SC-15

An Annotated Guide: Using Proc Tabulate And Proc Summary to Validate SAS Code

Russ Lavery, Contractor, Ardmore, PA

ABSTRACT

This paper discusses how Proc Tabulate and Proc Summary can be used to help a programmer avoid errors. This paper might be especially useful for new programmers.

At every step of a program, a programmer should know were "every observation goes" and Proc Tabulate can be used to keep track of observations, not just for creating reports. Additionally, Good Programming Practice requires that a programmer "exercise" (check in some way) every combination of the clauses in every IF statement to see how observations are classified by the IF. Proc Tabulate is an excellent, fast and simple tool for doing that checking. Importantly, Proc Tabulate produces a pattern in its output that makes checking fast. In most cases, the logic need only be closely examined where the pattern in the output changes. Additionally, using Proc Tabulate allows the programmer to account for every observation in the data set, a very "comforting" fact.

Many programing tasks require taking subsets of the data, based on logical rules. Programmers are often told to check the number of obseravtions in the starting data set and not to lose, or create, observations whithout knowing why. With Proc Summary we can also suggest that a programmer also check the sums of important variables (Dollars, Scripts, units sold) in the source data set and never change the sum wihtout knowing why.

SECTION 1) VALIDATING CODE WITH PROC TABULATE:

Complex if statements can be easily checked using Proc Tabulate by exploiting patterns in Proc Tabulate output. Learning some Proc Tabulate options make the job easier. First, use the ALL option on both the row and column dimensions in the table to show column, row and overall totals. Second, the missing option should be used so that missing values of the class variables show up in the table. Thirdly, use the standard tricks (noseps, PS=, LS= format= and RTS=) to make output fit on a page for easy reading. Finally, creating the table in a logical manner makes the pattern in the output easier to spot. As a warning, it is preferable to use this technique on a test data set or a final version of the data.

GOALS AND PROBLEMS IN DUBUGGING

A programmer should account for every observation and should "exercise" every path through a series of IF statements and rigorously check assignment formulas. The SAS method of propagating missing values makes this task difficult for complex assignment statements. This paper uses IF statements as examples for checking logic, but the technique is also useful for checking other assignments like formats.

Imagine a typical series of IF statements with four components and a result. (this example is not carried further in the paper)
If state="CA" and sex="F" and hobby="Beach" and Group="Beach Boys" then Personality="California Girl"
Else ……….

It is difficult to predict the number of obs. in any level of the variable Personality from looking at freqs of variables (state, hobby and group). Predicting counts of people in the different levels of Personality is especially complex where there are many missing values for state, sex, hobby and group. The predicting counts problem is difficult because observations can have missing values for more than one of the variables (state, sex, hobby or group). The simple freqs for state sex, hobby and group, shown below, do not show that there are obs that have multiple missing/Unknown values.

```
                                          Cumulative     Cumulative
sex       Frequency        Percent        Frequency      Percent

F         100              0.307          100            0.30
M         200              0.615          300            0.92
Unk       25               0.076          325            1.00

                                          Cumulative     Cumulative
State     Frequency         Percent       Frequency      Percent

CA        272               0.836         272            0.83
DE        8                 0.024         280            0.86
Unk       45                0.138         325            1.00
```

The output above does not show if there are obs that have missing values for both state and sex. The number of obs that “fail” the above IF statement might be different from the number of obs with missing sex added to the number of obs with missing state added to the number of obs with missing group. A ‘multivariate” tool is required for checking complex IF statements and assignments. Proc Tabulate is just that tool. For simple IF statements, Proc Freq can also be used in replace of Proc Tabulate. However, Proc Freq is less powerful than Proc Tabulate and it is more difficult to use Proc Freq to create output with a graphic pattern that can easily be checked.

Below is a program that reads data and creates variables using a fairly complex series of IF statements. It is not suggested that a reader study this coding. It is just an example of very messy code. It is included so that a reader can have an example of messy doce without having to create it. It is thought that a reader could cut and paste it into SAS of they want ot have code to play with.

Remember Proc Tabulate is a very powerful cross-tab producer. Think of the cross-tabs as having variables “on the side” of the table and “across the top” of the table.

The Proc Tabulate statement that specifies the look/form of the table is:
Table list of variables to be put on the side of the table (separated by an *) - a comma – a list of variables to be put across the across the top of the table.

The trick is: Put the variables on the left side of the = in your IF statement on the “side of the table” and the resulting class on the “top of the table”. Examples are shown below.

The task is to assign values of an imaginary physical exam test to three levels (low normal or high) based on the cutpoints below. Cutpoints vary with sex, age, lifestyle and test (though only a treadmill test is of interest to the client)

SAMPLE SAS CODE FOR THE EXAMPLE USED IN THE REST OF THE PAPER

```
/*****************************************************************
Program: TABULATE_4_CC     Programmer:  Russ Lavery            Date 11/25/2005
Purpose:
Illustrate using proc tabulate to check if statement coding.
If statements are to be used to check assignment of flags to lab tests
Create a data set with multiple missing values to make coding difficult
Logic for cutpoints is shown below:
                                          Lower Cutpoint     Upper Cutpoint
sex  age   pre_condition   Test      Lower Normal value  Upper normal value
M    7-12   sedentary      Treadmill   25                 28
M    13-21  sedentary      Treadmill   16                 22
M    21-50  sedentary      Treadmill   19                 24
M    51-70  sedentary      Treadmill   23                 26
M    7-12   active         Treadmill   20                 23
M    13-21  active         Treadmill   11                 17
M    21-50  active         Treadmill   14                 19
M    51-70  active         Treadmill   18                 21
M    7-12   athletic       Treadmill   15                 18
M    13-21  athletic       Treadmill    6                 12
M    21-50  athletic       Treadmill    9                 14
M    51-70  athletic       Treadmill   13                 16
F    7-12   sedentary      Treadmill   27                 30
F    13-21  sedentary      Treadmill   18                 24
F    21-50  sedentary      Treadmill   21                 26
F    51-70  sedentary      Treadmill   25                 27
F    7-12   active         Treadmill   22                 25
F    13-21  active         Treadmill   13                 19
F    21-50  active         Treadmill   16                 21
F    51-70  active         Treadmill   20                 23
F    7-12   athletic       Treadmill   17                 20
F    13-21  athletic       Treadmill    8                 14
F    21-50  athletic       Treadmill   11                 16
F    51-70  athletic       Treadmill   15                 18
*****************************************************************/
```

```
data imaginary_labs ;
infile datalines firstobs=4 missover;
 input @1 name $char6.  @9 visit 1.  @15 sex $1. @21 age 2.  @30 lifestyle $char9.
         @45 test $char9.  @60 value 2.0;
tflag=" Initial";
if test="Treadmill" then
do;
   tflag="looping"; /*Just to track execution*/
   if sex="M" then
   do;
      If lifestyle="Sedentary" then
      do;
       tflag="Sed";
          if      age GE 7  and age LT 12 then
             do;
                 if       value LT 25 then tflag="1_Low ";
                 else if  value GT 28 then tflag="3_High";
                 else tflag="2_Norm";
             end;
            Else if age GE 13  and age LT 21 then
              do;
                 if       value LT 16 then tflag="1_Low ";
                 Else if  value GT 22 then tflag="3_High";
                 else tflag="2_Norm";
              end;
           ELSE if age GE 21 and age LT 50 then
             do;
                 if       value LT 19 then tflag="1_Low ";
                 Else if  value GT 24 then tflag="3_High";
                 else tflag="2_Norm";
             end;
           Else if age GE 51 and age LE 70 Then
             do;
                 if      value LT 23 then tflag="1_Low ";
                 else if value GT 26 then tflag="3_High";
                 else tflag="2_Norm";
            end;
        end; /*Male Sedentary*/
          Else If lifestyle="Active" then
           do;
           tflag="Act";/*Just to track execution*/
             if      age GE 7  and age LE 12 then
              do;
                 if      value LT 20 then tflag="1_Low ";
                 else if value GT 23 then tflag="3_High";
           else tflag="2_Norm";
              end;
             if age GE 13 and age LE 21 then
              do;
                 if      value LT 11 then tflag="1_Low ";
                 else if value GT 17 then tflag="3_High";
                 else tflag="2_Norm";
               end;
             if age GE 21 and age LE 50 then
               do;
                 if      value LT 14 then tflag="1_Low ";
                 else if value GT 19 then tflag="3_High";
                 else tflag="2_Norm";
                 end;
              if age GE 51 and age LE 70 then
                do;
                  if      value LT 18 then tflag="1_Low ";
                  else if value GT 21 then tflag="3_High";
                  else tflag="2_Norm";
                 end;
           end; /*Male Active*/
          Else If lifestyle="Athletic" then
           do;
             tflag="Ath";/*Just to track execution*/
            if      age GE 7  and age LE 12 then
                  do;
                    if      value LT 15 then tflag="1_Low ";
                   else if value GT 18 then tflag="3_High";
                    else tflag="2_Norm";
                  end;
            if age GE 13 and age LE 21 then
                  do;
                    if      value LT  6 then tflag="1_Low ";
                    else if value GT 12 then tflag="3_High";
                    else tflag="2_Norm";
                  end;
            if age GE 21 and age LE 50 then
                  do;
                   if      value LT  9 then tflag="1_Low ";
                   else if value GT 14 then tflag="3_High";
                   else tflag="2_Norm";
                  end;
            if age GE 51 and age LE 70 then
                  do;
                  if      value LT 13 then tflag="1_Low ";
                  else if value GT 16 then tflag="3_High";
                  else tflag="2_Norm";
                  end;
           end; /*M athletic*/
   end; /*sex = M*/
   ELSE if sex="F" then
   do;
          If lifestyle="Sedentary" then
           do;
            tflag="Sed";
             if      age GE 7  and age LT 12 then
               do;
                  if       value LT 27 then tflag="1_Low ";
                  else if  value GT 30 then tflag="3_High";
                  else tflag="2_Norm";
               end;
            Else if age GE 13  and age LT 21 then
              do;
                  if       value LT 18 then tflag="1_Low ";
                  Else if  value GT 24 then tflag="3_High";
                  else tflag="2_Norm";
              end;
            ELSE if age GE 21 and age LT 50 then
              do;
                 if       value LT 21 then tflag="1_Low ";
                 Else if  value GT 26 then tflag="3_High";
                  else tflag="2_Norm";
                end;
            Else if age GE 51 and age LE 70 Then
                  do;
                  if      value LT 25 then tflag="1_Low ";
                  else if value GT 28 then tflag="3_High";
                 else tflag="2_Norm";
                 end;
           end; /*F Sedentary*/
          Else If lifestyle="Active" then
           do;
              tflag="Act";/*Just to track execution*/
            if      age GE 7  and age LE 12 then
                 do;
                 if      value LT 22 then tflag="1_Low ";
                 else if value GT 25 then tflag="3_High";
                 else tflag="2_Norm";
                 end;
            if age GE 13 and age LE 21 then
                 do;
                 if      value LT 13 then tflag="1_Low ";
                 else if value GT 19 then tflag="3_High";
                 else tflag="2_Norm";
                 end;
            if age GE 21 and age LE 50 then
                 do;
                 if      value LT 16 then tflag="1_Low ";
                 else if value GT 21 then tflag="3_High";
                 else tflag="2_Norm";
                 end;
                 if age GE 51 and age LE 70 then
                  do;
                   if      value LT 20 then tflag="1_Low ";
                   else if value GT 23 then tflag="3_High";
                   else tflag="2_Norm";
                   end;
           end; /*F Active*/
          Else If lifestyle="Athletic" then
           do;
              tflag="Ath";/*Just to track execution*/
            if      age GE 7  and age LE 12 then
                 do;
                 if      value LT 17 then tflag="1_Low ";
                 else if value GT 20 then tflag="3_High";
                 else tflag="2_Norm";
                 end;
            if age GE 13 and age LE 21 then
                 do;
                 if      value LT  8 then tflag="1_Low ";
                 else if value GT 14 then tflag="3_High";
                 else tflag="2_Norm";
                 end;
            if age GE 21 and age LE 50 then
                 do;
                 if      value LT 11 then tflag="1_Low ";
                 else if value GT 16 then tflag="3_High";
                 else tflag="2_Norm";
                 end;
            if age GE 51 and age LE 70 then
                 do;
                 if      value LT 15 then tflag="1_Low ";
                 else if value GT 18 then tflag="3_High";
                 else tflag="2_Norm";
                 end;
           end; /*F athletic*/
   end; /*sex = F*/
End ; /*test=treadmill*/
```

```
datalines;
          1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
name    visit sex   age     lifestyle      test           value
Illust  1     M      8      Sedentary      Treadmill      18
Illust  2     M      8      Sedentary      Treadmill      18
Illust  3     M      8      Sedentary      Treadmill      19
Illust  4     M      8      Sedentary      Treadmill      20
Illust  5     M      8      Sedentary      Treadmill      26
Illust  6     M      8      Sedentary      Treadmill      30
Illust  7     M      8      Sedentary      Treadmill      33
Illust  8     M      8      Sedentary      Treadmill      35
Russ    1     m     55      Sedentary      Treadmill      28
Russ    2     M     55      Sedentary      Treadmill      29
Russ    3     M     55      Sedentary      Treadmill      11
Debb    1     F     29      Athletic       Treadmill      09
Debb    2     F     29      Athletic       Treadmill      12
Debb    3     F     29      Athletic       Biking         18
Shu     1     F     29      Athletic       Treadmill      10
Shu     2     F     29      Athletic       Treadmill      11
Alex    1     M     12      Active         Treadmill      26
Alex    2     M     12      Active         treadmill      19
Pam     1     F     29      Athletic       Treadmill       7
Pam     2     F     29      Athletic       Treadmill      15
Pam     3     F     29      Athletic       Treadmill      20
Guy     1     M     29      Athletic       Treadmill      15
steve   1     M     30      active         Treadmill      16
steve   2     M     30      Active         Treadmill      16
steve   3     M     30      Active         Treadmill      .
steve   4           30                     Treadmill      16
; run;

options ls=110 PS=70;
PROC TABULATE DATA=imaginary_labs MISSING FORMAT=6.0;
CLASS SEX LIFESTYLE TEST AGE TFlaG VALUE;
TABLE test*SEX*LIFESTYLE*AGE*VALUE ALL,TFLAG ALL /RTS=60;  RUN;

options ls=150;
PROC TABULATE DATA=imaginary_labs MISSING FORMAT=5.0;
CLASS SEX LIFESTYLE TEST AGE TFlaG VALUE;
TABLE test*LIFESTYLE*AGE*VALUE ALL,SEX*TFLAG ALL /RTS=40;  RUN;

PROC TABULATE DATA=imaginary_labs MISSING FORMAT=5.0 noseps;
title "nospepts can sometimes be used to make the table fit on a page";
CLASS SEX LIFESTYLE TEST AGE TFlaG VALUE;
TABLE test*LIFESTYLE*AGE*VALUE ALL,SEX*TFLAG ALL /RTS=40;     RUN;
```

The IF statements take up many lines of code and our task is to check both the data and our accuracy in coding the IF statements. The above code and three Proc Tabulates can be pasted into SAS and used as a starting point for learning..

HINTS TO MAKE THE TECHNIQUE MORE EFFECTIVE

Proc Tabulate is most useful when a properly designed test data set is created to test the IF statements and then Proc Tabulate is used to show how those observations were assigned. With a properly constructed test dataset, the programmer can know that s/he has exercised all the logically possible paths through the IF statements.

However, it is reasonably good practice to use Proc Tabulate to check how “real data” is classified if the data is in its final form (no more refreshing to be done), even if *all* paths through the IF statements are not exercised. If the data set is in the final state, this technique tests all the combinations of variables usedin the IF statements *that are actually in the data*. This practice does not check that your code is perfect, but it does check that your code has perfectly classified all the data that happens to be in the dataset. If the data set is “frozen”, this is a reasonably good check of coding quality.

It is dangerous to use this Proc Tabulate technique on data that will be refreshed at a future date, as often happens in a clinical trial. It would be poor practice to check an early version of a clinical trial dataset (say with 25% of the observations entered) and to consider the program logic "validated" because Proc Tabulate shows that all that 25% of the obs were classified correctly. New data, with new combinations of variables, could bring new problems.

Below, in blue, find the log note for the creation of the example dataset followed by the output from the first Proc Tabulate. The log (pasted in in blue) shows there are 26 observations in the dataset and all 26 show up on the tabulate (SEE RED 26 IN Lower Right Hand Corner of the table below). Since the table shows every observation in the dataset, if we understand this table, we understand the whole dataset. If we use a QC procedure that checks every obs in the dataset, there will be no hidden surprises.

The use of continuous variables in the IF can make the table long, but there are some things to do to make the output easier to read and to make the logic pattern in the data stand out. It is usually not necessary to examine every line of the table. One can save much time by exploiting patterns in the table. Exploiting patterns will be illustrated below.

It is suggested that one use the all option on both the row and column dimension in the Proc Tabulate. This causes printing of total number of obs in the table in the lower right hand corner (the red 26 below). The missing option should be used in the Proc Tabulate so that missing values of the class variables show up (see blue box below) in the table. Using these options assures that the number is the lower right hand corner of the table is the number of obs in the data set. In this example, note that the last subject in the data set, Steve, has several missing values. An example of an obs with multiple missing class values is seen in the solid green box. An obs with a missing continuous value is shown in the dotted green box below.

Formatting the output so that all the output fits on one page, and structuring the table to emphasize the pattern in the logic, makes for easier reading. Use the standard Proc Tabulate tricks (noseps, PS=, LS=, format= and RTS=) to make output fit on a page.

The assigned Flag values have a numeric prefix to force a logical sorting (1-Low, 2_Norm…) order in the Proc Tabulate output. A logical sorting order makes the pattern in the output easier to spot (see red box to be discussed later). There are other ordering tricks.

OUTPUT AND INTERPRETATION

What jumps out from the table below is that data issues that have affected the IF statements. There is a biking test (green box), that should not be in the data at all. Treadmill is spelled two ways. Male is coded in upper and lower case and athletic is spelled two ways. Finally, there is a missing value for the variable named value. Tabulate lets a programmer check EVERY observation in the data set and determine why classifications were made as they were.

In well-conditioned data, patterns jump out of the table. This data set is small and of very poor quality so the pattern is not as strong as it usually is in real data. To see an example of the pattern, look at the solid red boxes. It is suggested that this pattern be read as: F or treadmill, athletic females with values from 7 to 15 are coded to "1_LOW". All subjects with values between 7 and 15 can be considered as a group. If subjects with value of 7 and 15 are classified correctly, we likely do not need to check the values in between 7 and 15. At 20, the pattern shifts to classifying athletic females to High.. Pay attntion to where the pattern shifts.

Even if there were a great many levels of the variable value, this table could be validated quickly by checking the first and last elements in a group.

The output can run over several pages and a speed trick is to check logic at 1) the start of a section (treadmill-M-8 years old), 2) the end of a section and 3) where the pattern changes inside a section. Two pattern changes are illustrated by the arrows slanting down and to the right. The "shift to the right" is a break in the pattern and an indication that a different IF statement took effect.

The pattern can be exploited to speed checking as follows: if M-Sedentary-8-18 is correct and M-Sedentary-8-20 is correct, we can assume rows between these two rows (here just M-Sedentary-8-19) are correct. In real data, there can be many rows between the values you check. If the data is fairly clean and the table can be structured so that the pattern is easy to see, every observation in very large data sets can be checked quickly by checking the places where the pattern changes.

```
NOTE: The data set WORK.IMAGINARY_LABS has 26 observations and 8 variables.
NOTE: DATA statement used (Total process time):
      real time           0.57 seconds        cpu time            0.07 seconds
„ffffffffffffffffffffffffffffffffffffffffffffffffffffff…ffffffffffffffffffffffffffffffffff…ffffff†
,                                                      ,              tflag               ,      ,
,                                                      ‡ffffff…ffffff…ffffff…ffffff…ffffff‰      ,
,                                                      ,Initi-,      ,      ,      ,loopi-,      ,
,                                                      ,  al  ,1_Low ,2_Norm,3_High,  ng  , All  ,
,                                                      ‡ffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,                                                      ,  N   ,  N   ,  N   ,  N   ,  N   ,  N   ,
‡ffffffffff…ffffffffff…ffffffffff…ffffffffff…ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,test      ,sex       ,lifestyle ,age       ,value         ,      ,      ,      ,      ,      ,      ,
‡ffffffffffˆffffffffffˆffffffffffˆffffffffffˆffffffffffffff‰      ,      ,      ,      ,      ,      ,
,Biking    ,F         ,Athletic  ,29        ,18            ,     1,     .,     .,     .,     .,     1,
‡ffffffffffˆffffffffffˆffffffffffˆffffffffffˆffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,Treadmill ,          ,          ,30        ,16            ,     .,     .,     .,     .,     1,     1,
,          ‡ffffffffffˆffffffffffˆffffffffffˆffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,F         ,Athletic  ,29        ,7             ,     .,     1,     .,     .,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,9             ,     .,     1,     .,     .,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,10            ,     .,     1,     .,     .,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,11            ,     .,     .,     1,     .,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,12            ,     .,     .,     1,     .,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,15            ,     .,     .,     1,     .,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,20            ,     .,     .,     .,     1,     .,     1,
,          ‡ffffffffffˆffffffffffˆffffffffffˆffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,M         ,Active    ,12        ,26            ,     .,     .,     .,     1,     .,     1,
,          ,          ,          ‡ffffffffffˆffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,30        ,.             ,     .,     1,     .,     .,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,16            ,     .,     .,     1,     .,     .,     1,
,          ,          ‡ffffffffffˆffffffffffˆffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,Athletic  ,29        ,15            ,     .,     .,     .,     1,     .,     1,
,          ,          ‡ffffffffffˆffffffffffˆffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,Sedentary ,8         ,18            ,     .,     2,     .,     .,     .,     2,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,19            ,     .,     1,     .,     .,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,20            ,     .,     1,     .,     .,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,26            ,     .,     .,     1,     .,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,30            ,     .,     .,     .,     1,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,33            ,     .,     .,     .,     1,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,35            ,     .,     .,     .,     1,     .,     1,
,          ,          ,          ‡ffffffffffˆffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,55        ,11            ,     .,     1,     .,     .,     .,     1,
,          ,          ,          ,          ‡ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,          ,          ,29            ,     .,     .,     .,     1,     .,     1,
,          ,          ‡ffffffffffˆffffffffffˆffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,          ,active    ,30        ,16            ,     .,     .,     .,     .,     1,     1,
,          ‡ffffffffffˆffffffffffˆffffffffffˆffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,          ,m         ,Sedentary ,55        ,28            ,     .,     .,     .,     .,     1,     1,
‡ffffffffffˆffffffffffˆffffffffffˆffffffffffˆffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,treadmill ,M         ,Active    ,12        ,19            ,     1,     .,     .,     .,     .,     1,
‡ffffffffff‹ffffffffff‹ffffffffff‹ffffffffff‹ffffffffffffffˆffffffˆffffffˆffffffˆffffffˆffffffˆffffff‰
,All                                                   ,     2,     9,     5,     7,     3,    26,
Šffffffffffffffffffffffffffffffffffffffffffffffffffffff‹ffffff‹ffffff‹ffffff‹ffffff‹ffffff‹ffffffŒ
```

Below, the table was changed to have Flag nested under Sex. It is not obvious in the SAS code, but the ranges for men and have thresholds that differ by 2 units. With table layout, a table that shows men and women side by side, lets that pattern show in the table. It can be seen, in the red box below, that an active 29-year-old with a value of 15 will be classified into normal if male, and high if female. Again, the table should be constructed to make the pattern easy to see.

```
„ffffffffffffffffffffffffffffffffffffff…fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff…fffff†
,                                      ,                               sex                               ,     ,
,                                      ‡fffff…fffffffffffffffffffffff…fffffffffffffffffffffffffffff…fffff‰     ,
,                                      ,     ,           F           ,              M              ,  m  ,     ,
,                                      ‡fffff^fffffffffffffffffffffff^fffffffffffffffffffffffffffff^fffff‰     ,
,                                      ,tflag,         tflag         ,            tflag            ,tflag,     ,
,                                      ‡fffff^fffff…fffff…fffff…fffff^fffff…fffff…fffff…fffff…fffff^fffff‰     ,
,                                      ,loop-,Init-,     ,2_No-,3_Hi-,Init-,     ,2_No-,3_Hi-,loop-,loop-,     ,
,                                      , ing , ial ,1_Low, rm  , gh  , ial ,1_Low, rm  , gh  , ing , ing , All ,
,                                      ‡fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,                                      ,  N  ,  N  ,  N  ,  N  ,  N  ,  N  ,  N  ,  N  ,  N  ,  N  ,  N  ,  N  ,
‡ffffffff…ffffffff…ffffffff…fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,test    ,lifesty-,age     ,value      ,     ,     ,     ,     ,     ,     ,     ,     ,     ,     ,     ,     ,
‡ffffffff‰le      ,        ,           ,     ,     ,     ,     ,     ,     ,     ,     ,     ,     ,     ,     ,
,Biking  ‡ffffffff^ffffffff^fffffffffff‰     ,     ,     ,     ,     ,     ,     ,     ,     ,     ,     ,     ,
,        ,Athletic,29      ,18         ,    .,    1,    .,    .,    .,    .,    .,    .,    .,    .,    .,    1,
‡ffffffff^ffffffff^ffffffff^fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,Treadmi-,        ,30      ,16         ,    1,    .,    .,    .,    .,    .,    .,    .,    .,    .,    .,    1,
,ll      ‡ffffffff^ffffffff^fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,Active  ,12      ,26         ,    .,    .,    .,    .,    .,    .,    .,    .,    1,    .,    .,    1,
,        ,        ‡ffffffff^fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,30      ,.          ,    .,    .,    .,    .,    .,    .,    1,    .,    .,    .,    .,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,16         ,    .,    .,    .,    .,    .,    .,    .,    1,    .,    .,    .,    1,
,        ‡ffffffff^ffffffff^fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,Athletic,29      ,7          ,    .,    .,    1,    .,    .,    .,    .,    .,    .,    .,    .,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,9          ,    .,    .,    1,    .,    .,    .,    .,    .,    .,    .,    .,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,10         ,    .,    .,    1,    .,    .,    .,    .,    .,    .,    .,    .,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,11         ,    .,    .,    .,    1,    .,    .,    .,    .,    .,    .,    .,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,12         ,    .,    .,    .,    1,    .,    .,    .,    .,    .,    .,    .,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,15         ,    .,    .,    .,    1,    .,    .,    .,    .,    1,    .,    .,    2,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,20         ,    .,    .,    .,    .,    1,    .,    .,    .,    .,    .,    .,    1,
,        ‡ffffffff^ffffffff^fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,Sedenta-,8       ,18         ,    .,    .,    .,    .,    .,    .,    2,    .,    .,    .,    .,    2,
,        ,ry      ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,19         ,    .,    .,    .,    .,    .,    .,    1,    .,    .,    .,    .,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,20         ,    .,    .,    .,    .,    .,    .,    1,    .,    .,    .,    .,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,26         ,    .,    .,    .,    .,    .,    .,    .,    1,    .,    .,    .,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,30         ,    .,    .,    .,    .,    .,    .,    .,    .,    1,    .,    .,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,33         ,    .,    .,    .,    .,    .,    .,    .,    .,    1,    .,    .,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,35         ,    .,    .,    .,    .,    .,    .,    .,    .,    1,    .,    .,    1,
,        ,        ‡ffffffff^fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,55      ,11         ,    .,    .,    .,    .,    .,    .,    1,    .,    .,    .,    .,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,28         ,    .,    .,    .,    .,    .,    .,    .,    .,    .,    .,    1,    1,
,        ,        ,        ‡fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,        ,        ,29         ,    .,    .,    .,    .,    .,    .,    .,    .,    1,    .,    .,    1,
,        ‡ffffffff^ffffffff^fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,        ,active  ,30      ,16         ,    .,    .,    .,    .,    .,    .,    .,    .,    .,    1,    .,    1,
‡ffffffff^ffffffff^ffffffff^fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,treadmi-,Active  ,12      ,19         ,     ,     ,     ,     ,     ,     ,     ,     ,     ,     ,     ,     ,
,ll      ,        ,        ,           ,    .,    .,    .,    .,    .,    1,    .,    .,    .,    .,    .,    1,
‡ffffffff‹ffffffff‹ffffffff‹fffffffffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff^fffff‰
,All                                   ,    1,    1,    3,    3,    1,    1,    6,    2,    6,    1,    1,   26,
Šffffffffffffffffffffffffffffffffffffff‹fffff‹fffff‹fffff‹fffff‹fffff‹fffff‹fffff‹fffff‹fffff‹fffff‹fffff‹fffffŒ
```

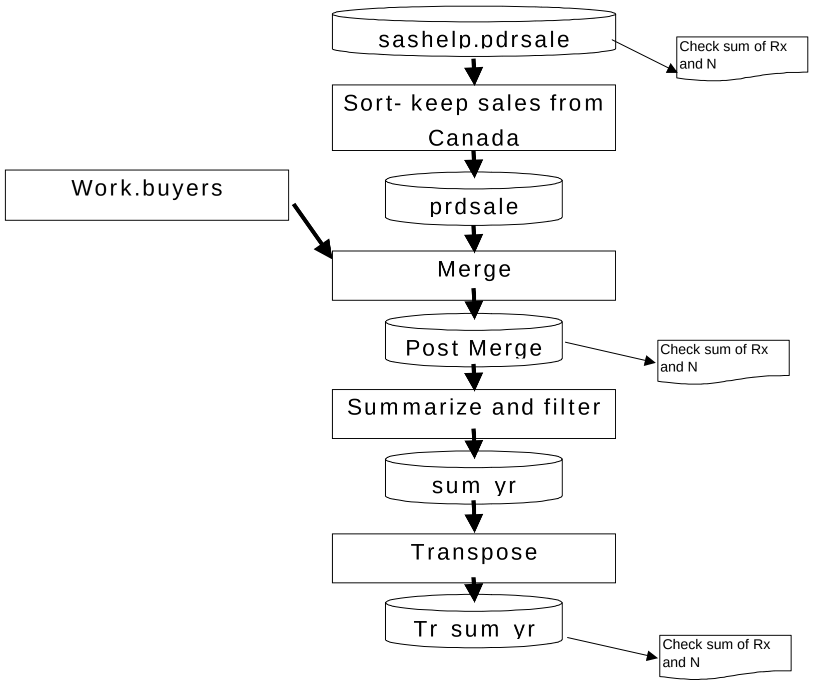
SECTION 2) VALIDATING CODE WITH PROC SUMMARY:

Almost every SAS® programmer was told at the beginning of her/his training to "never lose, or create, an observation without knowing why it happened." With Proc Summary, this instruction can be expanded to (assuming that the programmer is working with a data set containing dollars) to "never change the total dollar value in the date set without knowing why". If a programmer tracks every observation, and every dollar, from the raw data to the final data set, s/he can have a high level of comfort with the answer. This paper uses a simple project to illustrate the technique. Using Proc Summary with the types statement makes this simple to do.

The technique that is described in this paper is simple. It consists of running Proc Summary, with the "appropriate" types statement, on the raw dataset and again whenever the dataset is subset or merged. This paper will use the dataset sashelp.prdsales, and a data set I created, to illustrate this technique.

PROBLEMS STATEMENT/BUSINESS TASK

Imagine that the client wants a report on Canadian sales from the data set sashelp.prdsale. Specifically, the client wants a report on Canadian sales activity broken out by the person responsible for buying the type of goods (called the Buyer). A flowchart of the process is shown below. We will check the totals in the file after each subset and merge. Most often we will use a summary to check the totals.



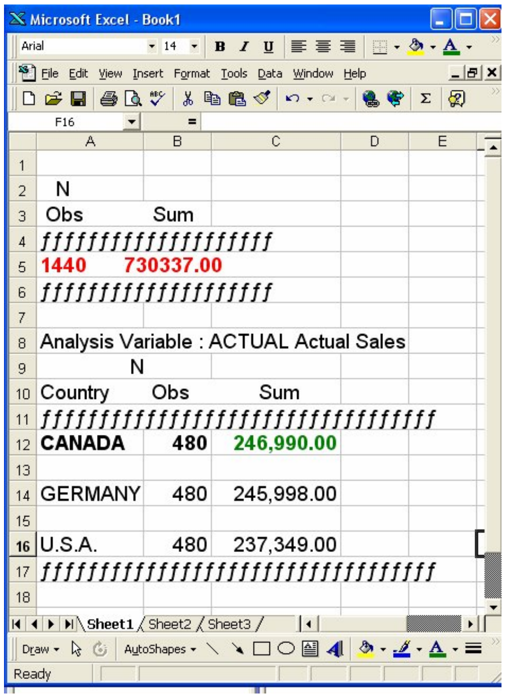
Start out with a Proc Summary on the raw dataset. A convenient trick is to paste the information from the listing and log into an XLS sheet. This allows the programmer to keep from having to remember long numbers. It also allows a programmer to do calculations easily.

```
%macro skip;
*initial summary on raw data;
Proc summary data=sashelp.prdsale
             missing print sum;
 class country ;
 var actual;
 types () country;
%mend skip;
```

NOTE: There were 1440 observations read from the data set SASHELP.PRDSALE.

The log shows that the data set has 1440 and the red Proc Summary output shows that I have accounted for every one. Canadian totals are 480 obs and 246,990 dollars. The programmer now knows s/he must keep track of 246,990 dollars in sales as the program progresses.

The log says Proc Summary read 1440 obs and the Proc Summary output says it read 1440 obs.



```
 N
Obs       Sum
ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
1440      730337.00
ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

Analysis Variable : ACTUAL Actual Sales
            N
Country      Obs          Sum
ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
CANADA       480   246,990.00

GERMANY      480   245,998.00

U.S.A.       480   237,349.00
ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
```

Proceed with the program. Bring in the names of the “buyers” and the products that they buy.

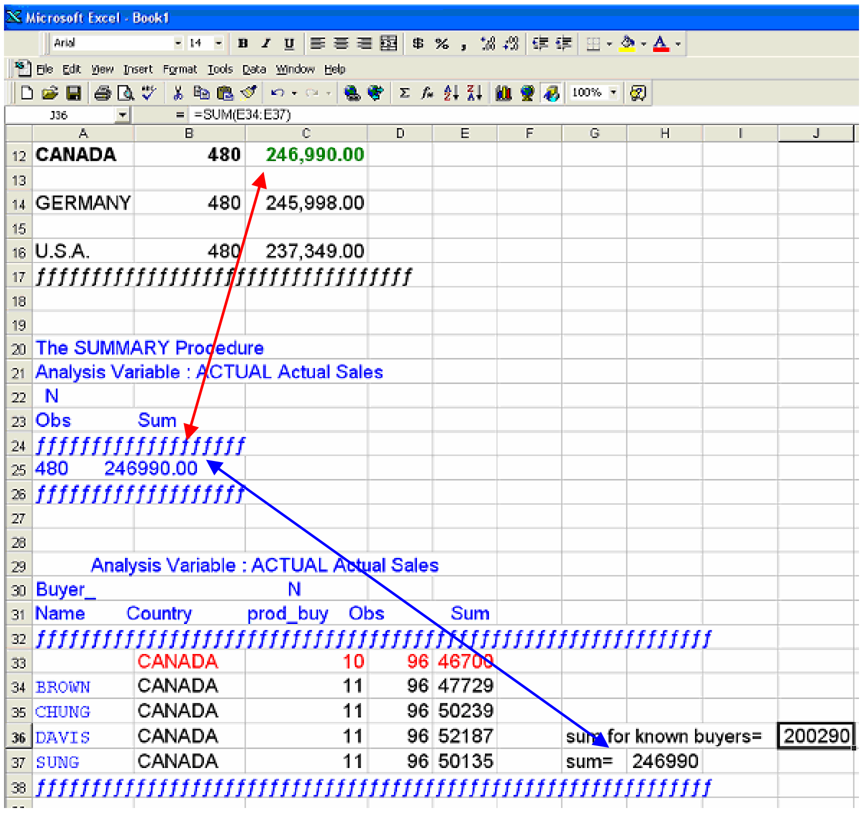
```
DATA BUYERS;
INFILE DATALINES missover;
INPUT @1 Buyer_Name $CHAR8. @10 PRODUCT $CHAR10.;
DATALINES;  /*NOTE no buyer for TABLE*/
BROWN    BED
CHUNG    CHAIR
DAVIS    DESK
SUNG     SOFA
; run;

proc sort data=buyers;
by product; run;


Proc sort data=sashelp.prdsale out=prdsale;
by product;
where country="CANADA"; run;  /*Get the Canadian Sales*/

data post_merge;
merge prdsale(in=p) BUYERS(in=B);/*Merge in the buyer information*/
by product;;
prod_buy=P*10 + B*1; /*this is a variable that can be used to monitor the merge*/
run;

%macro Skip;  /*Use proc summary to check n and sum after the merge*/
/*Use the skip macro trick to comment out the code but keep it in the program*/
Proc summary data=post_merge missing print sum;
Class Buyer_Name country prod_buy;    /*Check the total after a merge*/
var actual;
types () country*prod_buy*Buyer_Name; run;
%mend skip;
```

After a subset or merge we submit a Proc Summary. The types statement lets a programmer get overall and detail dollar summaries in one Proc Summary. Types and Ways statements are powerful.

I pasted results into Excel as a convenience. I reformatted the data into SAS font and did a text to columns conversion.

The red arrow points to the result for type () and shows that the subset of the data we are working with still contains has every dollar for Canadian sales.

The bottom of the blue arrow points to the result of a summation done after pasting the listing into Excel.. The blue arrow shows that we are accounting for all the dollars after we merged in buyer information. However, the red row, in the screenprint has a missing buyer and indicates that our data step did not "find" a match for one product.



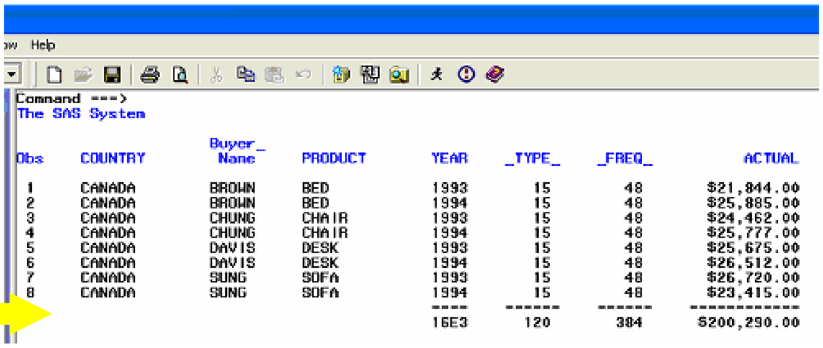
```
J36   = =SUM(E34:E37)
```

	A	B	C	D	E	F	G	H	I	J
12	**CANADA**	**480**	**246,990.00**							
13										
14	GERMANY	480	245,998.00							
15										
16	U.S.A.	480	237,349.00							
17	ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ									
18										
19										
20	The SUMMARY Procedure									
21	Analysis Variable : ACTUAL Actual Sales									
22	N									
23	Obs	Sum								
24	ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ									
25	480	246990.00								
26	ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ									
27										
28										
29	Analysis Variable : ACTUAL Actual Sales									
30	Buyer_		N							
31	Name	Country	prod_buy	Obs	Sum					
32	ƒƒ									
33		CANADA	10	96	46700					
34	BROWN	CANADA	11	96	47729					
35	CHUNG	CANADA	11	96	50239					
36	DAVIS	CANADA	11	96	52187		sum for known buyers=			200290
37	SUNG	CANADA	11	96	50135		sum=	246990		
38	ƒƒ									

IF WE ARE ONLY GOING TO REPORT ON KNOWN BUYERS, WE WILL PASS ON $200,290 DOLLARS AND 96*4=384 OBS.

Since we only want to report where we have buyer information. We summarize and then filter on the variable we used to monitor the merge. Proc Print with the sum statement can show us what the data set looks like and allow us to keep track of the total dollars.

```
Proc summary
data=post_merge;
class country Buyer_Name
product year; output
out=sum_yr sum(actual)=;
ways 4;
where prod_buy=11;
run;

%macro skip;
Proc print data=sum_yr;
sum _numeric_    ;
run;
%mend skip;
```



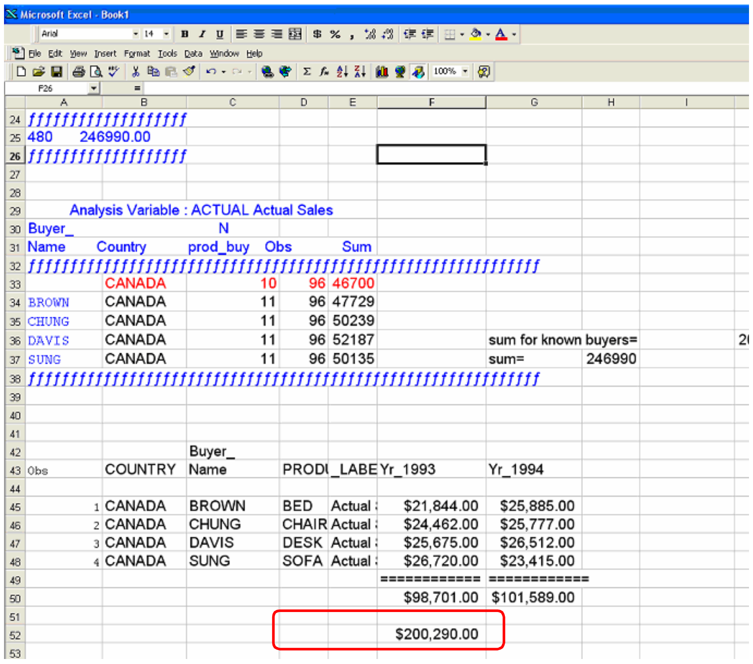
```
Command ===>
The SAS System

                      Buyer_
Obs    COUNTRY         Name     PRODUCT        YEAR    _TYPE_    _FREQ_          ACTUAL

 1     CANADA         BROWN     BED            1993        15        48      $21,844.00
 2     CANADA         BROWN     BED            1994        15        48      $25,885.00
 3     CANADA         CHUNG     CHAIR          1993        15        48      $24,462.00
 4     CANADA         CHUNG     CHAIR          1994        15        48      $25,777.00
 5     CANADA         DAVIS     DESK           1993        15        48      $25,675.00
 6     CANADA         DAVIS     DESK           1994        15        48      $26,512.00
 7     CANADA         SUNG      SOFA           1993        15        48      $26,720.00
 8     CANADA         SUNG      SOFA           1994        15        48      $23,415.00
                                               ----    ------    ------    ------------
                                               16E3       120       384     $200,290.00
```

We need to transpose the file to get the report to look the way the client wants it. That is shown below.

```
proc transpose data=sum_yr
     out=TR_sum_yr(drop=_name_)
         prefix=Yr_;
         var actual;
         id year;

by country Buyer_Name product;
         run;

proc print data=TR_sum_yr;
sum _numeric_;
run;
```

I pasted the listing into Excel and used Excel to check that the column totals still add to $200,290.

The report is now done and we have kept track of every dollar as it passed through the program. There can be high confidence in that the report is correct.



	A	B	C	D	E	F	G	H	I
24	ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ								
25	480	246990.00							
26	ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ								
27									
28									
29	Analysis Variable : ACTUAL Actual Sales								
30	Buyer_		N						
31	Name	Country	prod_buy	Obs	Sum				
32	ƒƒ								
33		CANADA	10	96	46700				
34	BROWN	CANADA	11	96	47729				
35	CHUNG	CANADA	11	96	50239				
36	DAVIS	CANADA	11	96	52187		sum for known buyers=		2
37	SUNG	CANADA	11	96	50135		sum=	246990	
38	ƒƒ								
39									
40									
41									
42			Buyer_						
43	Obs	COUNTRY	Name	PRODU	_LABE	Yr_1993	Yr_1994		
44									
45	1	CANADA	BROWN	BED	Actual :	$21,844.00	$25,885.00		
46	2	CANADA	CHUNG	CHAIR	Actual :	$24,462.00	$25,777.00		
47	3	CANADA	DAVIS	DESK	Actual :	$25,675.00	$26,512.00		
48	4	CANADA	SUNG	SOFA	Actual :	$26,720.00	$23,415.00		
49						============	============		
50						$98,701.00	$101,589.00		
51									
52						$200,290.00			
53									

CONCLUSION

Good Programming Practice requires that a programmer keep track of every observations and totals of important variables as they write programs.   Proc Tabulate and Proc Summary are both spowerful QC tools. They allow programmers to keep track of observations and variable totals as a program is being written..

CONTACT INFORMATION

Your comments and questions are valued and encouraged. You can contact the author at:

Russell Lavery, Independent Contractor   Email: russ.lavery@verizon.net