- Method for populating prompt:** User selects values from a static list
- Number of values:** Single value
- Minimum length:** (empty)
- Maximum length:** (empty)
- Include Special Values:**
 - All possible values
 - Missing values
- Append formatted values with unformatted values
- List of values:**

Unformatted Value	Formatted (Displayed) Value	Default
C	Character Format	<input type="radio"/>
N	Numeric Format	<input type="radio"/>
- Allow user to specify additional (unformatted) values

Buttons on the right side of the dialog include: Add, Get Values..., Delete, Clear Default, Move up, and Move down. At the bottom are OK, Cancel, and Help buttons.

Figure 14. Add New Prompt Window (Text)

We now have created and defined a prompt for each macro reference in the program. However, in order for these prompts to pop-up when the program is executed we need to add them to the properties of the program.

12. Select **Add**. Highlight all the prompts you just created and click **OK** to add them to the properties of the *Create Format from Data* program. Use the UP/DOWN arrows to the right of the field to order these prompts in the same order shown in Figure 15 so that the user will be prompted in that same order.

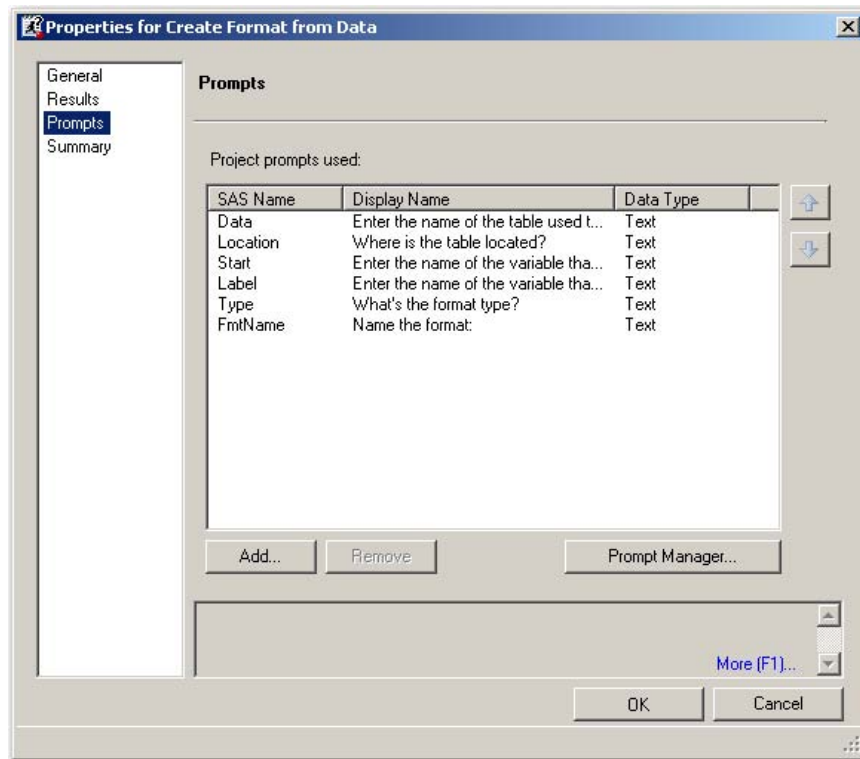


Figure 15. Properties for Create Format from Data Program

13. Click **OK** to exit the Properties window. Click **Run** above the program in the workspace area to submit the code.
14. In the Specify Values for Prompt Values window, enter the information¹⁵ shown in Figure 16.
15. Click **Run** to submit the code with the entered prompt values.

¹⁵ The location you stored the Products table in should be entered in the field for the second (Location) prompt.

Figure 16. Specify Values for Project Prompts Window

16. Check the log for the following message:

NOTE: Format PRODNAM has been written to SASUSER.FORMATS.

*We have now created a user-defined format that we can apply to the Product_ID column in our query as a look-up tool to grab product names. This avoids the more resource intensive joining of Orders with Products. Since we're now working with the edited generated code we can make changes to it instead of making modifications in the **Query Builder**. If we apply the Prodname format we just created to the Product_ID variable we can remove Product_Name from the query and thus eliminate the need for the join.*

17. Double-click on the *Code for Top 5 Products* icon. Edit the last **PROC SQL** step to look like the one shown below.

```
PROC SQL OUTOBS=5;
  SELECT Product_ID format=prodname.,
         /* SUM_of_Profit */
         (SUM(t1.Profit)) FORMAT=DOLLAR10.2 AS SUM_of_Profit
  FROM ORION.orders AS t1
  WHERE %_eg_WhereParam( t1.Order_Type, OrderType, IN, TYPE=N ) AND
         %_eg_WhereParam( t1.Order_Date, DateRange, BETWEEN, TYPE=D )
  GROUP BY Product_ID
  ORDER BY SUM_of_Profit DESCENDING;
QUIT;
```

18. Click **Run** to submit the code and enter the same values in the prompt window shown in Figure 12. We now have more efficiently created the same report shown in Figure 13.

By adding prompts to the user-generated code we were able to create a dynamic, interactive, and portable program that lets us create formats from a set of various parameters. By adding the *Prodname* format to the query code we created a more efficient program to achieve our results. Since we've placed this format in a permanent location we don't need to rerun the *Create Format from Data* code every time we need our report. We might, however, need to resubmit this code as new items get added to the product inventory. To prevent this code from executing each time

we run our project, we can place a conditional prompt that allows a user to select whether they want the program to execute. We'll talk more about conditional processing prompts in the next section.

CONDITIONAL PROCESSING PROMPTS

Conditional processing in SAS Enterprise Guide 4.2 allows a user control over whether tasks, queries, programs, and even e-mails and export results, execute when a project or process flow is run. Using IF-THEN/ELSE logic, conditions are created based on such factors as input data set values, time and date, and prompts. By including a conditional processing prompt in our project, we can control whether our *Create Format from Data* program is submitted.

Before we add the conditional processing prompt, let's first link the *Create Format from Data* program to our query code to ensure it executes first when we run the process flow. We can also clean up our project by removing the other unnecessary items.

To-Do:

1. In the Process Flow window, right-click on the *Create Format from Data* program icon and select **Link Create Format from Data to...** Select *Code for Top 5 Products* in the Link window and click **OK**.
2. Click and drag the mouse over the remaining items in your project, and select **Delete**¹⁶ to remove them. Click **Yes** to confirm, leaving the items shown in Figure 17.

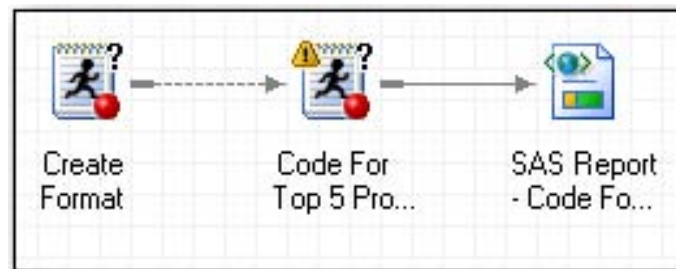


Figure 17. Process Flow Window

We can now add a condition to the *Create Format from Data* program that allows a user control over whether the program is processed by clicking **Yes** or **No** in a prompt window.

To-Do:

1. Right-click on the *Create Format from Data* code icon, and select **Condition** ⇒ **Add**. In the Conditional Processing window, select **Add** again.
2. Select **Prompts** in the **Based on:** field in the Add a Condition window to create a condition based on a prompt. Select **New** to open the Prompt Manager window.
3. Name your **Prompt** RunFmt and enter *Run the Create Format from Data program?* in the **Displayed text** field. Select the **Requires a non-blank value** check box.
4. In the **Prompt Types and Values** tab, select **Text** as the **Prompt type**, User **selects values from a static list** as the **Method for populating prompt**. Allow for a **Single value**.
5. Select **Add** to create the first selection. Type Y in the **Unformatted Value** field and Yes in the **Formatted (Displayed) Value** field.

¹⁶ If you want to keep the items for any reason, you can move them into a new process flow within the project by creating the process flow first and then right-clicking and selecting **Move to** → instead of deleting.

6. Select **Add** again to create the second selection. Type *N* in the **Unformatted Value** field and *No* in the **Formatted (Displayed) Value** field. Make *No* the default selection. Click **OK** to finish and create the prompt.
7. Ensure that the new prompt is now listed in the **Prompt:** field. Select **Equal to** in the **Operator:** field and enter *Y* in the **Value:** field.
Meaning, a user has to enter Yes (Y) when prompted in order to run Create Format from Data before the Query code.
8. Select **Add**. In the Conditional Processing window, select *Code for Top 5 Products* in the pull-down list for **Else run this task:**.
Since the Code for Top 5 Products will run, by default, only after the Create Format from Data program does, this will ensure it runs regardless of the value you select for the conditional prompt.
9. Click **OK**. See Figure 18 for reference.

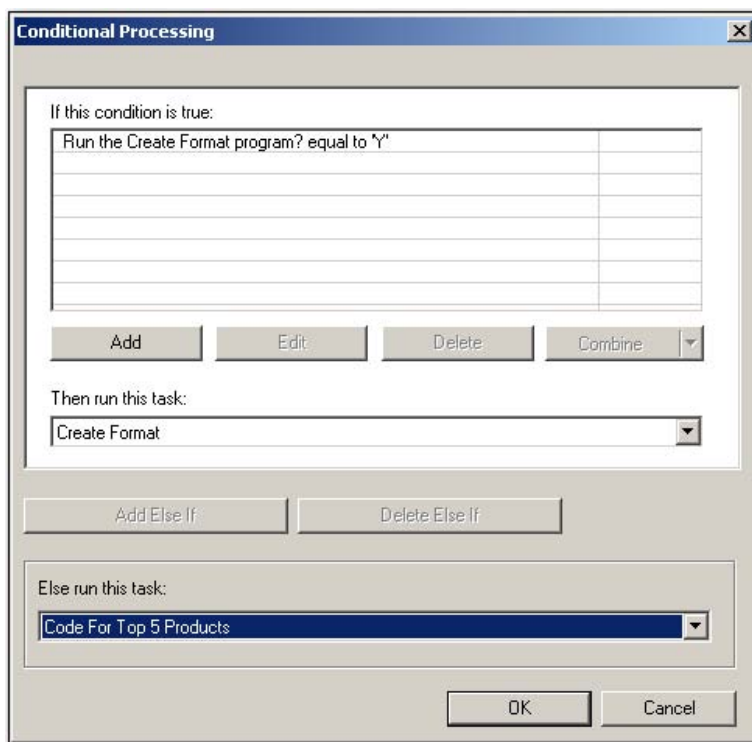


Figure 18. Conditional Processing Window

10. Run your project by selecting **Run** ⇒ **Run Process Flow** in the workspace area.
11. In the prompt window, select **No** for the first prompt.
Since we already have the format created, we don't need to update it. And since we're not running the Create Format from Data program, we don't need to worry about the prompt values for it.
12. Select the **Show only required items (denoted by *)** box at the top of the prompt window to remove any prompts that do not require a value.¹⁷
13. After removing the unnecessary prompt fields, enter the same values for the remaining prompts that were entered earlier. See Figure 19.

¹⁷ This is why we didn't check the **Requires a non-blank value** check box when creating the prompts for the *Create Format* program. If we had, values would have been required for each prompt field even though we weren't running the program.

14. Click **Run**.

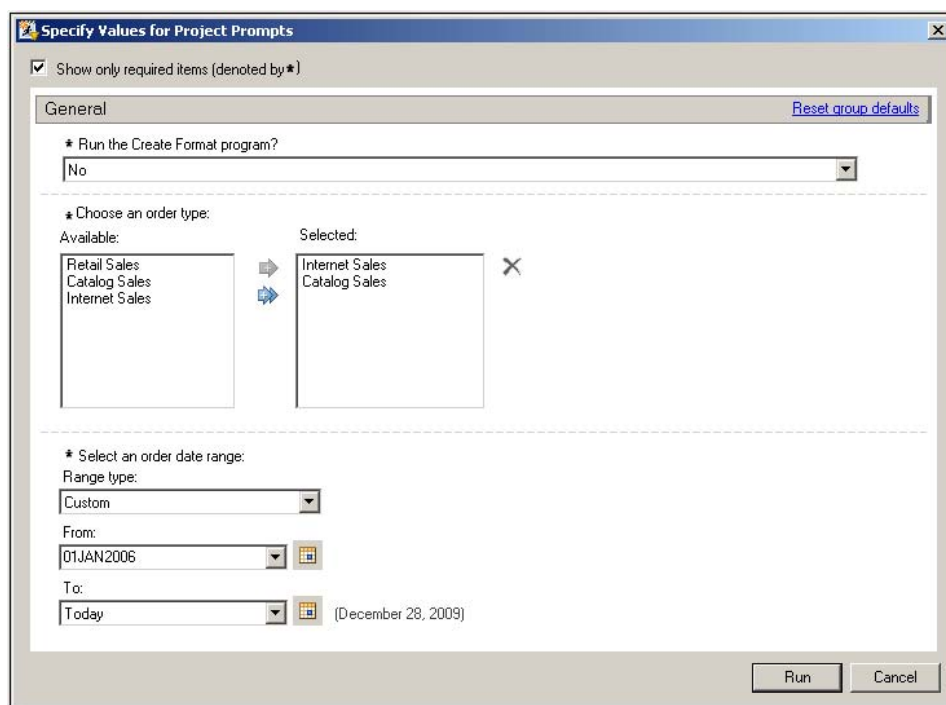


Figure 19. Prompt Window with Conditional Prompt

By adding the conditional processing prompt to our scenario we were able to prevent any unnecessary submission of steps in our project, making it more interactive and efficient. Although programmers can add macro conditional statements to their own code to control when and if steps execute based on various factors, a user controlled prompt window controlling the execution of code is unique to SAS Enterprise Guide.

CONCLUSION

Throughout this paper, we've developed a SAS Enterprise Guide 4.2 project producing an example report. By adding user-controlled prompts and various SAS macro programming techniques to our code we created a concise, dynamic, interactive, and efficient project to produce the desired results.

The examples we've seen included: the use of both single and range prompts in tasks and queries, ways to include the values of the user-entered information in things like titles and footnotes, adding prompts to a user's own code, and how to apply conditional processing prompts to parts of a project. In doing so, we were able to see how using prompts and macro language cannot only be useful for SAS Enterprise Guide users in their tasks and queries, but also for programmers developing and executing code in a SAS Enterprise Guide environment.

As you explore the Prompt Manager in SAS Enterprise Guide 4.2 you might see references to "dynamic" and "data dependent" prompts. Although I attempted to demonstrate how to add your own macro code to create more dynamic and dependent programs and projects, the use of these specific techniques when creating prompts require stored processes using SAS BI software. Because this paper was directed to beginner SAS Enterprise Guide users and/or Base SAS programmers, I decided to leave that subject for a later topic. Hopefully, this paper has encouraged you to add some of these techniques to your own work, as well as come up with some of your own.

ACKNOWLEDGMENTS

The author appreciates the feedback and suggestions provided by Stacey Syphus.

CONTACT INFORMATION

Your feedback is valued and encouraged. If you'd like to contact the author with comments or questions, or for a copy of the data used in this paper, please e-mail him at: Kenneth.Sucher@sas.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.