SAS® Macro Language 1: Essentials

Appendix A

Case Study

SAS® Macro Language 1: Essentials Case Study was developed by Stacey Syphus, Mark Jordan, and Kathy Passarella . Additional contributions were made by Brittany Coleman, Bruce Dawless, Davetta Dunlap, Brian Gayle, Gina Repole, and Theresa Stemler. Instructional design, editing, and production support was provided by the Learning Design and Development team.

SAS® Macro Language 1: Essentials Case Study

Book code E715851, course code LWMC1V2/MC1V2, prepared date 11Dec2019. LWMC1V2_001

ISBN 978-1-64295-754-9

Appendix A Case Study

A.1 Case Study Introduction

In this case study, you solve a real-world data problem by applying concepts that you learned in the SAS Macro 1: Essentials course. Be aware that there are numerous solutions to this problem, and some can include concepts that are outside the scope of the course.

Case Study Files

The files used in this case study are included in the SAS Macro 1: Essentials course data. It is assumed that the course data was previously set up in your SAS environment. Be sure to submit the **libname.sas** program in the course files to define the **Path** macro variable and the **MC1** SAS library.

The case study uses the following files:

From the **case_study** folder:

- **CaseStudyStart.sas** SAS program
- **ReplaceSpace_Solution.sas** SAS program
- **SupplierReport_Solution.sas** SAS program

From the **data** folder:

- **orders** table
- **products** table
- **country_codes** table

Note: Remember that you can use **&path** to reference the location of your files as you write the case study program.

How to Attempt the Case Study

There are three ways to complete this case study. Feel free to follow the method that fits your skill set.

Advanced Level

- If you feel comfortable with the topics in the SAS Macro 1: Essentials course and want to treat this as a real-world problem, read Section A.2, “Advanced Level,” of the PDF and begin. Only project requirements are defined, and you must determine the SAS code necessary to complete each task. During the process, feel free to use your course notes, SAS documentation, or other online resources.

Intermediate Level

- If you think you might need a bit of assistance in the case study, read Section A.3, “Intermediate Level,” of the PDF for a guide. The guide does not give you the solutions, but rather a roadmap for how to solve the problem, including specific suggestions for the SAS macro language. If you are stuck on a specific task, you can find the solution in Section A.4, “Beginner Level.”

Beginner Level

- If you are not familiar with the macro language and want to use the case study as a walk-through demo, feel free to do so. You can read Section A.4, “Beginner Level,” and follow a roadmap to solve the problem with a suggested solution for each task. We recommend that after you run through the case study as a demo, to go back and attempt it on your own.

If you have any questions regarding the case study, if you complete the case study and come up with different solutions and want to show them off, or if you want to post additional visualizations or analysis of the data, create a post in the SAS Training Forum. We would love to hear from you!

Business Problem

Orion Star company maintains detailed tables about the products that they offer, the orders placed by customers, and the suppliers for each product. A SAS program has been written to prepare the data to summarize profit for each supplier. The first portion of the program identifies the top suppliers. The second part of the program must be edited to generate a summary report for a particular supplier. Information about the selected supplier, including the ID, name, and country, must be manually replaced in the program.

The program should be modified to automatically generate a summary report for the top five suppliers for a selected method of order type (1=Retail, 2=Catalog, 3=Internet).

Deliverables

There are two deliverables that need to be completed by the end of the case study:

- **replacespace.sas** – A macro program that replaces all spaces in a string with an underscore. The macro should be stored and retrieved using the autocall facility.
- **supplierreport.sas** – A macro program that identifies the top *n* suppliers for either all or a selected order type and generates a summary PDF report for each separate supplier. The macro should automatically create the reports with the only input required being the selected order type and the number of suppliers to include. The macro should be stored and retrieved using the autocall facility.

Program Information

Part 1 of the **CaseStudyStart.sas** program does the following:

- creates a temporary table named **OrderDetail** that joins all rows from the **orders** table with the corresponding information from the **product_dim** table. Profit is calculated for each order where **Order_Type**=*1* (retail sales).
- summarizes **Profit** and identifies **Supplier_ID**, **Supplier_Name**, and **Supplier_Country** for each unique supplier. Results are ordered by descending values of **Profit**.

The **country_codes** lookup table is printed so that you can look up the corresponding country name for the top supplier **Supplier_Country**.

Part 2 of the program creates a PDF file for **Supplier_ID**=*1303*, the top supplier identified by running part 1 of the program. The report includes a bar chart that represents the sum of profit for each product category, and a table with the total and average profit for each product group.

Data Layout

The first PROC SQL step in the **CaseStudyStart.sas** program creates a table named **OrderDetail**. Below is the data layout for that table. The layout lists the column name and a description.

OrderDetail table

Column	Description
Order_ID	Unique ID for each order placed.
Product_ID	Unique ID for each product. **Product_ID** is used to join orders and **product_dim** tables.
Order_Type	Method for placing order: 1 = Retail Sale 2 = Catalog Sale 3 = Internet Sale
Product_Category	12 unique product category values: • Assorted Sports Articles • Children Sports • Clothes • Golf • Indoor Sports • Outdoors • Racket Sports • Running - Jogging • Shoes • Swim Sports • Team Sports • Winter Sports
Product_Group	57 unique product group values. Here are some examples: • A-Team, Kids • American Football • Anoraks&Parkas • Assorted Sports articles • Backpacks • Badminton • Baseball • Basket Ball • Bathing Suits • Bathing Suits, Kids
Product_Line	Four unique product line values: • Children • Clothes & Shoes • Outdoors • Sports

Column	Description
Product_Name	Unique product name.
Profit	Calculated as: (**total_retail_price** - **costprice_per_unit**) / **quantity**
Supplier_ID	Unique ID for each supplier.
Supplier_Name	Unique name for each supplier.
Supplier_Country	Two-letter country code for each supplier country.

A.2 Advanced Level

In the advanced version of the case study, you receive the high-level requirements to solve the business problem. There are multiple solutions to the problem, and how you solve it is your choice.

To solve the business problem, you must follow the requirements below given to you by your supervisor.

Deliverables and Requirements:

Your job is to familiarize yourself with the **CaseStudyStart.sas** program and identify what must be edited in the program to identify the top five suppliers and create a separate PDF report for each supplier.

1. Read the comments in the **CaseStudyStart.sas** program to get familiar with the code and the edits required to generate a report for a different supplier and subset of **Order_Type**.
2. Create a macro function named **%REPLACESPACE** that uses the TRANWRD function to replace all spaces in a string with underscores. Save the **replacespace.sas** macro program in the **autocall** folder and enable the autocall facility to read it.
3. Modify the code in the **CaseStudyStart.sas** program to build a macro named **%SupplierReport** with a parameter to select a particular **Order_Type** value.
4. Validate the **ot** parameter value to ensure that it is either *1*, *2*, or *3*. If no value is provided, write a custom message to the log. The message should indicate that a value is required and that the program will stop executing. It should also include a list of valid values. If a value other than *1*, *2*, or *3* is provided, write a custom error message to the log that prints a list of valid values and stops processing the rest of the program.

   Note: Be sure to use the MINOPERATOR option in the %MACRO statement to enable the macro IN operator.
5. If a value of *1*, *2*, or *3* is provided for the parameter, subset the **OrderDetail** table based on **Order_Type**.

   Modify the WHERE statement in the first PROC SQL step to include rows where **Order_Type** is equal to the **ot** macro variable.
6. Create a series of macro variables that will store **Supplier_ID**, **Supplier_Name**, **Supplier_Country**, and **Profit** for each of the top five suppliers. For example, the macro variables **TopSupp1**, **Name1**, **Country1**, and **Profit1** will store information about the top supplier; **TopSupp2**, **Name2**, **Country2**, and **Profit2** will store information about the second-ranked supplier; and so on.
7. Create a series of macro variables named **Country_*CC*** where ***CC*** is the two-letter **CountryCode** value read from the **work.country_codes** table. Assign the corresponding **CountryName** value to each macro variable.
8. Use a macro DO loop to repeat Part 2 of the program five times. The first time through the loop, the program should generate the PDF report for the top supplier. The report should be modified as follows:
   a. The prefix for each PDF file name should be the supplier rank number, 1 through 5. The name of each PDF file should be the value of **Supplier_Name** with all spaces replaced with underscores. Use the **REPLACESPACE** custom macro function.

   b. The first title should be the rank of the supplier and then the **Supplier_Name** value, followed by the full country name for that particular supplier (for example, *Orders for #1 Eclipse Inc, United States*).
   c. The second title should be one of the following, depending on the value of the **ot** parameter: *Retail Sales Only*, *Catalog Sales Only*, or *Internet Sales Only*.
   d. For the bar chart (PROC SGPLOT step), the data should be subset to include one supplier at a time.
   e. For the report (PROC SQL step), a footnote should be added below the report that includes the date and time that the report was created. The data should also be subset to include only the top supplier.
9. Test the **%SupplierReport** macro program with parameter values of *1*, *2*, *3*, *4*, and null. Make sure that when *4* is provided, the error message is written to the log and no output is generated. Test that error messages are also generated if a null value is provided.
10. Save the **supplierreport.sas** macro program in the **autocall** folder.

Answer the Following Business Questions:

Using the reports created by the macro program, answer the following business questions.

1. Which company is the #2 supplier for internet sales only (**Order_Type**=*3*)?

2. What was the top product group for the top supplier for catalog sales only (**Order_Type**=*2*)?

3. Which country is the #4 supplier for retail sales only (**Order_Type**=*1*)?

A.3 Intermediate Level

In the intermediate version of the case study, you receive the high-level requirements and specific suggestions to solve the business problem. If you are stuck, refer to Section A.4**, "**Beginner Level," in this document for solutions to the specific task, or post a question in the SAS Training Community.

Your job is to familiarize yourself with the **CaseStudyStart.sas** program and identify what must be edited in the program to identify the top five suppliers and create a separate PDF report for each supplier.

1. Read the comments in the **CaseStudyStart.sas** program to get familiar with the code and the edits required to generate a report for a different supplier and subset of **Order_Type**.
2. Create a macro function named **%REPLACESPACE** that uses the TRANWRD function to replace all spaces in a string with underscores. Save the **replacespace.sas** macro program in the **autocall** folder and enable the autocall facility to read it.
   a. Create a new program. Start a macro definition named **replacespace** with a single positional parameter named **Text**.
   b. Use the %SYSFUNC function to execute the TRANWRD function.
   c. The first argument is the value of the **Text** parameter. The second argument is the target character, which is a space. The third argument is the replacement character, which is an underscore.

      Note: Use appropriate macro quoting functions (if necessary) to the arguments.

   d. Close the macro definition.
   e. Save the program as **replacespace.sas** in the **autocall** folder.
   f. In the **CaseStudyStart.sas** program, add an OPTIONS statement to indicate that the SASAUTOS search path should include the **autocall** folder and the **SASAUTOS** library.
3. Modify the code in the **CaseStudyStart.sas** program to build a macro named **%SupplierReport** with a parameter to select a particular **Order_Type** value.
   a. At the top of the program, start a macro definition named **SupplierReport** with **ot** as a positional parameter.
   b. At the bottom of the program, end the macro definition with a %MEND statement.
4. Validate the **ot** parameter value to ensure that it is either *1*, *2*, or *3*. If no value is provided, write a custom message to the log. The message should indicate that a value is required and that the program will stop executing. It should also include a list of valid values. If a value other than *1*, *2*, or *3* is provided, write a custom error message to the log that prints a list of valid values and stops processing the rest of the program

   Note: Be sure to use the MINOPERATOR option in the %MACRO statement to enable the macro IN operator.

   a. Add the MINOPERATOR option in the %MACRO statement.
   b. Use %IF and %END statements to test whether the parameter is equal to a null value. If it is, use %PUT statements to write a custom error message to the log that indicates that a value is required and that the program will stop executing. It should also include a list of valid values. The error message could appear as follows:

```
ERROR: You did not specify an Order_Type code (required).
       Valid Order_Type values include blank, 1 (retail), 2 (catalog), or 3 (internet).
       Program will stop executing
```

c. Use the %RETURN statement to stop execution and %END to close the %IF block.

d. In the %MACRO statement, add the /MINOPERATOR option to be able to use the IN operator in a macro statement.

e. Use %ELSE, %IF, and %END statements to test whether the parameter is not in the list of *1*, *2*, or *3*. Write a custom error message to the log if an invalid value is provided. The program should also stop executing. The error message could appear as follows:

```
ERROR: Valid Order_Type values include blank, 1 (retail), 2 (catalog), or 3 (internet).
       Program will stop executing.
```

f. Use the %RETURN statement to stop execution and %END to close the %IF block.

g. Use %ELSE, %DO, and %END statements to indicate whether the parameter value is valid, and then the rest of the program should run.

5. If a value of *1*, *2*, or *3* is provided for the parameter, subset the **OrderDetail** table based on **Order_Type**.

   Modify the WHERE statement in the first PROC SQL step to include rows where **Order_Type** is equal to the **ot** macro variable.

6. Create a series of macro variables that will store **Supplier_ID**, **Supplier_Name**, **Supplier_Country**, and **Profit** for each of the top five suppliers. For example, the macro variables **TopSupp1**, **Name1**, **Country1**, and **Profit1** will store information about the top supplier; **TopSupp2**, **Name2**, **Country2**, and **Profit2** will store information about the second-ranked supplier; and so on.

   a. Find the second PROC SQL step that identifies the top five suppliers.

   b. Add an INTO clause to create the following series of macro variables for the top five suppliers:

      1) **TopSupp1**-**TopSupp5** to store the **Supplier_ID** values.

      2) **Name1**-**Name5** to store the **Supplier_Name** values.

      3) **Country1**-**Country5** to store the **Supplier_Country** values.

      4) **Profit1**-**Profit5** to store the sum of **Profit** values.

7. Create a series of macro variables named **Country_*CC*** where ***CC*** is the two-letter **CountryCode** value read from the **work.country_codes** table. Assign the corresponding **CountryName** value to each macro variable.

   a. Write a DATA step that reads the **work.country_codes** table.

   b. Use CALL SYMPUTX to create the series of macro variables. The first argument should concatenate *Country_* with the value of **CountryCode** to create the macro variable names. The second argument should assign the value from the **CountryName** column.

8. Use a macro DO loop to repeat Part 2 of the program five times. The first time through the loop, the program should generate the PDF report for the top supplier. The report should be modified as follows:

   a. The prefix for each PDF file name should be the supplier rank number, 1 through 5. The name of each PDF file should be the value of **Supplier_Name** with all spaces replaced with underscores. Use the **REPLACESPACE** custom macro function.

1) After the ODS GRAPHICS statement, add a %DO macro statement with an index variable **i** that starts at 1 and ends at 5.
2) At the end of the program, before the %END statement (this closes the %IF %THEN/%DO block), add another %END statement.
3) After the %MACRO statement, add a %LOCAL statement to ensure that **i** is written to and read from the local symbol table.
4) In the ODS PDF statement, delete the hardcoded supplier name, *1_Eclipse_Inc* (keep the .pdf extension) and replace it with an expression that does the following:
   a) calls the **%replacespace** macro
   b) includes the value of the macro variable **i** followed by an underscore as the file name prefix.
   c) uses an indirect macro variable reference as the parameter for the **%replacespace** macro. The indirect reference should substitute the value of the **Name1**, **Name2** (and so on) macro variable.

b. The first title should be the rank of the supplier and then the **Supplier_Name** value, followed by the full country name for that particular supplier (for example, *Orders for #1 Eclipse Inc, United States*).
   1) After the ODS statement, add a %LET statement to create a macro variable named **CC** that will be the two-letter **CountryCode** for the supplier being analyzed in the loop. (For example, when **i=1**, the value of **CC** is the **CountryCode** assigned to the **Country1** macro variable.) This requires an indirect macro variable reference. This macro variable is used later to insert the country name in the title.
   2) In the TITLE statement, use the **i** macro variable to substitute the rank number of the supplier.
   3) Use an indirect macro variable reference to substitute the macro variable value for **Name1**, **Name2**, and so on.
   4) Use an indirect macro variable reference to substitute the full country name. Remember that the macro variable **Country_*CC*,** where ***CC*** is the two-letter **CountryCode** for the supplier, stores the country name. Use the **CC** macro variable created earlier as part of the indirect reference.

c. The second title should be one of the following, depending on the value of the **ot** parameter: *Retail Sales Only*, *Catalog Sales Only*, or *Internet Sales Only*.

   To create the second title, use %IF, %THEN, and %END statements to provide unique TITLE2 statements depending on the value of the **ot** parameter.

d. For the bar chart (PROC SGPLOT step), the data should be subset to include one supplier at a time.

   Modify the WHERE statement to use an indirect macro variable reference to substitute the **Supplier_ID** value.

e. For the report (PROC SQL step), a footnote should be added below the report that includes the date and time that the report was created. The data should also be subset to include only the top supplier.
   1) Add a FOOTNOTE statement before the last PROC SQL step.
   2) Use %SYSFUNC to execute the TODAY() function and format it with an appropriate date format.

3) Use %SYSFUNC again to execute the TIME() function and format it with an appropriate time format.

9. Test the **%SupplierReport** macro program with parameter values of *1*, *2*, *3*, *4*, and null. Make sure that when *4* is provided, the error message is written to the log and no output is generated. Test that error messages are also generated if a null value is provided.
10. Save the **supplierreport.sas** macro program in the **autocall** folder.

Answer the Following Business Questions:

Using the reports created by the macro program, answer the following business questions.

1. Which company is the #2 supplier for internet sales only (**Order_Type**=3)?

2. What was the top product group for the top supplier for catalog sales only (**Order_Type**=2)?

3. Which country is the #4 supplier for retail sales only (**Order_Type**=1)?

A.4 Beginner Level

This section is a step-by-step guide for solving the case study, including solutions for each task. The steps are the same steps from Section A.3, “Intermediate Level,” but here the solution code for each step is available.

For more information, you can visit the SAS Documentation or post a question in the SAS Training Community.

Your job is to familiarize yourself with the **CaseStudyStart.sas** program and identify what must be edited in the program to identify the top five suppliers and create a separate PDF report for each supplier.

1. Read the comments in the **CaseStudyStart.sas** program to get familiar with the code and the edits required to generate a report for a different supplier and subset of **Order_Type**.
2. Create a macro function named **%REPLACESPACE** that uses the TRANWRD function to replace all spaces in a string with underscores. Save the **replacespace.sas** macro program in the **autocall** folder and enable the autocall facility to read it.
   a. Create a new program. Start a macro definition named **replacespace** with a single positional parameter named **Text**.
   b. Use the %SYSFUNC function to execute the TRANWRD function.
   c. The first argument is the value of the **Text** parameter. The second argument is the target character, which is a space. The third argument is the replacement character, which is an underscore.

      Note: Use appropriate macro quoting functions (if necessary) to the arguments.

   d. Close the macro definition.

```
%macro replacespace(text);
    %sysfunc(tranwrd(&text,%str( ),_))
%mend replacespace;
```

   e. Save the program as **replacespace.sas** in the **autocall** folder.
   f. In the **CaseStudyStart.sas** program, add an OPTIONS statement to indicate that the SASAUTOS search path should include the **autocall** folder and the **SASAUTOS** library.

```
options sasautos=("&path", SASAUTOS);
```

3. Modify the code in the **CaseStudyStart.sas** program to build a macro named **%SupplierReport** with a parameter to select a particular **Order_Type** value.
   a. At the top of the program, start a macro definition named **SupplierReport** with **ot** as a positional parameter.

```
%macro supplierreport(ot);
```

   b. At the bottom of the program, end the macro definition with a %MEND statement.

```
%mend;
```

4. Validate the **ot** parameter value to ensure that it is either *1*, *2*, or *3*. If no value is provided, write a custom message to the log. The message should indicate that a value is required and that the program will stop executing. It should also include a list of valid values. If a value other than *1*, *2*, or *3* is provided, write a custom error message to the log that prints a list of valid values and stops processing the rest of the program.

   Note: Be sure to use the MINOPERATOR option in the %MACRO statement to enable the macro IN operator.

   a. Add the MINOPERATOR option in the %MACRO statement.

```
%macro supplierreport(ot) / minoperator;
```

   b. Use %IF and %END statements to test whether the parameter is equal to a null value. If it is, use %PUT statements to write a custom error message to the log that indicates that a value is required and that the program will stop executing. It should also include a list of valid values. The error message could appear as follows:

```
ERROR: You did not specify an Order_Type code (required).
       Valid Order_Type values include blank, 1 (retail), 2 (catalog), or 3 (internet).
       Program will stop executing
```

```
%if &ot= %then %do;
    %put ERROR: You did not specify an Order_Type code (required).;
    %put ERROR- Valid Order_Type values include blank, 1 (retail),
2 (catalog), or 3 (internet).;
    %put ERROR- Program will stop executing;
    %return;
%end;
```

   c. Use the %RETURN statement to stop execution and %END to close the %IF block.

   d. In the %MACRO statement, add the /MINOPERATOR option to be able to use the IN operator in a macro statement.

   e. Use %ELSE, %IF, and %END statements to test whether the parameter is not in the list of *1*, *2*, or *3*. Write a custom error message to the log if an invalid value is provided. The program should also stop executing. The error message could appear as follows:

```
ERROR: Valid Order_Type values include blank, 1 (retail), 2 (catalog), or 3 (internet).
       Program will stop executing.
```

   f. Use the %RETURN statement to stop execution and %END to close the %IF block.

   g. Use %ELSE, %DO, and %END statements to indicate whether the parameter value is valid, and then the rest of the program should run.

```
%else %if not(&ot in 1 2 3) %then %do;
    %put ERROR: Valid Order_Type values include blank, 1 (retail),
     2 (catalog), or 3 (internet).;
    %put ERROR- Program will stop executing;
    %return;
%end;

%else %do;

/*PART 1*/
```

```
<…rest of program…>
%end;
run;
```

5. If a value of *1*, *2*, or *3* is provided for the parameter, subset the **OrderDetail** table based on **Order_Type**.

   Modify the WHERE statement in the first PROC SQL step to include rows where **Order_Type** is equal to the **ot** macro variable.

```
proc sql;
create table OrderDetail as
    select Order_ID, o.Product_ID, Order_Type, Product_Category,
         Product_Group, Product_Line, Product_Name,
         (total_retail_price-(costprice_per_unit*quantity))
               as Profit,
         Supplier_ID, Supplier_Name, Supplier_Country
    from work.orders as o
        left join work.product_dim as p
      on o.Product_ID=p.Product_ID
    where order_type=&ot;
quit;
```

6. Create a series of macro variables that will store **Supplier_ID**, **Supplier_Name**, **Supplier_Country**, and **Profit** for each of the top five suppliers. For example, the macro variables **TopSupp1**, **Name1**, **Country1**, and **Profit1** will store information about the top supplier; **TopSupp2**, **Name2**, **Country2**, and **Profit2** will store information about the second-ranked supplier; and so on.

   a. Find the second PROC SQL step that identifies the top five suppliers.

   b. Add an INTO clause to create the following series of macro variables for the top five suppliers:

      1) **TopSupp1**-**TopSupp5** to store the **Supplier_ID** values.

      2) **Name1**-**Name5** to store the **Supplier_Name** values.

      3) **Country1**-**Country5** to store the **Supplier_Country** values.

      4) **Profit1**-**Profit5** to store the sum of **Profit** values.

```
proc sql;
select Supplier_ID format=12.,
       Supplier_Name,
       Supplier_Country,
       sum(profit) as Profit
    into :topsupp1-:topsupp5, :name1-:name5,
         :country1-:country5, :profit1-:profit5
    from OrderDetail
    group by Supplier_ID, Supplier_Name, Supplier_Country
    order by Profit desc;
quit;
```

7. Create a series of macro variables named **Country_*CC*** where ***CC*** is the two-letter **CountryCode** value read from the **work.country_codes** table. Assign the corresponding **CountryName** value to each macro variable.

a. Write a DATA step that reads the **work.country_codes** table.

b. Use CALL SYMPUTX to create the series of macro variables. The first argument should concatenate *Country_* with the value of **CountryCode** to create the macro variable names. The second argument should assign the value from the **CountryName** column.

```
data _null_;
    set work.country_codes;
    call symputx(cats('country_',CountryCode), CountryName);
run;
```

8. Use a macro DO loop to repeat Part 2 of the program five times. The first time through the loop, the program should generate the PDF report for the top supplier. The report should be modified as follows:

   a. The prefix for each PDF file name should be the supplier rank number, 1 through 5. The name of each PDF file should be the value of **Supplier_Name** with all spaces replaced with underscores. Use the **REPLACESPACE** custom macro function.

      1) After the ODS GRAPHICS statement, add a %DO macro statement with an index variable **i** that starts at 1 and ends at 5.

      2) At the end of the program, before the %END statement (this closes the %IF %THEN/%DO block), add another %END statement.

      3) After the %MACRO statement, add a %LOCAL statement to ensure that **i** is written to and read from the local symbol table.

```
%macro supplierreport(ot) / minoperator;
%local i;
...
ods graphics on / imagefmt=png;
%do i=1 %to 5;
...

    footnote;
%end;
%end;
ods pdf close;
%mend supplierreport;
```

      4) In the ODS PDF statement, delete the hardcoded supplier name, *1_Eclipse_Inc* (keep the .pdf extension) and replace it with an expression that does the following:

         a) calls the **%replacespace** macro

         b) includes the value of the macro variable **i** followed by an underscore as the file name prefix.

         c) uses an indirect macro variable reference as the parameter for the **%replacespace** macro. The indirect reference should substitute the value of the **Name1**, **Name2** (and so on) macro variable.

```
ods pdf file="&path/%replacespace(&i &&name&i).pdf"
         style=meadow startpage=no nogtitle notoc;
```

b. The first title should be the rank of the supplier and then the **Supplier_Name** value, followed by the full country name for that particular supplier (for example, *Orders for #1 Eclipse Inc, United States*).

   1) After the ODS statement, add a %LET statement to create a macro variable named **CC** that will be the two-letter **CountryCode** for the supplier being analyzed in the loop. (For example, when **i=1**, the value of **CC** is the **CountryCode** assigned to the **Country1** macro variable.) This requires an indirect macro variable reference. This macro variable is used later to insert the country name in the title.

   2) In the TITLE statement, use the **i** macro variable to substitute the rank number of the supplier.

   3) Use an indirect macro variable reference to substitute the macro variable value for **Name1**, **Name2**, and so on.

   4) Use an indirect macro variable reference to substitute the full country name. Remember that the macro variable **Country_*CC***, where ***CC*** is the two-letter **CountryCode** for the supplier, stores the country name. Use the **CC** macro variable created earlier as part of the indirect reference.

```
%let cc=&&country&i;
title "Orders for #&i &&name&i, &&country_&cc";
```

c. The second title should be one of the following, depending on the value of the **ot** parameter: *Retail Sales Only*, *Catalog Sales Only*, or *Internet Sales Only*.

   To create the second title, use %IF, %THEN, and %END statements to provide unique TITLE2 statements depending on the value of the **ot** parameter.

```
%if &ot=1 %then %do;
    title2 "Retail Sales Only";
%end;
%else %if &ot=2 %then %do;
    title2 "Catalog Sales Only";
%end;
%else %if &ot=3 %then %do;
    title2 "Internet Sales Only";
%end;
%else %if &ot= %then %do;
    title2 "All Sales";
%end;
```

d. For the bar chart (PROC SGPLOT step), the data should be subset to include one supplier at a time.

   Modify the WHERE statement to use an indirect macro variable reference to substitute the **Supplier_ID** value.

```
proc sgplot data=OrderDetail;
    hbar Product_Category / response=profit stat=sum
                           group=Product_Group
                           categoryorder=respdesc;
    where Supplier_ID=&&topsupp&i;
    format profit dollar8.;
run;
```

e. For the report (PROC SQL step), a footnote should be added below the report that includes the date and time that the report was created. The data should also be subset to include only the top supplier.
   1) Add a FOOTNOTE statement before the last PROC SQL step.
   2) Use %SYSFUNC to execute the TODAY() function and format it with an appropriate date format.
   3) Use %SYSFUNC again to execute the TIME() function and format it with an appropriate time format.

```
...
footnote "As of %sysfunc(today(),weekdate.) at
  %sysfunc(time(),timeampm.)";
proc sql;
    select Product_Group,
...
```

9. Test the **%SupplierReport** macro program with parameter values of *1*, *2*, *3*, *4*, and null. Make sure that when *4* is provided, the error message is written to the log and no output is generated. Test that error messages are also generated if a null value is provided.

```
%supplierreport(1)
%supplierreport(2)
%supplierreport(3)
%supplierreport(4)
%supplierreport()
```

10. Save the **supplierreport.sas** macro program in the **autocall** folder.

```
options sasautos=("&path/autocall", SASAUTOS);

/*Create macro SupplierReport to generate PDF for each of the top 5
suppliers*/

%macro supplierreport(ot) / minoperator;
%local i;
%if &ot= %then %do;
    %put ERROR: You did not specify an Order_Type code (required).;
     %put ERROR- Valid Order_Type values include blank, 1 (retail),
2 (catalog), or 3 (internet).;
    %put ERROR- Program will stop executing;
    %return;
%end;

%else %if not(&ot in 1 2 3) %then %do;
     %put ERROR: Valid Order_Type values include blank, 1 (retail),
2 (catalog), or 3 (internet).;
     %put ERROR- Program will stop executing;
     %return;
%end;

%else %do;
```

```
/*PART 1*/
/*This step creates the OrderDetail table that joins Orders with
Products and Country_Codes.
  It calculates Profit for each row and include Retail Sales
(order_type=1) only */
     proc sql;
     create table OrderDetail as
          select Order_ID, o.Product_ID, Order_Type,
                 Product_Category, Product_Group, Product_Line,
                 Product_Name,
                (total_retail_price-(costprice_per_unit*quantity))
                 as Profit,
                Supplier_ID, Supplier_Name, Supplier_Country
           from mc1.orders as o
                left join mc1.products as p
                on o.Product_ID=p.Product_ID
           where order_type=&ot;
     quit;

     /*This step summarizes profit and ranks suppliers*/
     /*Generate macro variables for top 5 suppliers ID, Name, and
       sum of Profit*/
     proc sql;
     select Supplier_ID format=12.,
            Supplier_Name,
            Supplier_Country,
            sum(profit) as Profit
          into :topsupp1-:topsupp5, :name1-:name5,
               :country1-:country5, :profit1-:profit5
          from OrderDetail
          group by Supplier_ID, Supplier_Name, Supplier_Country
          order by Profit desc;
     quit;

    /*Use CALL SYMPUTX to create a series of macro variables named
      Country_CC where CC is the 2-letter CountryCode. Assign the
      corresponding CountryName value.*/
     data _null_;
         set mc1.country_codes;
         call symputx(cats('country_',CountryCode),CountryName);
     run;

     options nodate;
     ods graphics on / imagefmt=png;
     %do i=1 %to 5;

     ods pdf file="&path/case_study/%replacespace(&i &&name&i).pdf"
         style=meadow startpage=no nogtitle notoc;
```

```
     %let cc=&&country&i;
     title "Orders for #&i &&name&i, &&country_&cc";
     %if &ot=1 %then %do;
          title2 "Retail Sales Only";
     %end;
     %else %if &ot=2 %then %do;
          title2 "Catalog Sales Only";
     %end;
     %else %if &ot=3 %then %do;
          title2 "Internet Sales Only";
     %end;
     %else %if &ot= %then %do;
          title2 "All Sales";
     %end;

     proc sgplot data=OrderDetail;
         hbar Product_Category / response=profit stat=sum
         group=Product_Group categoryorder=respdesc;
         where Supplier_ID=&&topsupp&i;
         format profit dollar8.;
     run;

     title;
     footnote "As of %sysfunc(today(),weekdate.) at
               %sysfunc(time(),timeampm.)";
     proc sql;
         select Product_Group,
                count(order_id) as NumOrders "Number of Orders",
                sum(profit) as TotalProfit "Total Profit"
                format=dollar8.,
                avg(profit) as AvgProfit "Average Profit per Order"
                format=dollar6.
         from OrderDetail
         where Supplier_ID=&&topsupp&i
         group by Product_Group
         order by calculated numorders desc;
     quit;

     footnote;
     %end;
%end;
ods pdf close;
%mend supplierreport;

%supplierreport(1)
*%supplierreport(2)
*%supplierreport(3)
*%supplierreport(4)
*%supplierreport()
```

Answer the Following Business Questions:

Using the reports created by the macro program, answer the following business questions.

1. Which company is the #2 supplier for internet sales only (**Order_Type**=*3*)? **Magnifico Sports**
2. What was the top product group for the top supplier for catalog sales only (**Order_Type**=*2*)? **Eclipse Shoes**
3. Which country is the #4 supplier for retail sales only (**Order_Type**=*1*)? **Sweden**