Paper 146-30

# New to SAS® and New to Programming? What You Need to Do Before Typing Code

Stephanie R. Thompson, The University of Memphis, Memphis, TN


## ABSTRACT

People from all kinds of backgrounds are using SAS® software to meet their business needs. Some come with previous programming experience and some have never typed a line of code in their career. Have you been told what each proc does, that each DATA step must end in a run command, and then been told to write a SAS program? Have you been given code to launch and told to " just change the date and run it every week"? Beware. There is a lot more to SAS programming than knowing some syntax and launching a job. Being a successful SAS programmer means bringing yourself up to speed with some basic programming skills. Before typing even the first line of code into the Enhanced Editor, some preparation needs to be done. Doing this will help you more than you can imagine. This paper provides the basics of programming that can help you write good code and ensure you are getting the answer you intended to get.


## INTRODUCTION

If you are new to SAS, you may already know that companies today do not always have the budget for training. When new people join a company they are at times given a few quick pointers on SAS or maybe told to work with a " mentor" before being left on their own to write code. This is not enough "training" to use a tool like SAS. Knowing some syntax is not the same as understanding how to program. This paper is intended to provide some tools for writing efficient programs and to help ensure the expected results are achieved. This is not a paper on SAS syntax, but this paper will demonstrate an example that incorporates SAS to illustrate programming pointers.

I have worked with many people who are new to SAS. Some have a programming background and others do not. Those that have programmed before seem to pick up the syntax of SAS fairly easily. They use the basic precepts of good programming to form their program. Those without a programming background struggle at times with what to type first and begin typing code into the editor without a clear plan. Some may generate code that gets an answer, but in many instances it is cumbersome and not always answering the question being asked. On many occasions, I have been asked by a new programmer to " look at my code and tell me if the answer is right." My question to them is always, " What question were you trying to answer?"

Code without errors and warnings will produce output. But is this the output that was intended? Was the question answered? That depends. If the right data sources are being hit and if good code has been written then the answer is more than likely yes. But is the output correct? Following these few simple steps will help ensure the answer to the question is yes.

1. Define the question to answer.
2. Identify the data needed to answer the question.
3. Draw out the flow of steps needed to get an answer.
4. Add broad code elements to the program blocks.
5. Fill in computations.
6. Do a logic check.
7. Start coding in SAS software.

These seven steps outlined above will be explained further within the context of an example.

The Example:
Write a program to generate weekly sales for the month of October 2004 and determine the gross profit for each week.

## DEFINE THE QUESTION TO ANSWER

One of the most important steps is to define the question that needs to be answered. It is impossible to determine the correct answer if the complexities of the question are not understood. . Ask questions to find out what type of information is needed and over what time interval. Is summary data or time series data needed? What types of statistics should be calculated? These are just a few examples. Information gathered from this step will be fundamental when building the SAS program in a later step.

## IDENTIFY THE DATA NEEDED TO ANSWER THE QUESTION

This is the step where data sources and specific variables are identified. You need to know where to get the data in order to start answering the question. Are the data in one table or many tables? Are the tables in SAS datasets or do you need to import the data from another source? Sometimes there are multiple sources for the same data. For example, there may be one inventory table that has a weekending inventory and another table that has a current inventory. Which one should be used? Snapshot tables (i.e., the weekend inventory) tend to run faster than live tables with changing data. Also, using a static table allows the results to be reproduced each time the program runs. If the answer needs to come from the live tables, the program may run a bit slower and the results can change each time the program runs.

In the example problem, there is a table called dsales.sas7bdat that contains daily sales. This table includes item number, quantity sold, and retail price for each day. Cost data, needed to calculate gross profit, is in the table called cost.sas7bdat and it includes item number and cost. The common element between the tables is item number. This is how the two data tables will be combined.

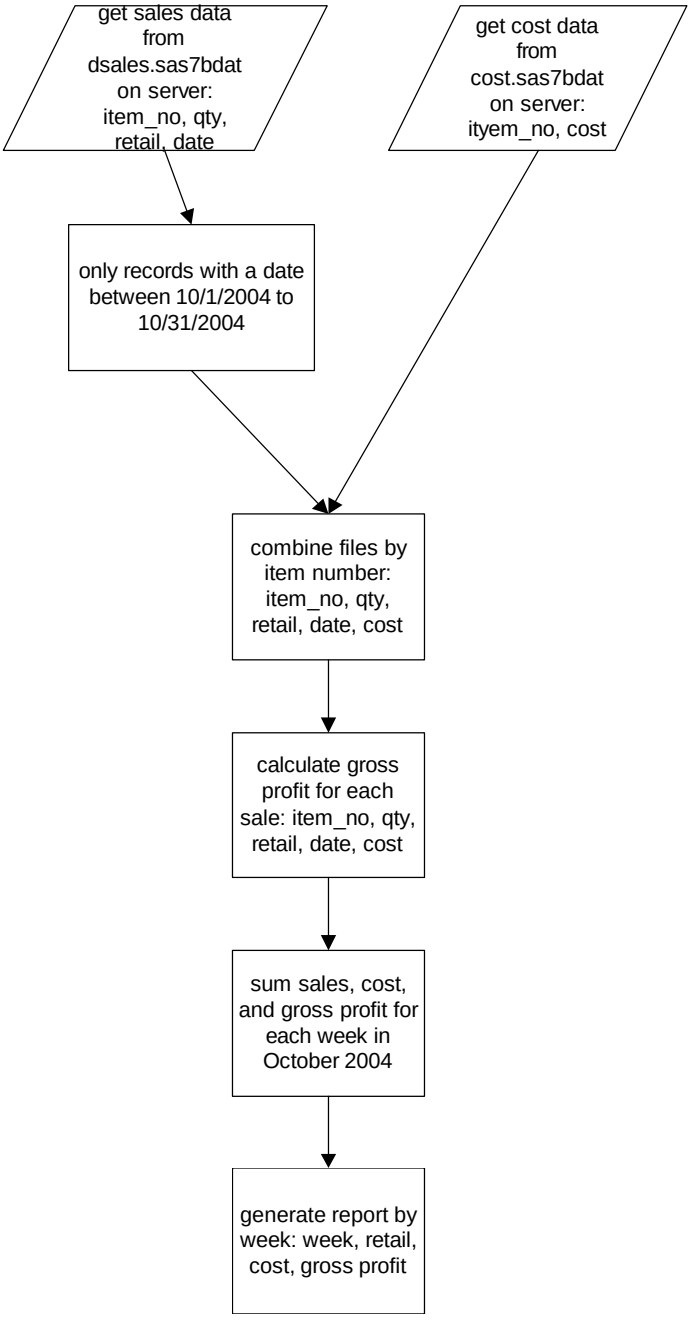
## DRAW OUT THE FLOW OF THE STEPS NEEDED TO GET AN ANSWER

Flowcharting is the term that was used when I took my first programming classes in college. In formal flowcharting, there are special shapes to use for each function and a lot of terminology to remember. Using fancy templates or special drawing software is not necessary. Flowcharting can be accomplished with nothing more than scrap paper and a pencil. The main point of this step is recognizing and understanding the logical flow for your program.

This step is where the flow or sequence of the program is physically laid out. How things come together and what happens next is shown graphically. This visual aid makes it much easier to spot inefficiencies and bad logic. Add the variable names that you need from each table and what is carried along in each step. This will keep table sizes smaller by not carrying along variables that are not used. Additional details will be added to the drawing further on.

Use broad program blocks to identify each step in solving the problem. Think of the program blocks like steps in a path. Draw as many blocks necessary to get the answer and describe the steps well so that it is apparent what each step is accomplishing. A block called “get data” is not specific enough. Calling this block “get sales data from dsales.sas7bdat on shared drive” is much clearer.

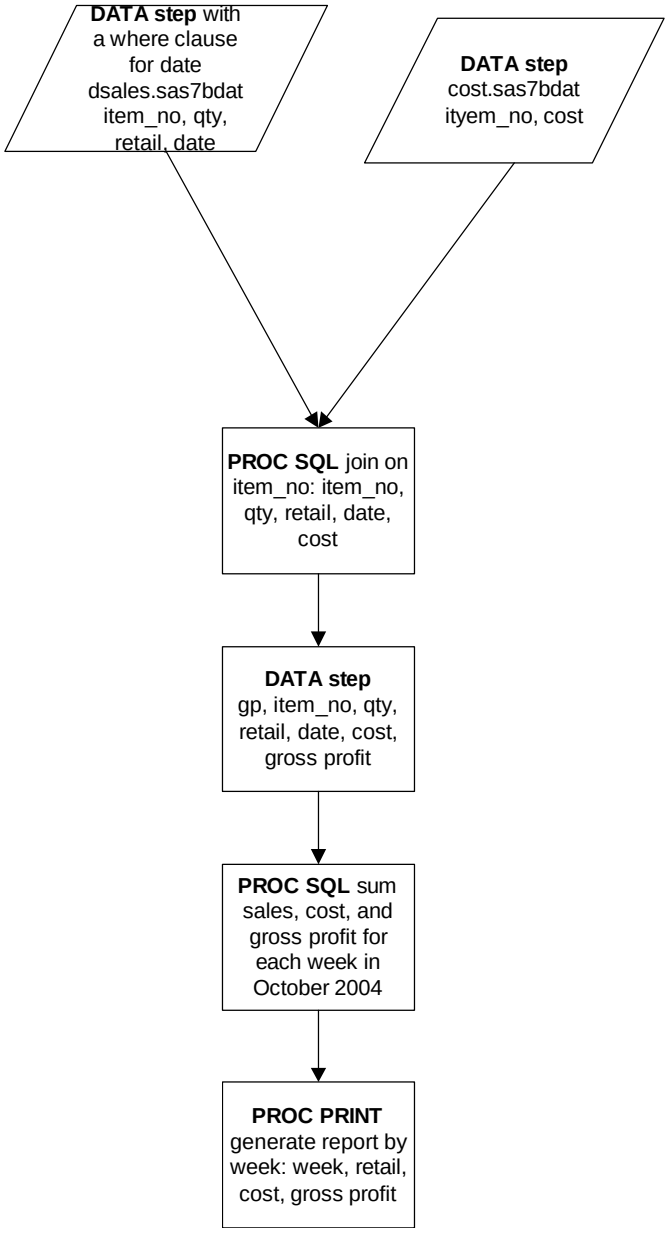
The drawing on the next page depicts how the flow chart would be drawn and labeled:


get sales data from dsales.sas7bdat on server: item_no, qty, retail, date
get cost data from cost.sas7bdat on server: ityem_no, cost
only records with a date between 10/1/2004 to 10/31/2004
combine files by item number: item_no, qty, retail, date, cost
calculate gross profit for each sale: item_no, qty, retail, date, cost
sum sales, cost, and gross profit for each week in October 2004
generate report by week: week, retail, cost, gross profit

## ADD BROAD CODE ELEMENTS TO THE PROGRAM BLOCKS

The next step is to identify the SAS procedure or other type of SAS code for each step in the drawing. Sometimes this is referred to as pseudo code. This is the first time that SAS is being used in the process of developing this program. You are not writing full code here; just note broadly what is being used in each block. Use terms like "PROC SQL," "PROC UPLOAD," and " DATA step to calculate loss." Choose the code that seems best. If you prefer SQL to a merge then the SQL should be identified. Use the name of the main part of the proc or other element on the drawing. Do not try to fill in all of the needed code or syntax.

The following flowchart has been updated to include the code elements.

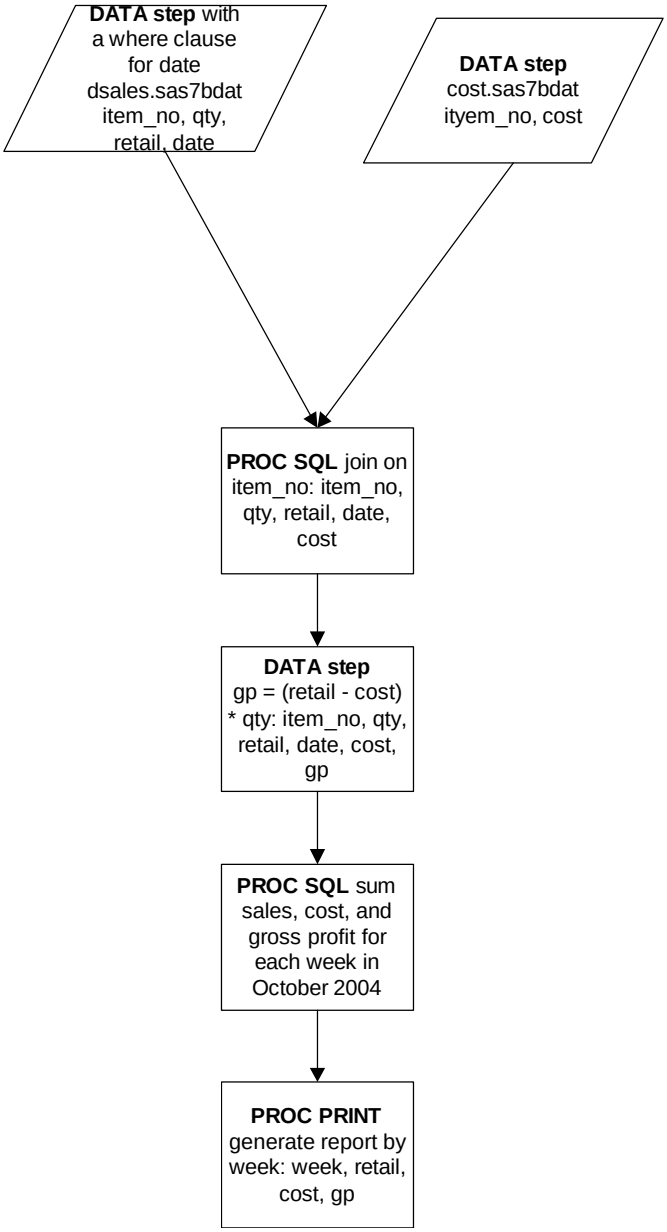


**FILL IN ANY COMPUTATIONS**

This simple step identifies any other actions occurring in the program. Included here are mathematical operations and the use of functions (i.e., max, min, average) or calculations generating a new variable. Identify the names of any new variables you are creating (i.e., gp for gross profit) and list them along with the other variables.

In the example, gross profit needs to be calculated. The equation gross profit = total retail dollars – total cost dollars will be used.

The computations are added to the following drawing:



## DO A LOGIC CHECK

Have any steps been missed? Is there a better way to perform a function? Now is the time to see if everything flows correctly and to tweak the steps. Taking time to check the logic will greatly assist in the coding that follows in the next step.

## START CODING IN SAS SOFTWARE

In this step, the actual SAS coding begins. Everything up to this point has been in preparation for writing code. The fleshed-out drawing can be used as the basis of the code. All of the necessary steps and some of the general code will have been identified. This process makes the actual coding much less time consuming and more accurate. Now it is just a matter of typing and syntax.

## CONCLUSION

There are many ways to answer a question using SAS. Different procedures and elements will get the same answer. Again, this paper is not intended to help you select which to use. The intent is to promote a better understanding of how efficient programs are put together. The approach covered in this paper can be used for very simple programs, like the one in the example, to the very complex.

The steps in developing SAS programs are not a set of hard and fast rules. It is intended as an outline to allow you to build code more effectively. The newer you are to programming, the more closely you will want to follow this process. As a programmer's experience level increases, the process becomes more comfortable. Eventually some of the formal steps may be truncated or modified to meet the needs of proficient programmers. The mental picture of the process becomes clearer after doing this for a longer period of time. Planning before coding is the foundation for good programming.

## ACKNOWLEDGMENTS

Thank you, Darla Keel, for your insightful editing and time.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at the following address:

Stephanie R. Thompson
The University of Memphis
411 Administration Bldg.
Memphis, TN 38152-3370
Work Phone: (901)678-5529
Fax: (901) 678-5138
Email: srthmpsn@memphis.edu
Web: http://oir.memphis.edu