Paper TU-08

An Animated Guide©: Proc Report: The file behind the scenes

Russell Lavery, Contractor, Ardmore, PA

ABSTRACT

Proc report builds a Temporary Internal File (TIF) behind the scenes and uses that file, and a Report Data Vector (RDV), to compute new variables.

Understanding the creation of the Temporary Internal File, and the flow of data through the Report Data Vector, makes programming and debugging easier. The option Output=out, on the Proc Report line, is the key to investigating and understanding Proc Report. It writes the TIF to a regular SAS file.

This presentation does NOT cover formatting but will focus on:

1) The creation of the temporary internal table.
2) Relating the temporary internal table to syntax and output.
3) How proc report uses the report data vector.
4) Breaks and Rbreaks.
5) Differences among statements like: Compute before/after var Vs. Compute Before/After Vs. Compute
6) Data variables vs. Report Variables and using the automatic retain in Proc Report

INTRODUCTION

Understanding the full power of Proc Report is been made more difficult because the "automatic and behind the scenes processes" have received little attention.

This paper examines three major "behind the scenes" process for Proc Report: 1) the creation of a Temporary Internal File (TIF), 2) the use of the Report Data Vector (RDV) and 3) the use of the Data Variable Table (DVT) . This paper will show the relationships between these processes and make explicit the timing order of calculations done by Proc Report.

Finally, it will relate the timing of calculations to problems/unexpected results. It is hoped that understanding these processes will ease debugging and make some of the advanced features of this useful report writer more understandable.

A REPRESENTATION OF PROC REPORT

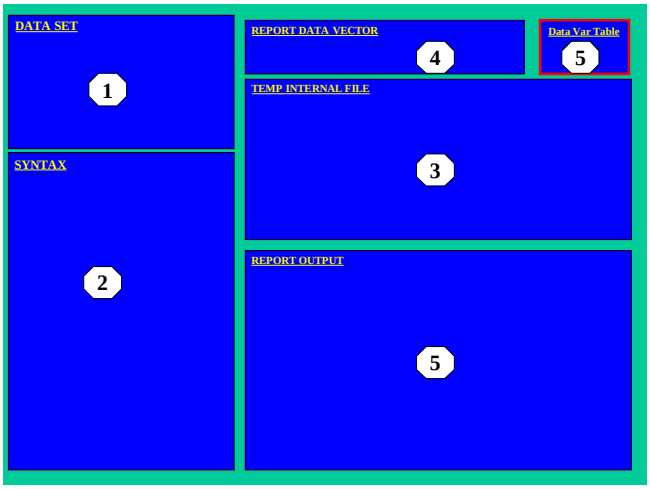

Figure 1

A graphical representation is useful for describing the underlying process. Figure 1 shows a structure for thinking about the process of SAS Proc Report.

To understand Proc Report, one must consider (numbers refer to sections of Figure 1):

(1) the original SAS data set.
(2) the syntax that the programmer writes
(3) the Temporary Internal File (TIF) of summary information (imagine it similar to a data set produced by a Proc Summary)
(4) the Report Data Vector (consider it similar in function to the program data vector)
(5) the Data Variable Table (a special memory-resident file that holds data variables -similar to the macro symbol table)
(6) the report output (on the SAS listing file).

All these can be conveniently (though with small fonts) represented on the graphic above.

THE PROC REPORT PROCESS

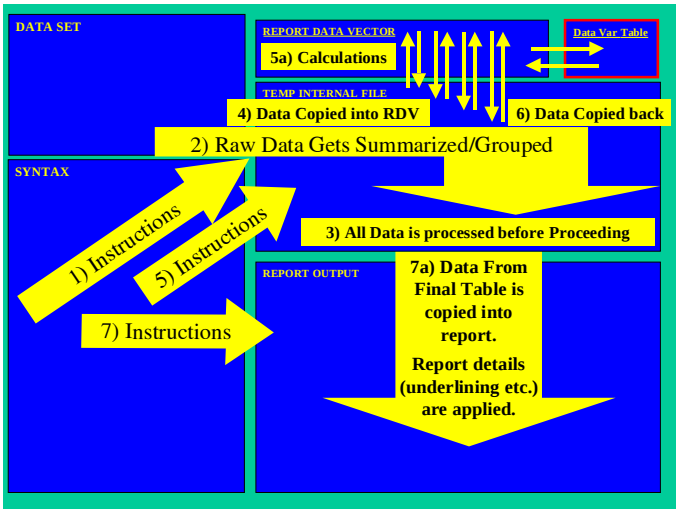

Figure 2

The Proc Report process has several steps and Figure 2 provides an overview of the steps. SAS goes to the program syntax (1) for instructions on how to create thestructure of the Temporary Internal File **(TIF)** (what variables to include, what summaries to perform etc.). In order to have a variable in the Temporary Internal File, it must be mentioned in a "column statement" in the Proc Report Syntax.

The TIF is not a copy of the raw data set. You need not bring all the variables in the data set into the Temporary Internal File and Proc Report can create variables that are not in the input data set. SAS then processes all of the raw data set into the Temporary Internal File (TIF) (2). If the syntax included instructions for creating summary statistics, they would be created at this time. This complete pre-processing of the source file (3), combined with storing of summary statistics in the TIF is the reason why summary statistics are "available" to the first line of Proc Report output. The whole external file is processed in this step, and never accessed again. After the TIF is created, all further processing uses the TIF.

One of the features of Proc Report is the ability to create new variables that were not in the original data set. Proc Report creates new variables with a process very similar to that used by the data

step- a data vector.

Assume that, after processing the data file, the TIF has many lines of data. Lines of data are copied (4), one line at a time, into the Report Data Vector **(RDV)**. The RDV is like a one line Excel spreadsheet and all calculations are performed there. Instructions in your syntax (5), and the structure of the TIF, control what/when calculations take place (5a).

Most SAS assignment functions are allowed in Proc Report-with one limitation. If you imagine variables are in columns, calculations must take place from left to right. When creating/modifying variable you can only use as input variables that are in columns to the left of the variable being created/modified. SAS moves from left to right across the variables in the RDV and executes assignment statements that are required to create or modify those variables. The programmer must arrange variables in the column statement so that calculations can be performed.

After calculations are completed, the values are then copied (6) from the RDV back to the TIF. SAS then clears the RDV and copies another line of data into the RDV. Variables in the RDV are set to missing before a new line of data is copied into the RDV. This automatic resetting to missing is a feature of Proc Report, and occasionally a problem.

Often, a programmer will want to make a value available for use in calculations that must be performed on every row of the TIF, in effect retaining a variable value across rows. This "retaining" is often used in calculating percentages, where the denominator for the calculation must be the same for all rows of data. To "retain" a variable's value and use it on other lines in the TIF, the value must be copied out of the RDV into the Data Variable Table (DVT) (A.K.A. the Report Symbol Table). The DVT is a memory area that simply holds variable names and their values. It is conceptually similar to the macro symbol table. Values in the DVT are not automatically set to missing as new observations are processed in the DVT. The DVT is shown in Figures 1 and 2 but will not be shown in examples where it does not play a part. This is to allow use of larger fonts and more readable graphics.

7) and 7a) When all of the data has been processed through the RDV, and copied back to the TIF, SAS uses the TIF and instructions in the syntax to determine what to do as it copies information from the TIF to the SAS listing. Often, instructions will specify formats to be applied, column widths, column headings and the like. It is not required that all the rows in the TIF be printed in the listing.

The best method of explaining Proc Report process is to load the graphic in Figure 1 with a small report and examine the details. It should be remembered that the graphic in this paper, as opposed to the animations in the presentation, is not an accurate representation of the system. For example, Figure 3 shows activity in the data set, the TIF, the RDV and report output in the SAS listing. All these do not occur at the same time but are shown in one graphic to meet space limitations on the paper.

The rules that one must consider when programming Proc Report are listed below. This paper will proceed from simple to complex reports to illustrate these rules. Rather than study the rules (as a set of rules), it is suggested that the rules be compared to their usage as shown in Figures 3, 4 and 5. Not all rules are illustrated in any one example.

Important Proc Report Rules:
The column statement determines the number of columns in the TIF. Syntax determines the number of rows in the TIF.

Define statements assign characteristics to variables (group, order, format, width, title, group, order, across, sum, NOPRINT).

Rbreak Before/After, or Compute Before/After statements create report level summary lines in the Temp Internal File. These lines will be at the top or the bottom of the TIF

Compute Before/After Varname, create a summary line in the TIF when levels of the variable change (order or grouping is advised). These lines will be in the middle of the TIF, just before/after the variable changes value.

Breaks, and Rbreaks, create summary lines in the Temp Internal Table but no summary information is printed unless a break line includes the summarize option.

Compute before/after, create summary lines in the Temp Internal Table but nothing is printed unless the programmer also specifies a same-level break with summarize option. Note: that Compute before/after code does not, by itself, produce output.

Columns in the TIF contain the same type of information at different levels of summarization.
IF YOU WANT CONTINUED ACCESS TO A PARTICULAR LEVEL OF SUMMARIZATION, CODE IN A "COMPUTE BEFORE" statement and copy the information you want to "retain or continue to access" INTO ANOTHER (a Data) VARIABLE.

EXAMPLE 1:

(Illustrating: order, sum, rbreak before)
Please note the out= syntax (A) in Figure 3 that has made this paper possible. If the reader wishes to experiment with Proc Report, or to debug a problem report, this option is invaluable.

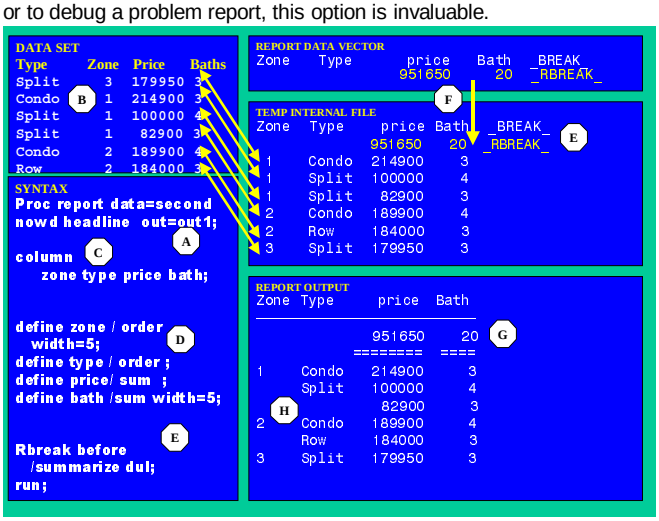

Figure 3 (shown enlarged at end)

In Figure 3, the complete data set (B) is shown in the upper left corner of the graphic. Note that the order of the variables in the source data set is not the same as it is in the column statement (C). The column statement is used to bring variables from the source data set into the TIF and to set the left-to-right order for variables the TIF and SAS listing. In order to have a variable in the TIF, it must be in the column statement. In summary, the TIF contains variables in the column statement – in the order in which they appear in the column statement (C).

If you are creating a new variable inside the Proc Report itself – and want it in the TIF and SAS listing the name of the variable must be in the column statement. This implies that you can have variables in the column statement that are not in the source data set.

Very importantly, the TIF also contains one extra column,

automatically created by SAS, that contains information about the "break level of the data". A column in the TIF can contain data on different levels of summarization and SAS uses the __BREAK__ column to "keep track" of the level of summarization. This is similar to the way Proc Summary uses the __type__ variable. The first row of the TIF, in Figure 3, contains price and number of bath information, summarized for the whole data set. The other rows of the TIF contain price bathroom information on individual houses.

The define statements (D) provide instructions on how the individual variables are to be handled. Zone and type are defined as "ORDER". This will cause rows in the TIF, and SAS listing, will be sorted/ordered by these variables. Choices for how variables are to be handled are: order, group, computed, display, sum, min, max and other algebraic functions. If variables are defined as group, the TIF and the listing will only contain one line for every level of the grouping variable(s). This is similar to grouping in SQL and class variables in Proc Summary.

If the variable is not in the source data set, it must be handled as "computed" and must have some compute syntax (not yet seen, or defined) associated with it. Display tells Proc Report to simply list the values. Arithmetic functions (sum, min, mean, etc.) tell SAS how to how to handle that variable on summary lines (group and break lines). In Figure 3 , price and path are defined as sum. Whenever a line in the TIF is holding information from multiple lines in the input data set, these variables will be summed.

Since there are no variables defined as group in Figure 3, the data in the original data set will be copied to the TIF, line by line. This will produce a simple listing of the data in the SAS data set.

Note that zone is specified to be 5 columns wide in the listing and type will use the default width. There is a Rbreak Before line that produces a summary information on all the observations in the data set. Price and bath, on this line in the TIF, are the sum of all the prices and baths in the data set. (951,650 dollars and 20 bathrooms).

The __break__ column (E) in the TIF, and the value of __RBREAK__ (E) were created by the Rbreak Before statement (E). There are 4 types of break statements and they are the keys to creating summary statistics. Think of the command "Rbreak Before" as being a shorthand for "Report Level Break Before the detail data". You can break after the whole report (Rbreak After) or before the whole report (Rbreak Before) as in (E). These two commands give access to summary information on the whole data set. You can also issue a command Break Before/After variable_name. This would create summarizations in the TIF before/after the variable changes its value.

Break commands create lines in the TIF but do not produce any output in the SAS listing *unless* the option /summarize is specified as an option on the break command. This two command process is useful because programmers sometimes want to create summary lines in the TIF (so that summary information can be "accessed and retained") but do not want the rows containing summary information printed in the listing.

When the syntax executes, lines of data are copied from the TIF to the RDV and any calculations, specified in the syntax, happen in the RDV. Any calculations are performed in the order in which the variables appear in the column statement (reading left to right). Variables must have been "read by SAS" before they can be used as input in a calculation. Accordingly the following syntax would be processed successfully. When SAS attempts to calculate doll_bath it "knows" the values for price and bath.

Column zone type price bath **Bath_doll**;
Define bath_doll/computed;
Compute bath_doll;
bath_doll =bath/price;
Endcomp;

This syntax below would not be processed successfully.
When SAS attempts to calculate doll_bath it "does not know" the values for price or bath;

Column zone type **bath_doll** price bath;
Define bath_doll /computed;
Compute bath_doll;
bath_doll =bath/price;
Endcomp;

EXAMPLE 2:

(illustrating: Break after varname , compute after statement).

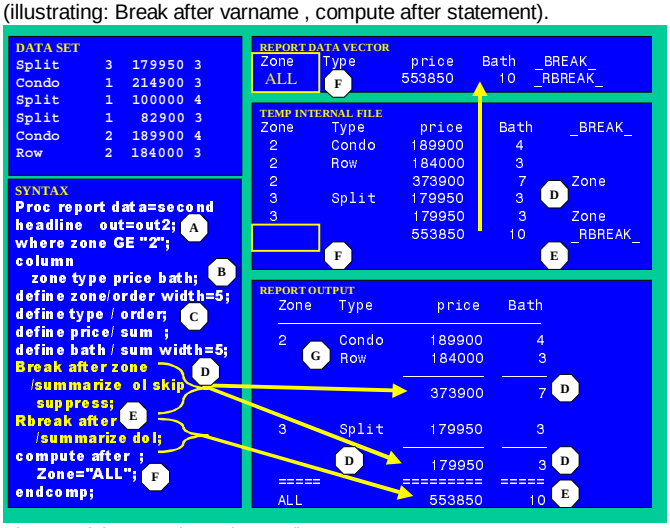

Figure 4 (shown enlarged at end)

To make output fit on the slide in Figure 4, a where clause was used (A). The column statement (B) specifies the variables that are in the TIF and their left-to-right order. Zone and Type are defined as order (C) and this determines the top-to-bottom order of the TIF and SAS listing. There is a command to break after zone (D) that produces a summary line in the TIF every time zone changes value. Since price and bath (C) are type SUM, summed values appear on the break-after lines (D). The summarize option causes the break-after lines to print in the SAS listing. Additional options specify skipping a line after zone changes value, a single OverLine when zone changes value and a Double OverLine over information produced by the Rbreak After statement. The Suppress option causes Proc Report to suppress the printing of the value of the zone on the break line.

The compute after statement (F) executes when the last observation passes through the RDV. In Figure 4, all the observations, except one, in the TIF have been processed through the RDV. The last observation is shown still in the RDV, about to be transferred back to the TIF. After all observations in the TIF have been processes through the RDV, data will flow from the TIF to the listing. In Figure 4, the "compute after statements" (F) have executed and Zone now has the value of "ALL";

The compute after statement is "linked" to the Rbreak After. Either would produce a row in the TIF, but they do not automatically cause information to be written to the SAS listing. This allows programmers to create summary level information to be used in calculations without having to have that information on the SAS listing.

The final report is shown in the bottom right hand corner. Note that the summarize option (E) has caused the summary information to be printed. The DUL option has produced Double UnderLines in the listing. Because there were no grouping variables, the raw data (rather than grouped data) is presented in the SAS listing.

Finally, note (G) that Proc Report will automatically suppress the repeated printing of ordered and grouped values. This makes for a cleaner looking report, but sometimes causes confusion.

EXAMPLE 3:

(Illustrating: grouping observations)

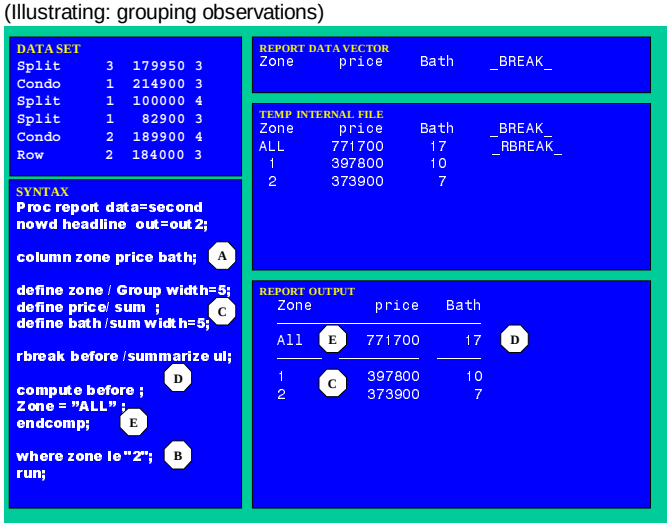

Figure 5 (shown enlarged at end)

Figure 5 shows the state of the system when the Proc Report has finished writing to the SAS listing and is about to delete the TIF. Figure 5 uses the column statement (A) to create zone, price, and bath in the TIF. Note the where statement (B), on the variable zone, is used to filter out observations.

Zone is defined as group (C) and SAS produces one observation in the TIF for each level of Zone that meets the where criteria. The Rbreak Before (D) creates a line in the TIF and the summarize option on that statement causes the line to print in the listing. The Compute Before statement (E) only executes when the first line of the TIF is in the RDV. As in Figure 4, it changes the value of zone to "All".

EXAMPLE 4 (Figure is shown enlarged at end of Paper)

(Illustrating: timing of calculations)

This complex example illustrates the timing of different operations, especially the different types of computes. It also shows the compute after/before statements, and compute after/before variable_name statements, creating lines in the TIF.

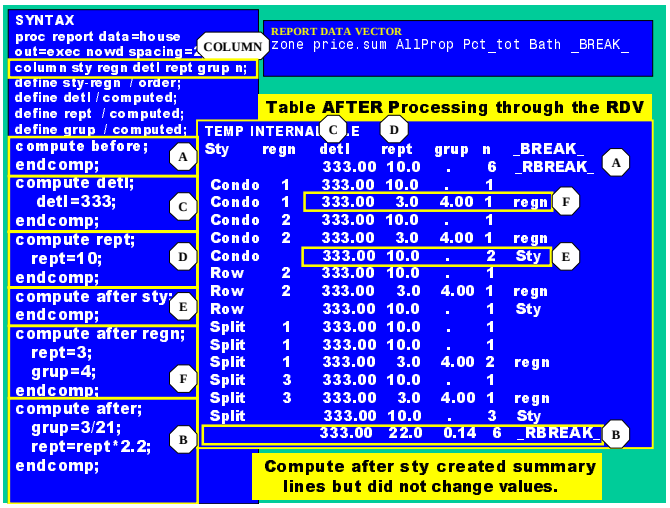

Figure 6 (shown enlarged at end)

First, the column and define parts of the syntax execute and Proc Report creates the shell of the TIF. Secondly raw data is processed/summarized/loaded into the TIF. Finally, compute statements start to execute..

There are three general types of compute statements. A simple compute statement is an instruction to compute a value for a variable (C) (D) as each line of the TIF passes through the RDV. A second type of compute statement is an instruction to compute a value for variable when a variable "breaks" (E) (F) or changes value (compute before/after var_name). A third type of compute statement (compute before/after) is an instruction to compute a value for a variable and also modify summary information that describes the whole data set (A) (B).

As processing starts, the first line from the TIF is loaded into the PDV. The first compute statement to execute is the compute associated with the left-most variable in the column statement. Sty and regn are in Compute After statements and do not trigger execution. The compute detl statements (C) execute and detl is assigned a value of 333. Next the compute rept statements (D) execute and rept is assigned a value of 10. Then the compute before statements (A) execute. They have no instructions to the RDV. Here, the compute before statements only create a line in the TIF. Since they are not associated with a summarize option, these statements do not produce output in the listing. These statements were included simply to show that a compute before statement creates a row in the TIF.

We should examine the processes when the third observation (F) from the TIF enters the RDV. First detl (C) is set to 333. Then rept (D) is set to a value of 10. Then, the break after statements (F) for regn execute and assign new values to rept and grup. Compute after statements execute after compute var_name statements. The fact that the value of rept, in the TIF, is 3 indicates the order of execution of the compute commands. The compute var_name (compute rept) executed first and set the value of rept to 10. Then the "compute after region executed" and re-set the value of rept to 4.

We should examine the processes when the sixth observation (E) from the TIF enters the RDV. First detl (C) is set to 333. Then rept (D) is set to a value of 10. Then, the "compute after sty" statements (E) execute but have no instructions. The "compute after varname" statements create lines in the TIF. The set of nonsense statements (E) was included for the sole purpose of just showing that the compute after statement will create a line in the TIF.

The last line in the TIF offers strong evidence on timing. When the last observation (B) from the TIF enters the RDV some interesting

things happen. First detl (C) is set to 333. Then rept (D) is set to a value of 10. Then, the compute after statements (B) execute. Grup is set to .14 in a not very interesting hard-coded division.

However, thinking about the value of rept (B) is illuminating. The only way that the rept assignment formula (rept=rept*2.2) could return a value of 22 is if rept had a value of 10 before the assign formula executed. Rept was set to 10 by the compute rept statements (D). That value of 10 was used as input in the "compute after" statement. The result of 22 is offered as proof that the "compute rept" statements executed before the "compute after" statement. The general timing of computes is left to right in the order that the variables appear in the column statement. Associated with that timing rule is a rule that "compute varname" statements execute before "compute before/after".

EXAMPLE 5:

(Illustrating: Data Variables or retaining information across rows).

Figures 1 and 2 showed the Data Variable Table (or Report symbol table) with a red border. The Data Variable Table has not been shown in other figures to allow the use of larger fonts. It is now time to consider the Data Variable Table, because it is that table that allows us to retain values across rows in the TIF.

A data variable is defined as a variable that has been loaded from the RDV into the Data Variable Table. Data variables use is considered one of the more confusing parts of Proc Report because of how Data Variables are created. Data Variables are not created by a programmer typing one line of code but by the interaction of several lines of code- lines of code that also have other functions. This interaction of commands is what has made Data Variables an advanced topic.

It is time to review several facts about Proc Report. Remember that the column statement was used to create the TIF. Variables in the column statement may, or may not, have compute statements associated with them. If a variable is in the column statement, regardless if it is has an associated compute statement, it will be in the TIF and in the RDV… and *it will be reset to missing* before each new observation enters the RDV.

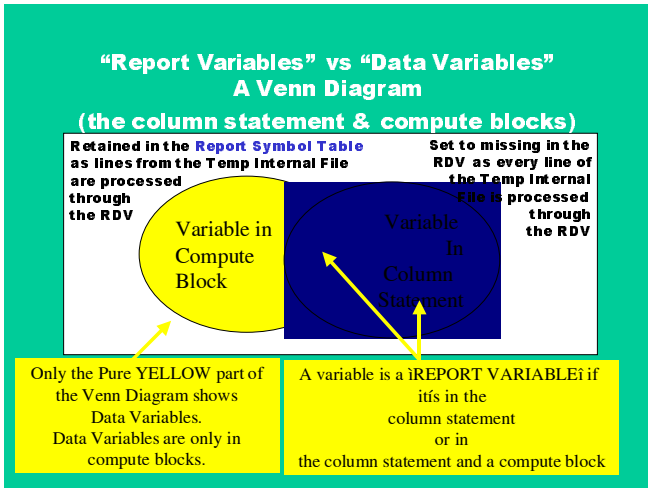

Figure 7

If a variable is in a compute statement, *and only in a compute statement*, it will not be created in the TIF or RDV. It will be created in a special table- the Data Variable Table (AKA the Report Symbol Table). The Data Variable Table is much like the macro symbol table. Variables in the Data Variable Table are not automatically reset to missing as observations are processed through the RDV. The Venn diagram in Figure 7 illustrates this logic of Report Variables and Data Variables.

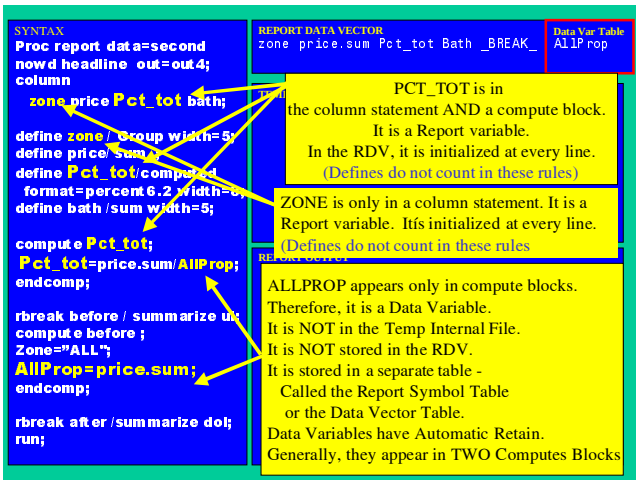

Figure 8 (shown enlarged at end)

Figure 8 shows a section of SAS code that creates Report Variables and a Data Variable. The reader is encouraged to work through the text in the yellow boxes, and the syntax in Figure 9, to increase his/her understanding.

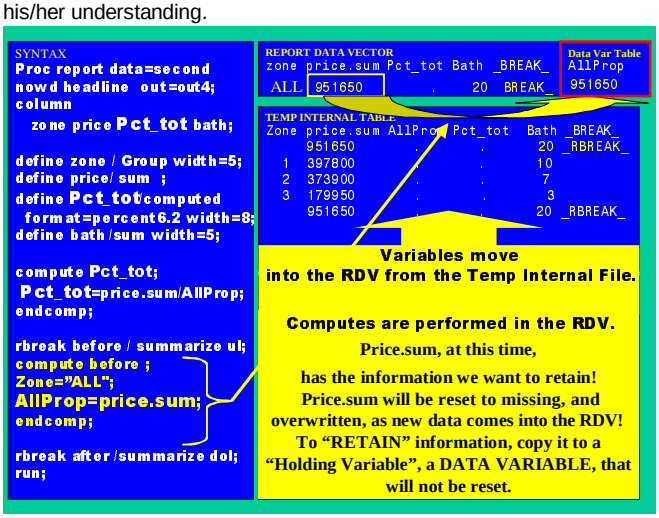

Figure 9 (shown enlarged at end)

Figure 9 illustrates the process of creating a Data Variable. This code is going to calculate the percentage that each line in the TIF represents of the total, for the variable price, in the data set. Price is a sum variable and is referenced with a two part name (price.sum).

When the "Compute Before" executes (as seen in Figure 9), price, in the RDV, is holding the value we need for the divisor. Before the second observation enters the RDV, the value of price.sum will be initialized to missing and the information will be lost to use- unless action is taken.

The solution is to copy the information we want to retain out of the RDV into a different memory section- one that does not automatically get updated as new observations are processed. This copying is what is shown in Figure 9. The trick for retaining information/variables is: when the information you wish to retain is in the RDV, copy it to a new variable – a variable that is never mentioned in the column statement. This is how desired information is copied out of the RDV and into the Data Variable Table (or Report Symbol Table).

Use of the value stored in the Data Variable Table is shown in Figure 10. As we discussed above, "compute varname" statements execute as every line from the TIF passes through the RDV. In Figure 10 we see the percent calculation happening as the compute Pct_tot statements execute.

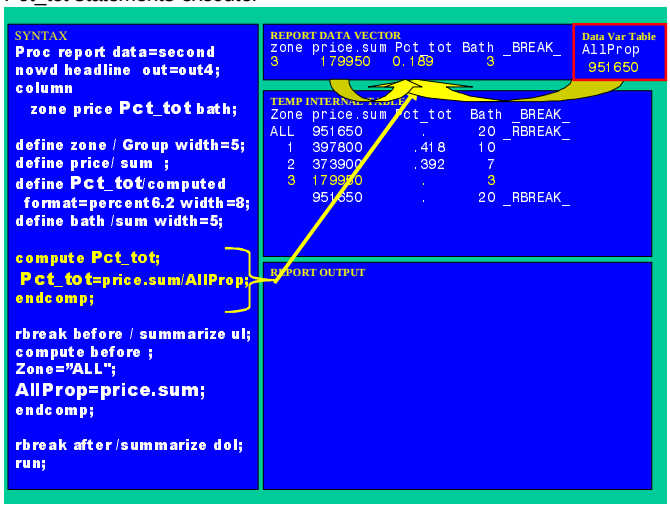

Figure 10 (shown enlarged at end)

It is interesting to note that the value of Pct_tot is missing in the first row of the TIF. That is because "compute varname" statements execute before "compute before" statements. The sequence of operations that resulted in this missing value for Pct_tot follows. The first observation moved from the TIF into the RDV. The "Compute Pct_tot" statements executed but allprop was missing and the result of the assignment was a missing value. Then the "Compute Before" statements executed. Zone was set to ALL and the value of price.sum was copied into allprop.

CONCLUSION

Proc Report is a powerful tool for generating complex reports. The understanding of the Temporary Internal File that Proc Report creates is useful for creating complex reports and for debugging.

Use of the outfile=option is key to understanding proc report.

ACKNOWLEDGMENTS (HEADER 1)

Thanks to Sandy MCNeill of SAS Institute for sharing her time and understanding.of this material.

CONTACT INFORMATION (HEADER 1)

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

Russell Lavery
Independent Contractor
9 Station Ave. Apt 1
Ardmore, PA 19003
610-645-0735 # 3
Email: russell.lavery@att.net

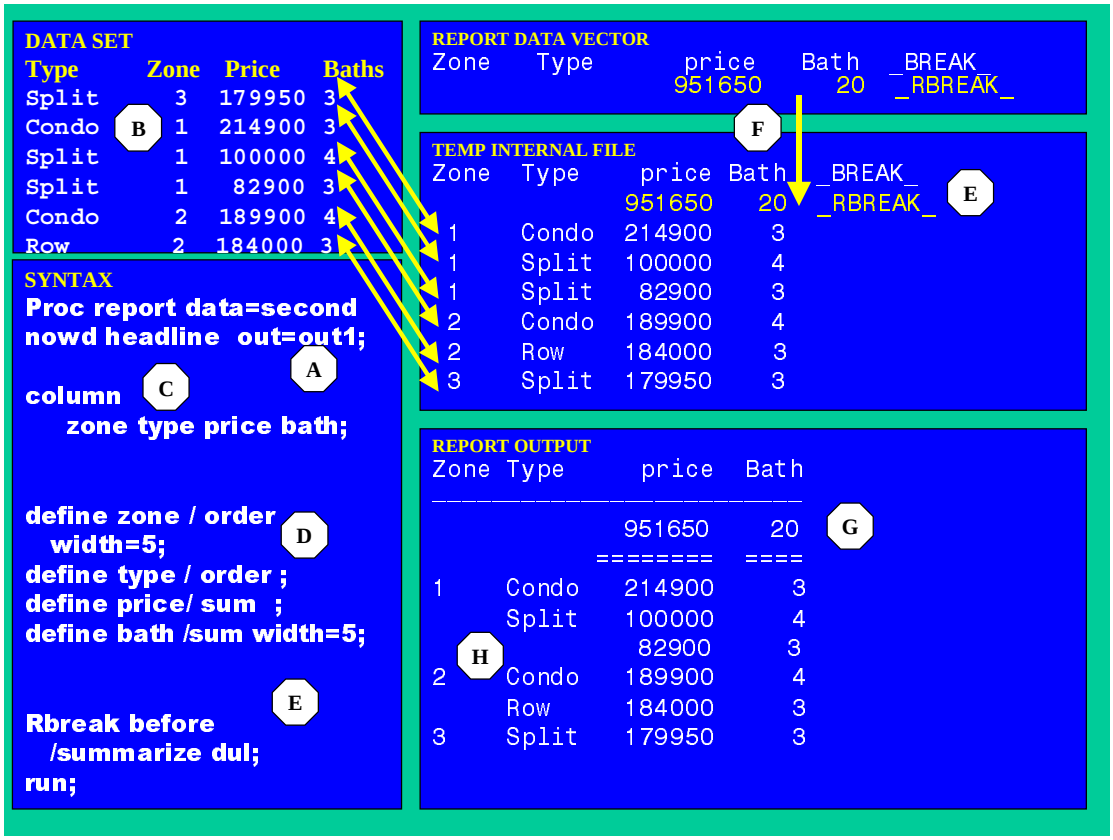

Figure 3

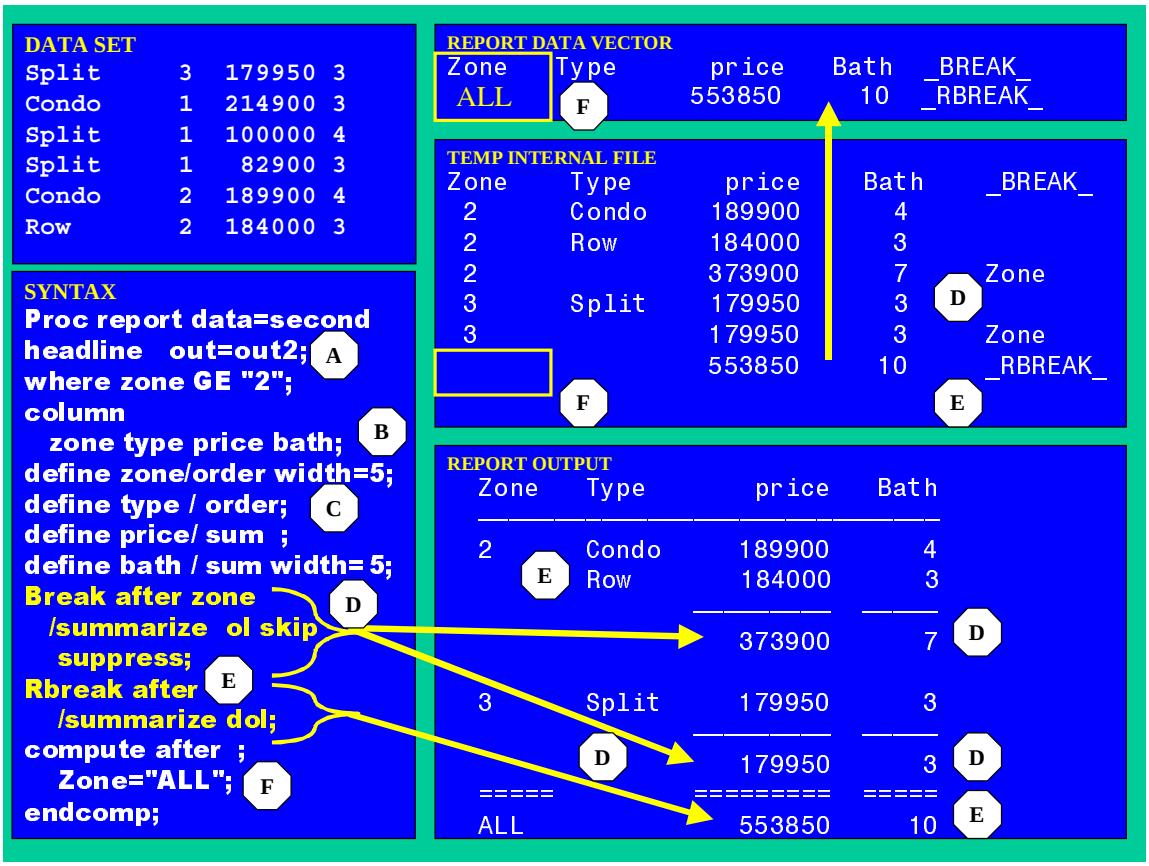

Figure 4

Figure 5

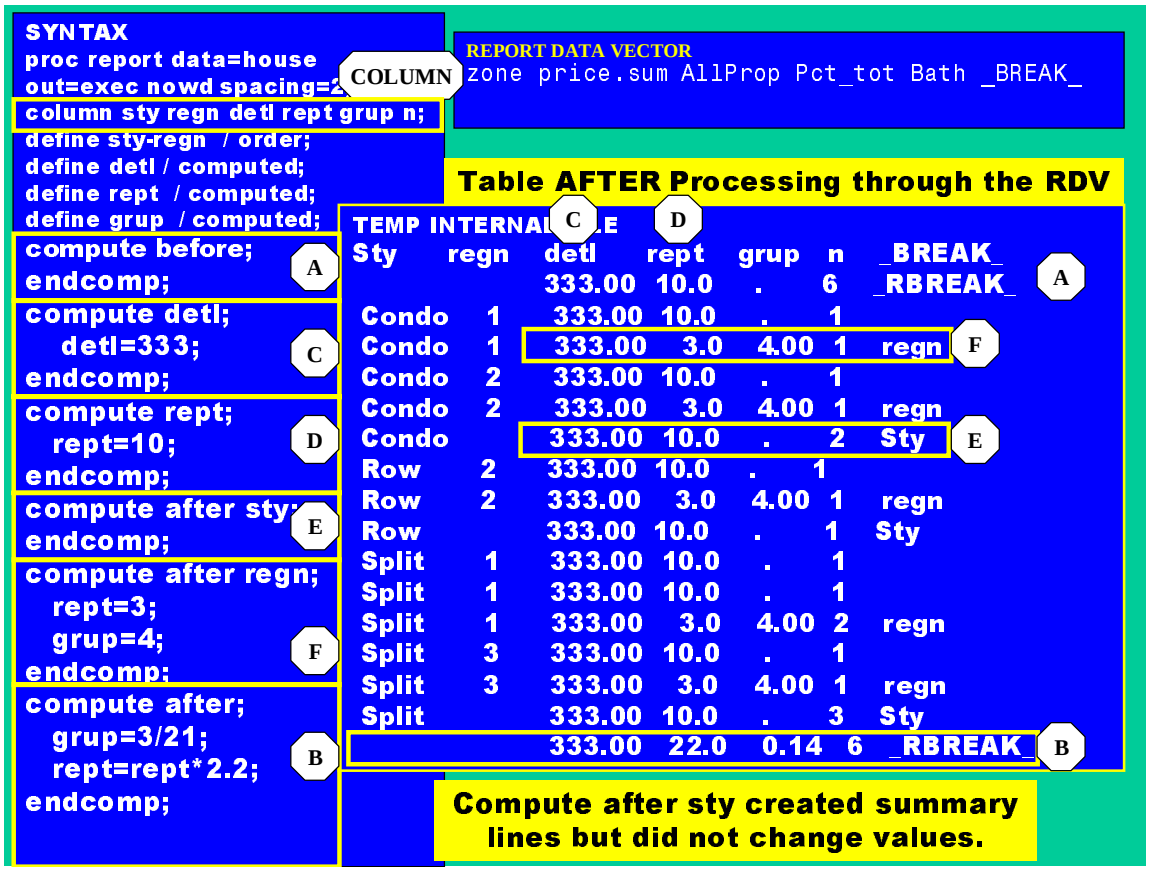

Figure 6

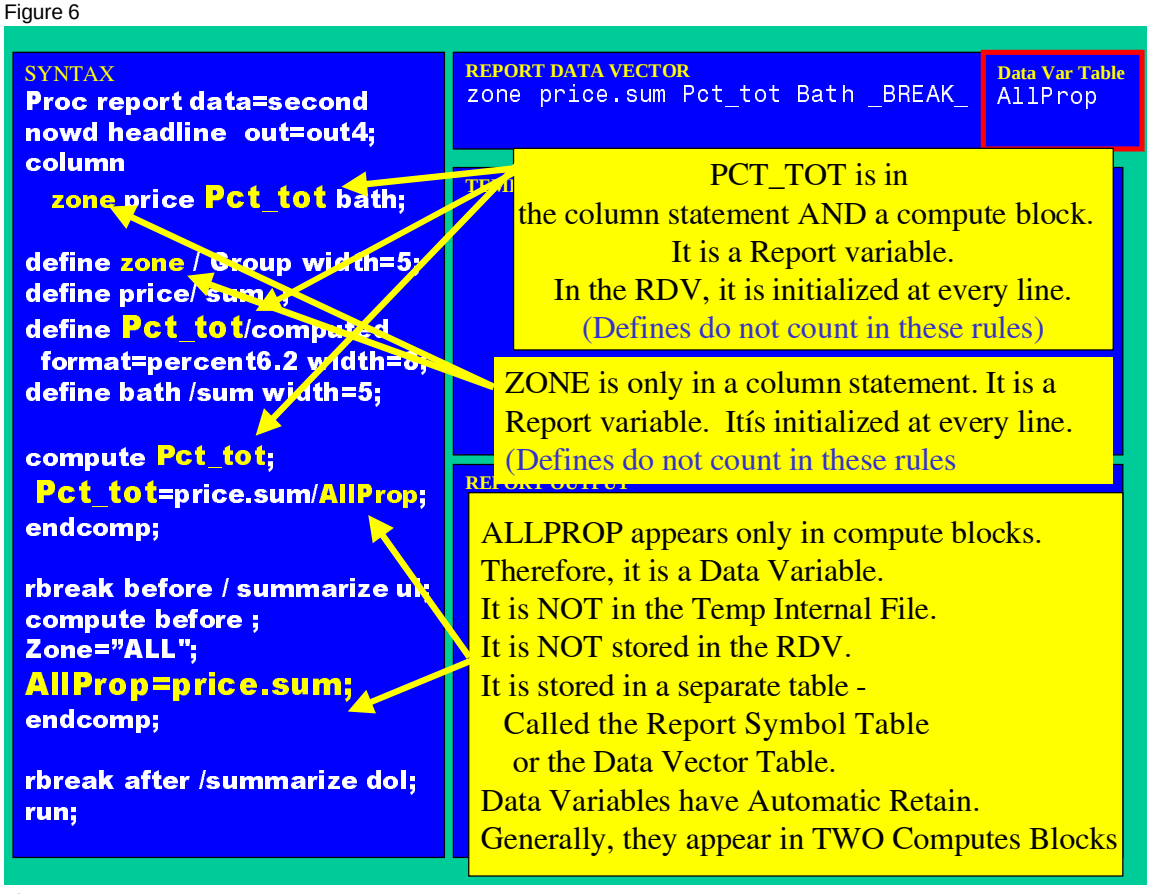

Figure 8

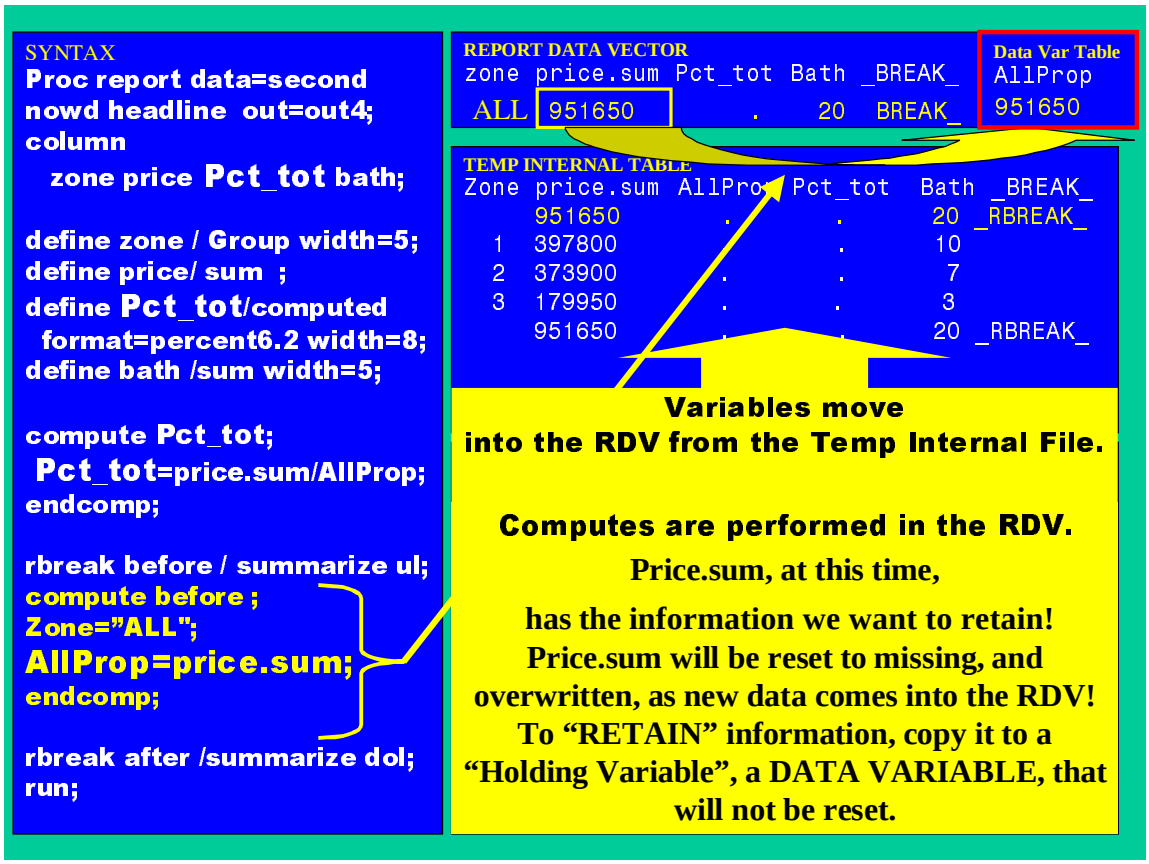

Figure 9

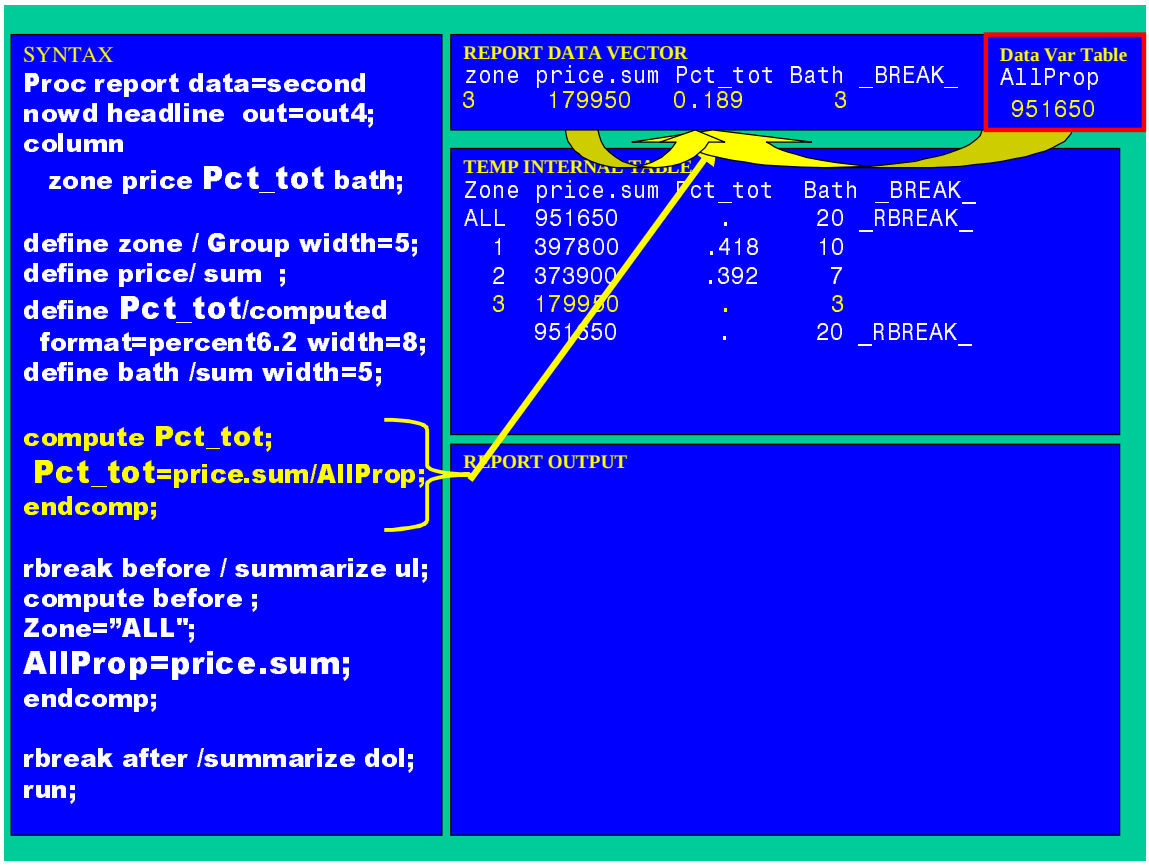

Figure 10

DATA HOUSE1; INFILE DATALINES;

```
INPUT TYPE $ ZONE $ PRICE BATH @@;
DATALINES;
Split    3  179950 3   Condo    1  214900 3
Split    1  100000 4  Split    1   82900 3
Condo    2  189900 4  Row      2  184000 3
;
RUN;

**EXAMPLE 1*** BBREAK BEFORE;
Proc report data=HOUSE nowd headline
out=out1;
column      zone type price bath;
define zone / order width=5;
define type / order ;
define price/ sum  ;
define bath /sum width=5;
Rbreak before   /summarize dul;
run;

PROC PRINT DATA=OUT1;
RUN;

**EXAMPLE 2 ** COMPUTE AFTER ;
Proc report data= HOUSING1 headline  NOWD
out=out2;
where zone GE "2";
column    zone type price bath;
define zone      /order width=5;
define type      / order;
define price     / sum  ;
define bath      / sum width=5;
Break after zone /summarize  ol skip     suppress;
Rbreak  after    /summarize dol;
compute after  ;
   Zone="ALL";
endcomp;
QUIT;

PROC PRINT DATA=OUT2;
RUN;

*****EXAMPLE 3 ;
 Proc report data=HOUSING1 nowd headline
out=out3;
column zone price bath;
define zone   / Group width=5;
define price  / sum  ;
define bath   /sum width=5;
rbreak before /summarize ul;
compute before ;
Zone = 'ALL' ;
endcomp;
where zone le '2';
run;

PROC PRINT DATA=OUT3;
RUN;

************************************************************
*******;
**MAKE VAR NAMES SHORTER SO THEY FIIT
ON A SLIDE**************;
DATA FEW;
INFILE DATALINES;
INPUT STY $ REGN $ PRICE BATH @@;
DATALINES;
Split    3  179950 3   Condo    1  214900 3
Split    1  100000 4  Split    1   82900 3
Condo    2  189900 4  Row      2  184000 3
;
RUN;

*****EXAMPLE 4a ;
proc report data=few out=out4a nowd spacing=2;
column sty regn detl rept grup n;
define sty / order;
define sty / order;
define detl / computed;
define rept  / computed;
define grup  / computed;
compute before;
endcomp;
compute detl;
   detl=333;
endcomp;
compute rept;
  rept=10;
endcomp;
compute after sty;
endcomp;
compute after regn;
  rept=3;
  grup=4;
endcomp;
compute after;
  grup=3/21;
  rept=rept*2.2;
endcomp;
proc print data=out4a;
run;



**example 4b *NEEDS  WORK  *******;
PROC REPORT data=few out=out4b nowd
spacing=2;
column sty regn detl rept grup n;
define sty   /order;
define regn  /order ;
define detl  /computed;
define rept  /computed;
define grup  /computed;


compute before;
endcomp;

compute detl;
if sty="" and _Break_= "_RBREAK_"  then
detl=111;
 else if sty="" then detl=222;
 else if regn=. then detl=777;
 else detl=333;
endcomp;

compute rept;
  rept=10;
endcomp;

compute after sty;
endcomp;

compute after regn;
rept=3;
grup=4;
endcomp;

compute after;
grup=3/21;
rept=rept*2.2;
endcomp;
run;
quit;

proc print data=out4b;
run;

*****BACK TO HOUSING ONE DATA
SET**********************;
*****EXAMPLE 5 DATA VARIABLES **
RETAIINED VAIRABLES ***;
Proc report data=housing1
nowd headline  out=out5;
column
  zone price Pct_tot bath;

define zone / Group width=5;
define price/ sum  ;
define Pct_tot/computed   format=percent6.2
width=8;
define bath /sum width=5;

copute Pct_tot;
 Pct_tot=price.sum/AllProp;
endcomp;

rbreak before / summarize ul;
compute before ;
Zone="ALL";
AllProp=price.sum;
endcomp;

rbreak after /summarize dol;
run;

proc print data=out5;
run;

******Example 6**** ;
OPTIONS MISSING=0;
data h_2;
infile datalines;
input type $ zone price_k bath;
datalines;
Split   3    179.9    3
Condo   1    214.9    3
Split   1    100.0    4
Split   1     82.9    3
Condo   2    189.9    4
Row     2    184.0    3

run;
proc print;
run;

proc report data=H_2 NOWD out=o6;
column  type zone  price_k=count price_k bath;
Define type    /group  width=12;
define zone    /across "# Prop./in  Zone"
width=2;
define count   /n "# Prop/on Mkt."    width=11;
define price_k  /analysis mean format=6.1
"Avg./Price" width=10;
rbreak after    / dol   summarize skip;
RUN;
QUIT;

Proc print data=O6;
run;
```