Paper SAS431-2017

Go Ahead and _BREAK_-down: Advanced COMPUTE Block Examples

Cynthia Zender, SAS Institute Inc.

ABSTRACT

When you look at examples of the REPORT procedure, you see code that tests _BREAK_ and _RBREAK_, but you wonder what's the breakdown of the COMPUTE block? And, sometimes, you need more than one break line on a report, or you need a customized or adjusted number at the break. Everything in PROC REPORT that is advanced seems to involve a COMPUTE block. This paper provides examples of advanced PROC REPORT output that uses _BREAK_ and _RBREAK_ to customize the extra break lines that you can request with PROC REPORT. Examples include how to get custom percentages with PROC REPORT, how to get multiple break lines at the bottom of the report, how to customize break lines, and how to customize LINE statement output. This presentation is aimed at the intermediate to advanced report writer who knows some about PROC REPORT, but wants to get the breakdown of how to do more with PROC REPORT and the COMPUTE block.

INTRODUCTION

Don't break up with PROC REPORT over the COMPUTE block and BREAK processing. This paper will cover advanced COMPUTE block examples using PROC REPORT. PROC REPORT is one of those procedures that you can use for a long time without needing to do anything advanced with the COMPUTE block. PROC REPORT can produce many types of output, from simple detail reports to summarized reports. Like its predecessors PROC PRINT and PROC MEANS, PROC REPORT has syntax that enables you to control the variables that you want to see on a report and the statistics that you will get for analysis variables. Like PROC TABULATE, PROC REPORT can generate crosstabular reports.

What PROC REPORT brings to the party is the ability to customize report output by calculating new report items and customizing the report at specific places on the report. The customization steps occur in the PROC REPORT COMPUTE block, which is not included in PRINT, MEANS, and TABULATE.

All the examples in the paper use the SASHELP.SHOES data set. In the ZIP file that contains the programs, all the WORK data sets that are needed for the examples are created in the **Example_Before_Setup.sas** program. Please refer to that file to see the subsets that are used. SASHELP.SHOES has ten unique REGION values. However, for most of these examples, only two or three REGION values are needed for illustration purposes. In some programs, a WHERE statement is used in PROC REPORT to reduce the number of PRODUCTS that will be shown on a report.

PROC REPORT, in the COMPUTE block, enables you to include DATA step processing and to use familiar DATA step statements such as the assignment statement, the SUM statement, and the LENGTH statement. When PROC REPORT first starts, it enters a "preprocessing" phase. It scans the REPORT syntax to perform any necessary setup such as any summarizing or ordering that needs to happen based on the type of report you've requested.

In addition, PROC REPORT puts the COMPUTE block syntax in a "hold" state so that the appropriate statements can be called into action as needed. But, to understand the COMPUTE block, first you need be comfortable with simple COMPUTE block syntax.

FIRST THINGS FIRST: SIMPLE COMPUTE BLOCKS AND PROC REPORT

What I call a "simple" COMPUTE block is the form of COMPUTE block that most people learn first. It is the COMPUTE block that creates a new report item on the report output. The item is assigned a value in the COMPUTE block. The value could be a constant value or a calculated item, or it could be derived from other variables or report items that are in the COLUMN statement.

The first program illustrates defaults (**Example_Beginning_Defaults.sas**), which produces Output 1. Report A1 shows the default report that you would get from the following code:

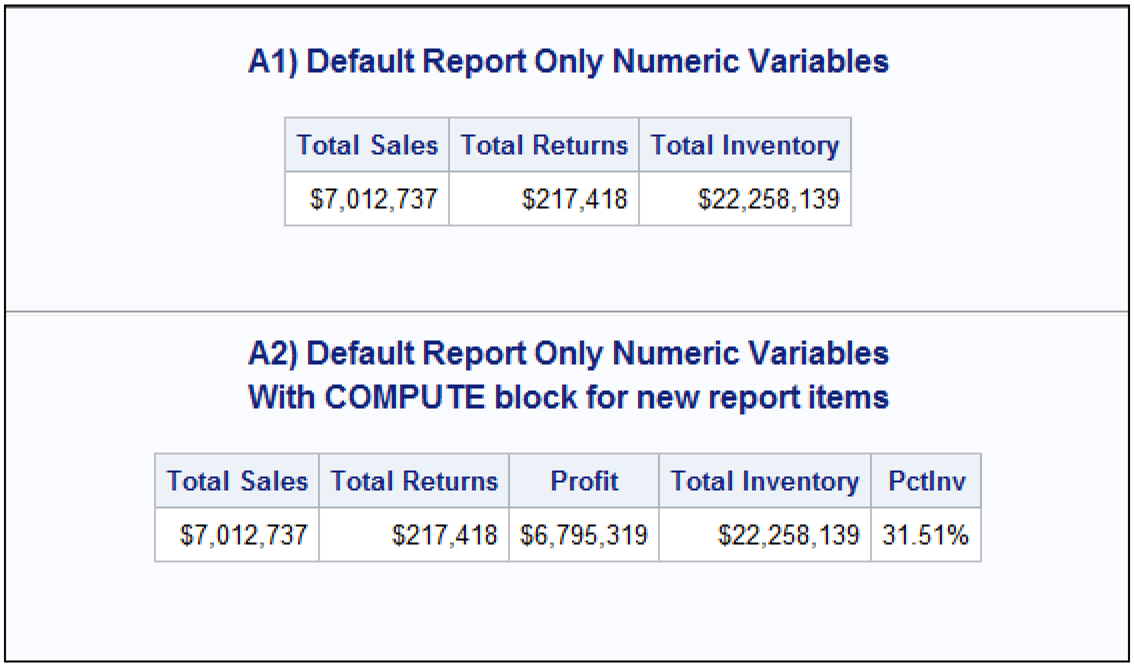
```
** Refer to PROC SORT to see how subset created;
proc report data=shoes;
  title 'A1) Default Report Only Numeric Variables';
  column sales returns inventory ;
run;
```

Because the COLUMN statement contains only numeric variables, all of the rows for the SALES, RETURNS, and INVENTORY variables from the WORK.SHOES data are summarized, producing a one-line "grand total" report.

Report A2, in the same program, illustrates the use of the simple COMPUTE block to calculate two new report items, called PROFIT and PCTINV. These items do not exist in SASHELP.CLASS, so they do not exist in the subset of data that is sent to PROC REPORT. Instead, PROC REPORT computes these values by executing the statements inside the COMPUTE block every time it needs to populate a data cell. In this case, the COMPUTE block executes only one time for each item because there is only one row on the report.

Here is the code that generates Report A2:

```
proc report data=shoes;
  title 'A2) Default Report Only Numeric Variables';
  title2 'With COMPUTE block for new report items';
  column sales returns Profit inventory PctInv; ←1
  define profit / computed f=dollar14.; ←2
  define pctinv / computed f=percent9.2; ←2
  compute profit; ←3
    profit = sum(sales.sum,-1*returns.sum);
  endcomp;
  compute pctinv; ←3
    pctinv = sales.sum / inventory.sum;
  endcomp;
run;
```



A1) Default Report Only Numeric Variables

Total Sales	Total Returns	Total Inventory
$7,012,737	$217,418	$22,258,139

A2) Default Report Only Numeric Variables
With COMPUTE block for new report items

Total Sales	Total Returns	Profit	Total Inventory	PctInv
$7,012,737	$217,418	$6,795,319	$22,258,139	31.51%

Output 1: Report with Computed Items

In order for PROFIT and PCTINV to be computed correctly, three statements in the PROC REPORT code are necessary because the statements work together:

1. Both PROFIT and PCTINV had to be listed in the COLUMN statement in the order (from left to right) in which they should appear on the report.
2. The DEFINE statements had to be provided to specify a usage of COMPUTED and a format for each report item.
3. The COMPUTE/ENDCOMP block for each item instructed PROC REPORT what expression should be used in order to assign a value to PROFIT and PCTINV.

Because SALES, INVENTORY, and RETURNS are being treated as ANALYSIS variables with a default statistic of SUM, they must be referred to in an assignment statement using a compound name: SALES.SUM, INVENTORY.SUM and RETURNS.SUM.

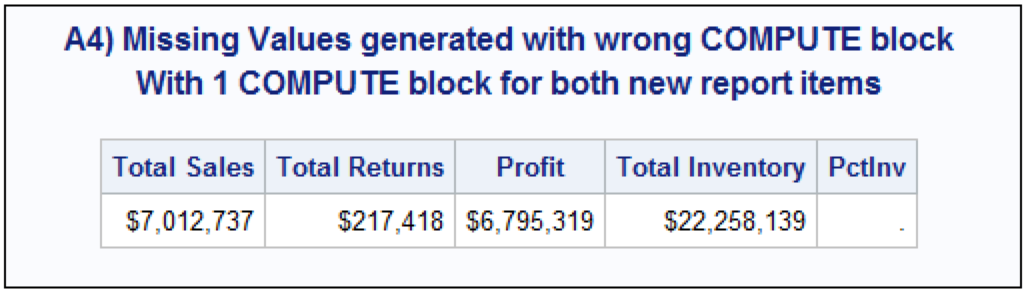
In the code for Report A2, there is a separate COMPUTE block for each computed report item. However, one of the features of PROC REPORT is that it builds each report row from left to right. Therefore, the exact same output could be generated by putting both assignment statements in the COMPUTE block for PCTINV, as shown below, with the modified code for report A3 that produces the same output as shown for report A2 in Output 1.

```
proc report data=shoes;
  title 'A3) Default Report Only Numeric Variables';
  column sales returns Profit inventory PctInv;
  define profit / computed f=dollar14.;
  define pctinv / computed f=percent9.2;
  compute pctinv;
    profit = sum(sales.sum,-1*returns.sum);
    pctinv = sales.sum / inventory.sum;
  endcomp;
run;
```

If you prefer to keep your simple COMPUTE block assignment statements together, then this technique enables you to accomplish that goal. Remember that PROC REPORT fills the data cells on a report row by working from left to right. So this same technique, which puts both assignment statements in the COMPUTE block for PROFIT, would not work:

```
proc report data=shoes;
  title 'A4) Default Report Only Numeric Variables';
  title2 'With 1 COMPUTE block for both new report items';
  column sales returns Profit inventory PctInv;
  define profit / computed f=dollar14.;
  define pctinv / computed f=percent9.2;
  compute profit;
    profit = sum(sales.sum,-1*returns.sum);
    pctinv = sales.sum / inventory.sum;
  endcomp;
run;
```

This code will not generate errors, but it does generate missing values for PCTINV. The reason this code generates missing values for PCTINV is that the point in time at which PROC REPORT puts PROFIT on the report row, PROC REPORT has not yet placed INVENTORY on the report row. If you run this program, your results will not be what you expect. Output 2 shows the result of using the wrong placement for PCTINV in the COMPUTE block for PROFIT:


A4) Missing Values generated with wrong COMPUTE block
With 1 COMPUTE block for both new report items

Total Sales	Total Returns	Profit	Total Inventory	PctInv
$7,012,737	$217,418	$6,795,319	$22,258,139	.

Output 2: Missing Values for the PCTINV Computed Item

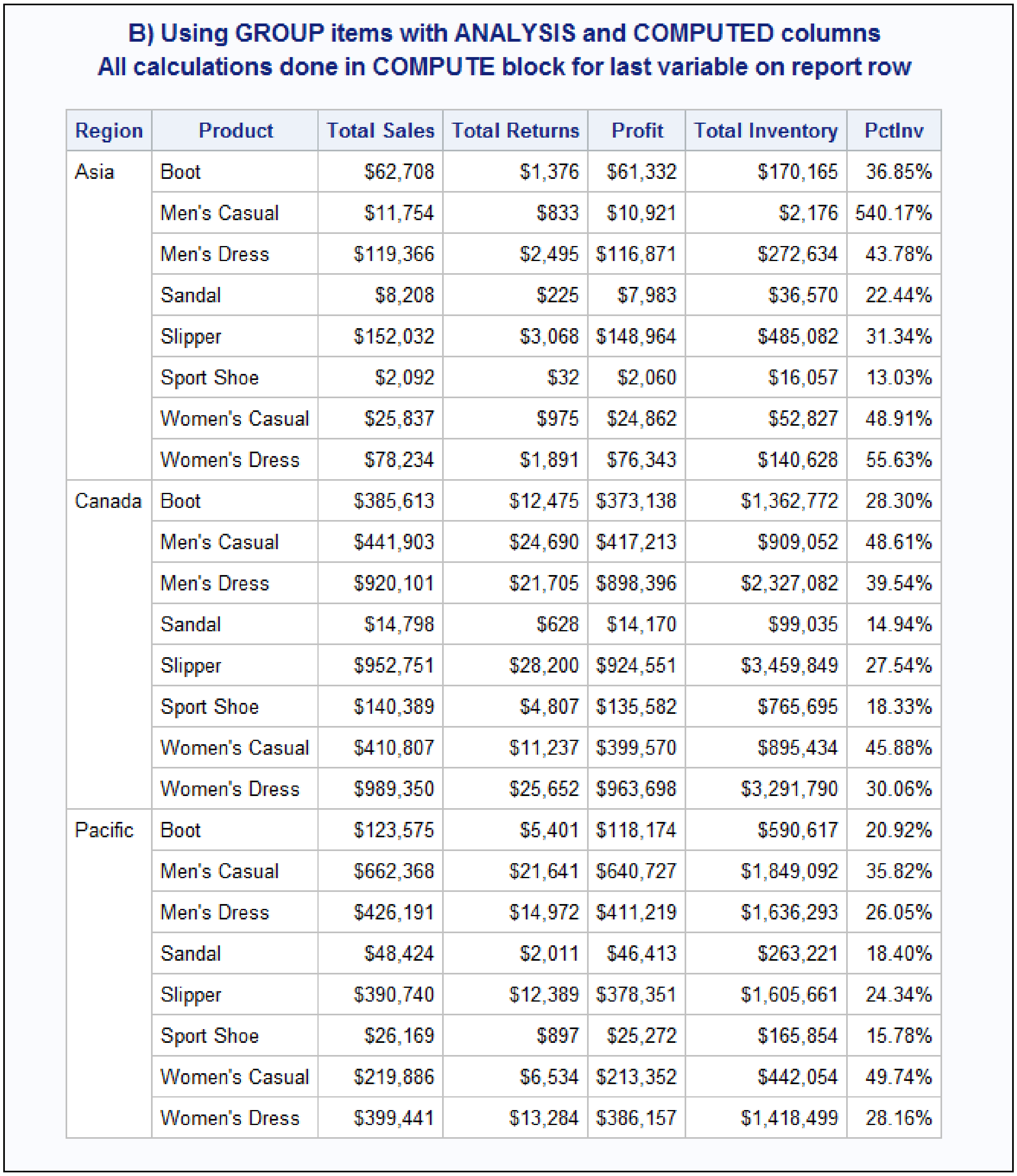
The two most challenging aspects about using the simple COMPUTE block are remembering the left-to-right rule of PROC REPORT and using the COMPOUND name, *variable.statistic* for analysis variables in your assignment statement.

But what if you want to see these same columns, but organized by REGION and PRODUCT? That is easy to do by adding REGION and PRODUCT to the COLUMN statement. By default, PROC REPORT will treat the variables as display items and will display every row in the input data. This subset of data has 96 observations. However, the desired report should have one summary line for every unique combination of REGION and PRODUCT. This means that both REGION and PRODUCT will need to have a DEFINE statement with a usage of GROUP. If you just add REGION and PRODUCT to the COLUMN statement, without a DEFINE usage of GROUP, then PROC REPORT, by default, will treat the variables as display items and will display every row in the input data. This subset of data has 96 observations. However, the desired report will have one summary line for every unique combination for REGION and PRODUCT. In other words, the desired report is a summary report, which means that both REGION and PRODUCT must have a DEFINE statement with a usage of GROUP.

This code will generate the desired summary report:

```
proc report data=shoes spanrows;
  column region product sales returns Profit inventory PctInv ;
  define region / group;
  define product / group;
  define profit / computed f=dollar14.;
  define pctinv / computed f=percent9.2;
  compute pctinv;
    profit = sum(sales.sum,-1*returns.sum);
    pctinv = sales.sum / inventory.sum;
  endcomp;
run;
```

The majority of the program is the same as that seen previously. The only changes are the use of the SPANROWS option, which causes the GROUP variable for REGION to span multiple rows for PRODUCT. For example, in Output 3 notice how the cell for Asia in the REGION column spans all the cells for the PRODUCT rows. In addition, a DEFINE statement is used for REGION and PRODUCT to indicate that they should have a usage of GROUP so that multiple rows for each unique REGION/PRODUCT combination would be collapsed or summarized. The resulting output is shown in Output 3, Report B.



B) Using GROUP items with ANALYSIS and COMPUTED columns
All calculations done in COMPUTE block for last variable on report row

Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Asia	Boot	$62,708	$1,376	$61,332	$170,165	36.85%
	Men's Casual	$11,754	$833	$10,921	$2,176	540.17%
	Men's Dress	$119,366	$2,495	$116,871	$272,634	43.78%
	Sandal	$8,208	$225	$7,983	$36,570	22.44%
	Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
	Sport Shoe	$2,092	$32	$2,060	$16,057	13.03%
	Women's Casual	$25,837	$975	$24,862	$52,827	48.91%
	Women's Dress	$78,234	$1,891	$76,343	$140,628	55.63%
Canada	Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
	Men's Casual	$441,903	$24,690	$417,213	$909,052	48.61%
	Men's Dress	$920,101	$21,705	$898,396	$2,327,082	39.54%
	Sandal	$14,798	$628	$14,170	$99,035	14.94%
	Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
	Sport Shoe	$140,389	$4,807	$135,582	$765,695	18.33%
	Women's Casual	$410,807	$11,237	$399,570	$895,434	45.88%
	Women's Dress	$989,350	$25,652	$963,698	$3,291,790	30.06%
Pacific	Boot	$123,575	$5,401	$118,174	$590,617	20.92%
	Men's Casual	$662,368	$21,641	$640,727	$1,849,092	35.82%
	Men's Dress	$426,191	$14,972	$411,219	$1,636,293	26.05%
	Sandal	$48,424	$2,011	$46,413	$263,221	18.40%
	Slipper	$390,740	$12,389	$378,351	$1,605,661	24.34%
	Sport Shoe	$26,169	$897	$25,272	$165,854	15.78%
	Women's Casual	$219,886	$6,534	$213,352	$442,054	49.74%
	Women's Dress	$399,441	$13,284	$386,157	$1,418,499	28.16%

Output 3: Using GROUP Items with COMPUTED Items

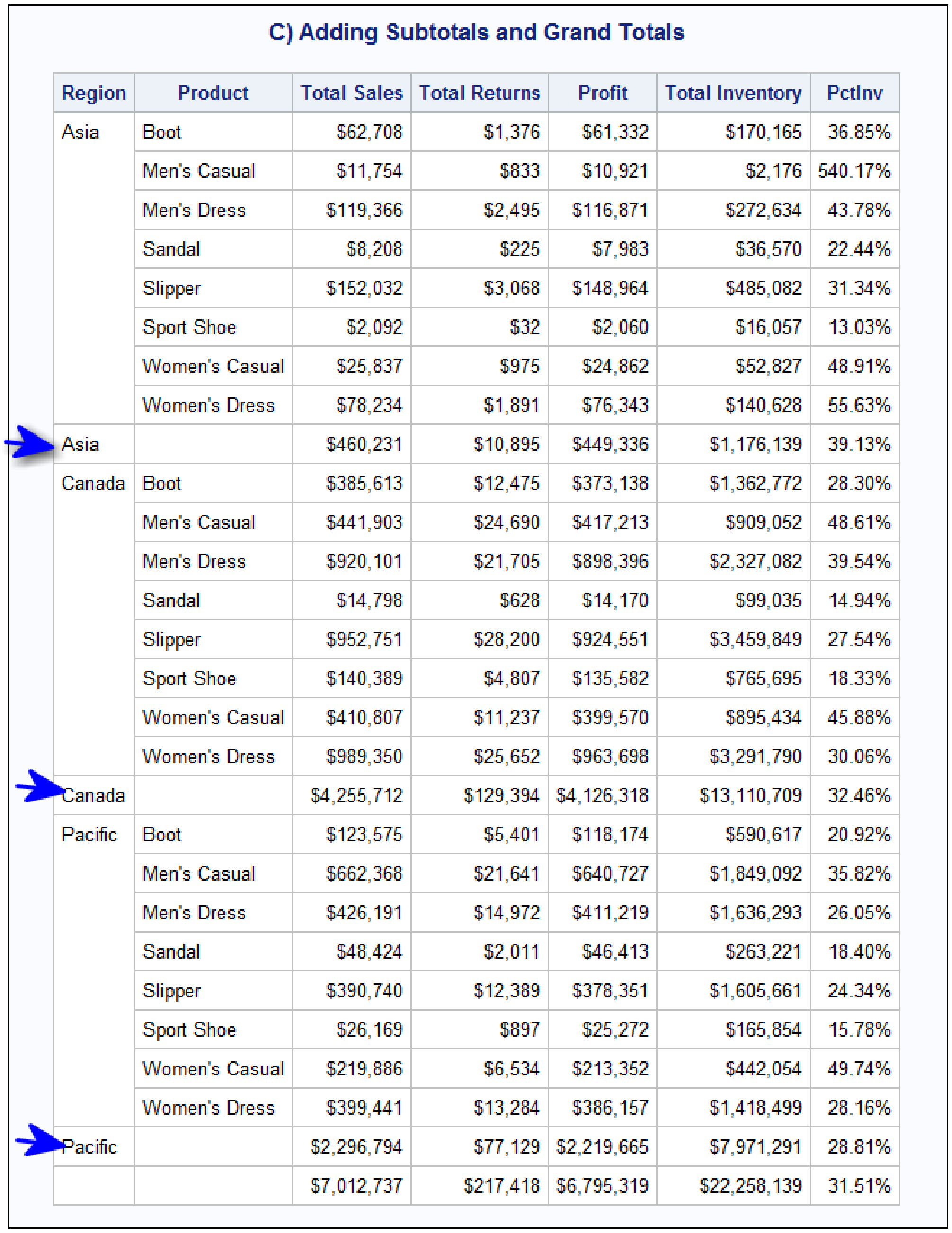
So far, in the output, the only rows on the report are the column header rows and the cells that show the summarized data. There are no extra report rows, and there are no subtotals or grand totals on the report.

If a usage of ORDER had been specified for REGION or PRODUCT or both REGION and PRODUCT, then the report would have shown all 96 observations from WORK.SHOES. Instead, the resulting report has three regions, with the summary for each of the eight products within each region. The final report has a total of 24 report rows, plus the column header row.

To get subtotals and grand totals, only two statements must be added to the PROC REPORT code:

```
  break after region/summarize;
  rbreak after / summarize;
```

The BREAK statement will add an extra subtotal row after every unique value of REGION, and the RBREAK statement will add a grand total row at the bottom of the report, as shown in Output 4.



C) Adding Subtotals and Grand Totals

Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Asia	Boot	$62,708	$1,376	$61,332	$170,165	36.85%
	Men's Casual	$11,754	$833	$10,921	$2,176	540.17%
	Men's Dress	$119,366	$2,495	$116,871	$272,634	43.78%
	Sandal	$8,208	$225	$7,983	$36,570	22.44%
	Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
	Sport Shoe	$2,092	$32	$2,060	$16,057	13.03%
	Women's Casual	$25,837	$975	$24,862	$52,827	48.91%
	Women's Dress	$78,234	$1,891	$76,343	$140,628	55.63%
Asia		$460,231	$10,895	$449,336	$1,176,139	39.13%
Canada	Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
	Men's Casual	$441,903	$24,690	$417,213	$909,052	48.61%
	Men's Dress	$920,101	$21,705	$898,396	$2,327,082	39.54%
	Sandal	$14,798	$628	$14,170	$99,035	14.94%
	Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
	Sport Shoe	$140,389	$4,807	$135,582	$765,695	18.33%
	Women's Casual	$410,807	$11,237	$399,570	$895,434	45.88%
	Women's Dress	$989,350	$25,652	$963,698	$3,291,790	30.06%
Canada		$4,255,712	$129,394	$4,126,318	$13,110,709	32.46%
Pacific	Boot	$123,575	$5,401	$118,174	$590,617	20.92%
	Men's Casual	$662,368	$21,641	$640,727	$1,849,092	35.82%
	Men's Dress	$426,191	$14,972	$411,219	$1,636,293	26.05%
	Sandal	$48,424	$2,011	$46,413	$263,221	18.40%
	Slipper	$390,740	$12,389	$378,351	$1,605,661	24.34%
	Sport Shoe	$26,169	$897	$25,272	$165,854	15.78%
	Women's Casual	$219,886	$6,534	$213,352	$442,054	49.74%
	Women's Dress	$399,441	$13,284	$386,157	$1,418,499	28.16%
Pacific		$2,296,794	$77,129	$2,219,665	$7,971,291	28.81%
		$7,012,737	$217,418	$6,795,319	$22,258,139	31.51%

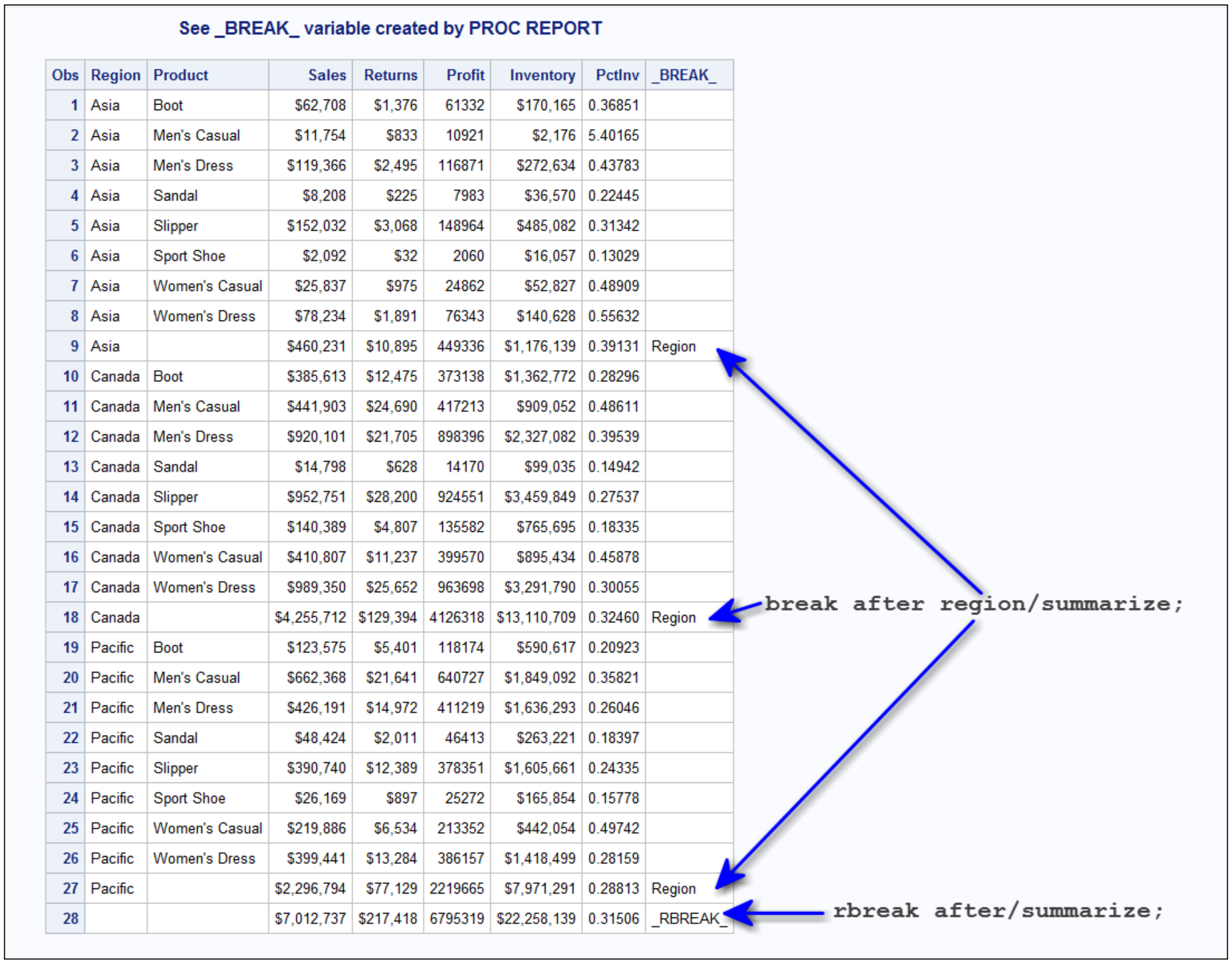
Output 4: Adding Subtotals and Grand Totals

The subtotal rows are highlighted with a blue arrow in Output 4, Report C. You see the grand total row at the bottom of the report. PROC REPORT has added four additional report rows, all derived from the data and placed at the location that is specified in the BREAK and RBREAK statements.

Internally, when PROC REPORT performs break processing, it uses a helper variable called _BREAK_ to identify which rows are derived from data and which rows are on the report because of break processing. We use this helper variable in upcoming programs. A useful technique for understanding how PROC REPORT sets the value of the _BREAK_ variable is to create an output data set using the OUT= option of PROC REPORT:

```
    proc report data=shoes spanrows out=work.repout;
```

The WORK.REPOUT data set is displayed using PROC PRINT in Output 5.



See _BREAK_ variable created by PROC REPORT

Obs	Region	Product	Sales	Returns	Profit	Inventory	PctInv	_BREAK_
1	Asia	Boot	$62,708	$1,376	61332	$170,165	0.36851	
2	Asia	Men's Casual	$11,754	$833	10921	$2,176	5.40165	
3	Asia	Men's Dress	$119,366	$2,495	116871	$272,634	0.43783	
4	Asia	Sandal	$8,208	$225	7983	$36,570	0.22445	
5	Asia	Slipper	$152,032	$3,068	148964	$485,082	0.31342	
6	Asia	Sport Shoe	$2,092	$32	2060	$16,057	0.13029	
7	Asia	Women's Casual	$25,837	$975	24862	$52,827	0.48909	
8	Asia	Women's Dress	$78,234	$1,891	76343	$140,628	0.55632	
9	Asia		$460,231	$10,895	449336	$1,176,139	0.39131	Region
10	Canada	Boot	$385,613	$12,475	373138	$1,362,772	0.28296	
11	Canada	Men's Casual	$441,903	$24,690	417213	$909,052	0.48611	
12	Canada	Men's Dress	$920,101	$21,705	898396	$2,327,082	0.39539	
13	Canada	Sandal	$14,798	$628	14170	$99,035	0.14942	
14	Canada	Slipper	$952,751	$28,200	924551	$3,459,849	0.27537	
15	Canada	Sport Shoe	$140,389	$4,807	135582	$765,695	0.18335	
16	Canada	Women's Casual	$410,807	$11,237	399570	$895,434	0.45878	
17	Canada	Women's Dress	$989,350	$25,652	963698	$3,291,790	0.30055	
18	Canada		$4,255,712	$129,394	4126318	$13,110,709	0.32460	Region
19	Pacific	Boot	$123,575	$5,401	118174	$590,617	0.20923	
20	Pacific	Men's Casual	$662,368	$21,641	640727	$1,849,092	0.35821	
21	Pacific	Men's Dress	$426,191	$14,972	411219	$1,636,293	0.26046	
22	Pacific	Sandal	$48,424	$2,011	46413	$263,221	0.18397	
23	Pacific	Slipper	$390,740	$12,389	378351	$1,605,661	0.24335	
24	Pacific	Sport Shoe	$26,169	$897	25272	$165,854	0.15778	
25	Pacific	Women's Casual	$219,886	$6,534	213352	$442,054	0.49742	
26	Pacific	Women's Dress	$399,441	$13,284	386157	$1,418,499	0.28159	
27	Pacific		$2,296,794	$77,129	2219665	$7,971,291	0.28813	Region
28			$7,012,737	$217,418	6795319	$22,258,139	0.31506	_RBREAK_

Output 5: Values of the _BREAK_ Variable Displayed in PROC PRINT Output

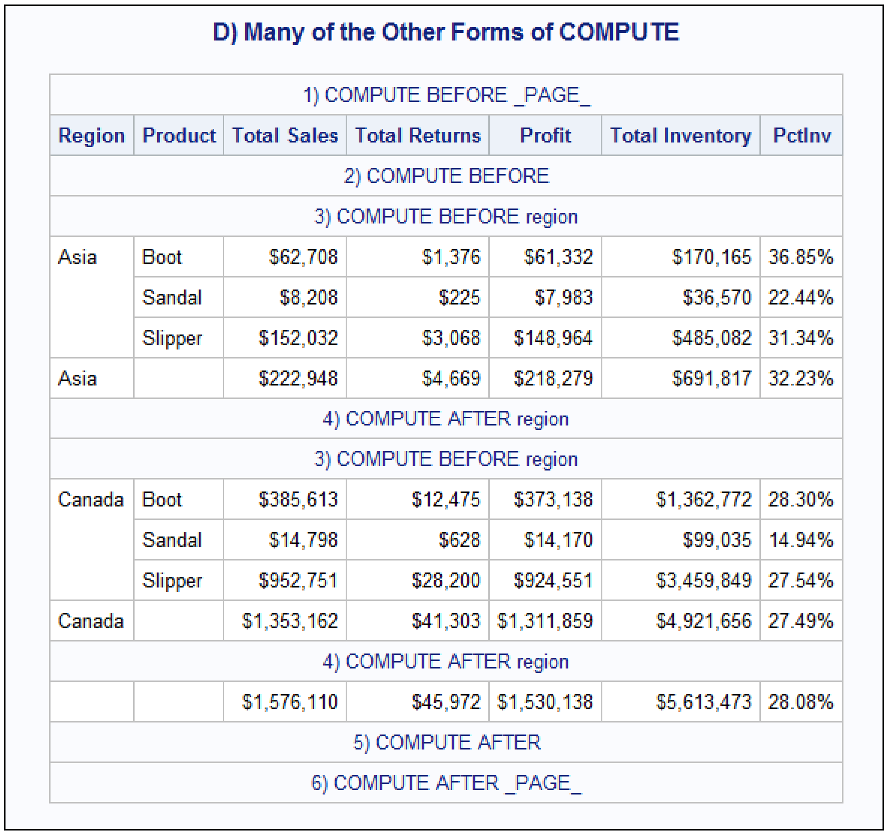
The value of the _BREAK_ variable from the BREAK statement is the name of the variable, REGION, which was listed in the BREAK statement. The value of the _BREAK_ variable from the RBREAK statement (at the bottom of the report) is _RBREAK_. This makes it easy to perform conditional processing in your COMPUTE blocks. The name of the variable to test is always _BREAK_ and the possible values are the name of a GROUP or ORDER item, _RBREAK_, or _PAGE_.

Notice in Output 5 for the report rows that were summarized from the data, the value of _BREAK_ is blank. Now, you can see exactly which rows were added to the report by the break processing.

The final example in this section illustrates other uses of the COMPUTE block with the LINE statement, which is not related to the creation of a report item. To illustrate how COMPUTE processing works, quite a few extra rows are added to the report that we've been working with. To fit the output onto one page and to illustrate the COMPUTE behavior, the number of regions is limited to two and the number of products limited to three in the code, using a WHERE statement:

```
        proc report data=shoes spanrows;
          where region in ('Asia' 'Canada') and
                product in ('Boot' 'Sandal' 'Slipper');
          title 'D) Many of the Other Forms of COMPUTE';
          ** rest of code same as previous **;
          compute before _page_;
            line '1) COMPUTE BEFORE _PAGE_';
          endcomp;
          compute before;
            line '2) COMPUTE BEFORE';
          endcomp;
          compute before region;
            line '3) COMPUTE BEFORE region';
          endcomp;
          compute after region;
            line '4) COMPUTE AFTER region';
          endcomp;
          compute after;
            line '5) COMPUTE AFTER';
          endcomp;
          compute after _page_;
            line '6) COMPUTE AFTER _PAGE_';
          endcomp;
      run;
```

The LINE statement, which is unique to PROC REPORT, functions like a PUT statement in the DATA step. In Output 6, the results of the various LINE statements within each COMPUTE block are shown.



D) Many of the Other Forms of COMPUTE

1) COMPUTE BEFORE _PAGE_						
Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
2) COMPUTE BEFORE						
3) COMPUTE BEFORE region						
Asia	Boot	$62,708	$1,376	$61,332	$170,165	36.85%
	Sandal	$8,208	$225	$7,983	$36,570	22.44%
	Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
Asia		$222,948	$4,669	$218,279	$691,817	32.23%
4) COMPUTE AFTER region						
3) COMPUTE BEFORE region						
Canada	Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
	Sandal	$14,798	$628	$14,170	$99,035	14.94%
	Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
Canada		$1,353,162	$41,303	$1,311,859	$4,921,656	27.49%
4) COMPUTE AFTER region						
		$1,576,110	$45,972	$1,530,138	$5,613,473	28.08%
5) COMPUTE AFTER						
6) COMPUTE AFTER _PAGE_						

Output 6: Possible Locations for the COMPUTE Blocks to Add Rows to the Output

For a thorough description of COMPUTE block execution, refer to Jane Eslinger's 2015 paper R, which is listed in the References section.

The difference between the simple COMPUTE and the COMPUTE output that is shown in Output 6 is that the COMPUTE block with a report item reference, COMPUTE PROFIT or COMPUTE PCTINV**,** has an impact on a data cell on each report row, and the LINE statements in the 6 COMPUTE blocks inserts a row at a specific location on the report. Notice how lines 3 and 4 were repeated for every unique value of the group item REGION, immediately before the first occurrence of REGION and immediately after the last occurrence of REGION.

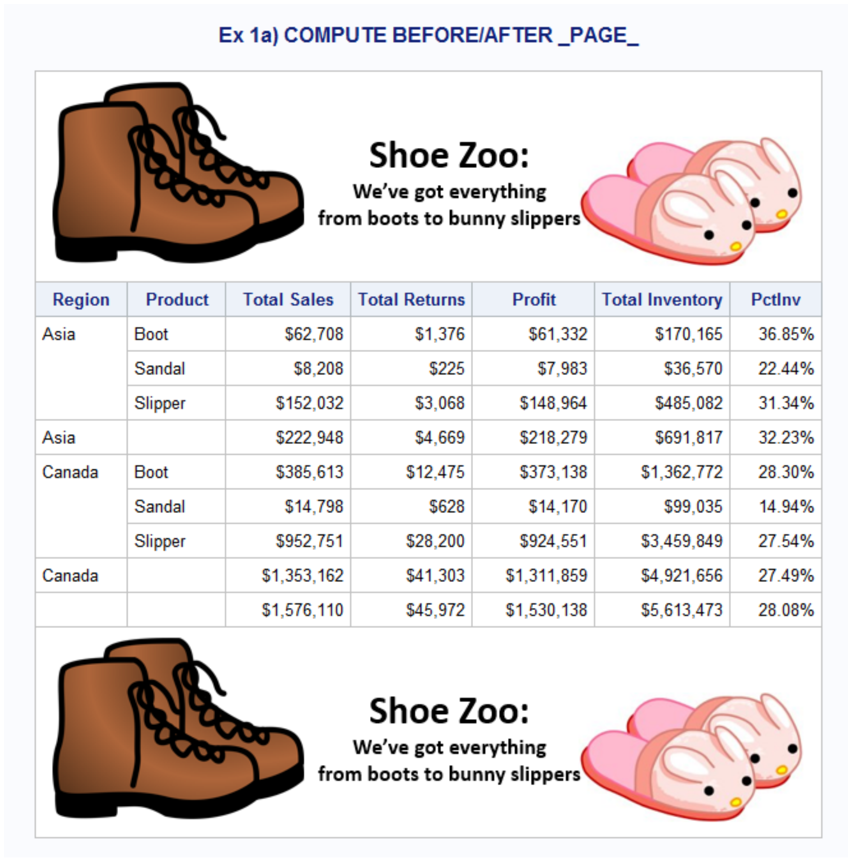
Another important feature to note is that the COMPUTE output always stays within the boundary of the table and, as you can see, the LINE statement output spans the entire table width. In Output 6, a gray frame surrounds the entire table, and the LINE statement output stays within this frame area. You can use ODS STYLE overrides to alter the presentation aspects of the line output. You'll see examples of such overrides in the following sections.

EXAMPLE 1: COMPUTE BEFORE/AFTER _PAGE_

There are two urgent reasons to use COMPUTE BEFORE or AFTER _PAGE_. One reason is to insert logos or images either before or after your report table. The other reason is to use the area above or below the table for captions or explanatory text. Both examples are shown in the following code. For the caption example, only the changed COMPUTE blocks are shown:

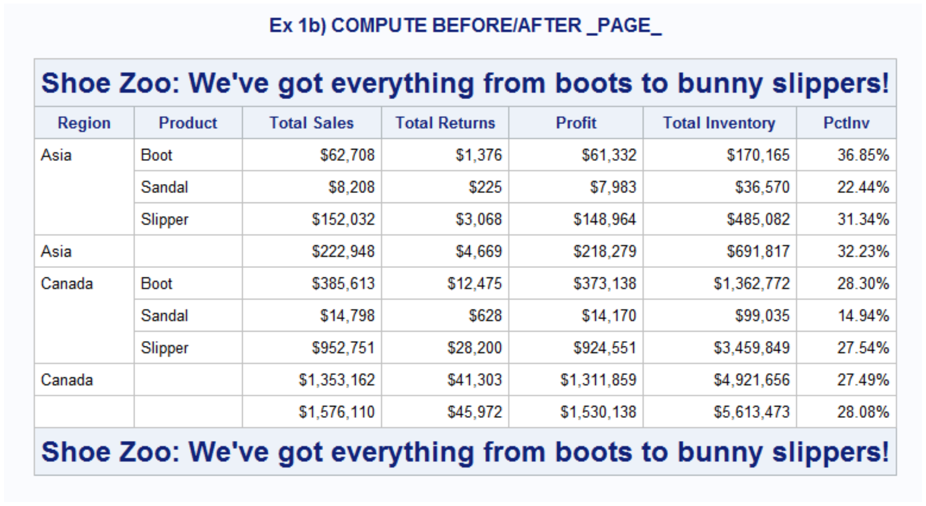
```
proc report data=shoes spanrows;
  where region in ('Asia' 'Canada') and
        product in ('Boot' 'Sandal' 'Slipper');
  title 'Ex 1a) COMPUTE BEFORE/AFTER _PAGE_';
  column region product sales returns Profit inventory PctInv ;
  define region / group;
  define product / group;
  define profit / computed f=dollar14.;
  define pctinv / computed f=percent9.2;
  break after region/summarize;
  rbreak after / summarize;
  compute pctinv;
    profit = sum(sales.sum,-1*returns.sum);
    pctinv = sales.sum / inventory.sum;
  endcomp;
  compute before _page_ / style={background=white
                        preimage="shoe_zoo.png"};
     line ' ';
  endcomp;
  compute after _page_ / style={background=white
                       preimage="shoe_zoo.png"};
     line ' ';
  endcomp;
run;

proc report data=shoes spanrows;
  title 'Ex 1b) COMPUTE BEFORE/AFTER _PAGE_';
  ** same code as Ex 1a **;
  compute before _page_ / style=Header{fontweight=bold fontsize=18pt};
     line "Shoe Zoo: We've got everything from boots to bunny slippers!";
  endcomp;
  compute after _page_ / style=Header{fontweight=bold fontsize=18pt};
     line "Shoe Zoo: We've got everything from boots to bunny slippers!";
  endcomp;
run;
```

Output 7 shows the results of using the COMPUTE BEFORE _PAGE_ and COMPUTE AFTER _PAGE_ statements to insert images, and Output 8 shows the insertion of captions. The image, in this case, is stored in the same location where the output file is stored. If the image were in a different location from the location of the HTML file or if you were using ODS PDF or ODS RTF, you would have to use a fully qualified pathname for the image location.


Ex 1a) COMPUTE BEFORE/AFTER _PAGE_

Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Asia	Boot	$62,708	$1,376	$61,332	$170,165	36.85%
	Sandal	$8,208	$225	$7,983	$36,570	22.44%
	Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
Asia		$222,948	$4,669	$218,279	$691,817	32.23%
Canada	Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
	Sandal	$14,798	$628	$14,170	$99,035	14.94%
	Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
Canada		$1,353,162	$41,303	$1,311,859	$4,921,656	27.49%
		$1,576,110	$45,972	$1,530,138	$5,613,473	28.08%

Output 7: Insert Images with COMPUTE BEFORE/AFTER _PAGE_


Ex 1b) COMPUTE BEFORE/AFTER _PAGE_

Shoe Zoo: We've got everything from boots to bunny slippers!

Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Asia	Boot	$62,708	$1,376	$61,332	$170,165	36.85%
	Sandal	$8,208	$225	$7,983	$36,570	22.44%
	Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
Asia		$222,948	$4,669	$218,279	$691,817	32.23%
Canada	Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
	Sandal	$14,798	$628	$14,170	$99,035	14.94%
	Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
Canada		$1,353,162	$41,303	$1,311,859	$4,921,656	27.49%
		$1,576,110	$45,972	$1,530,138	$5,613,473	28.08%

Shoe Zoo: We've got everything from boots to bunny slippers!

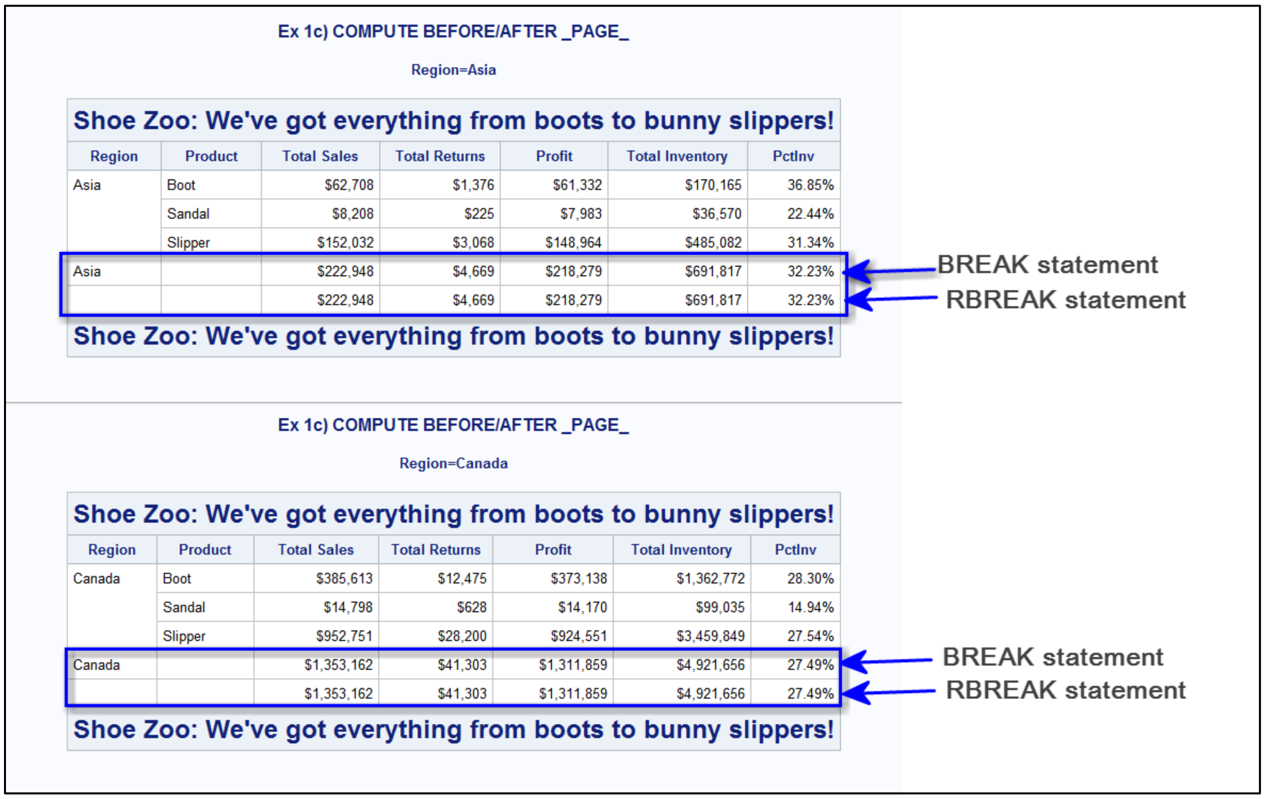
Output 8: Insert Captions with COMPUTE BEFORE/AFTER _PAGE_

When you use the PAGE option in PROC REPORT or BY-group processing, then the COMPUTE BEFORE _PAGE_ block will be executed for every page that is generated. For example, with only one change to the previous caption program, the results will be dramatically different because there will be a logical page (for HTML) that is created for every BY group. For RTF and PDF, there will be a physical page for every BY group unless the STARTPAGE=NO option is used.

When a logical or physical page starts, SAS will repeat all the PROC REPORT titles and the column headers, and the COMPUTE BEFORE _PAGE_ text. Here is the code:

```
proc report data=shoes spanrows;
by region;
  where region in ('Asia' 'Canada') and
        product in ('Boot' 'Sandal' 'Slipper');
  title 'Ex 1c) COMPUTE BEFORE/AFTER _PAGE_';
  column region product sales returns Profit inventory PctInv ;
  define region / group;
  define product / group;
  define profit / computed f=dollar14.;
  define pctinv / computed f=percent9.2;
  break after region/summarize;
  rbreak after / summarize;
  compute pctinv;
    profit = sum(sales.sum,-1*returns.sum);
    pctinv = sales.sum / inventory.sum;
  endcomp;
  compute before _page_ / style=Header{fontweight=bold fontsize=18pt};
     line "Shoe Zoo: We've got everything from boots to bunny slippers!";
  endcomp;
  compute after _page_ / style=Header{fontweight=bold fontsize=18pt};
     line "Shoe Zoo: We've got everything from boots to bunny slippers!";
  endcomp;
run;
```

In Output 9, you can tell that the output was created with BY-group processing because the BYLINE appears under the title, and outside the frame of the table.



Ex 1c) COMPUTE BEFORE/AFTER _PAGE_

Region=Asia

Shoe Zoo: We've got everything from boots to bunny slippers!						
Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Asia	Boot	$62,708	$1,376	$61,332	$170,165	36.85%
	Sandal	$8,208	$225	$7,983	$36,570	22.44%
	Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
Asia		$222,948	$4,669	$218,279	$691,817	32.23%
		$222,948	$4,669	$218,279	$691,817	32.23%
Shoe Zoo: We've got everything from boots to bunny slippers!						

Ex 1c) COMPUTE BEFORE/AFTER _PAGE_

Region=Canada

Shoe Zoo: We've got everything from boots to bunny slippers!						
Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Canada	Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
	Sandal	$14,798	$628	$14,170	$99,035	14.94%
	Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
Canada		$1,353,162	$41,303	$1,311,859	$4,921,656	27.49%
		$1,353,162	$41,303	$1,311,859	$4,921,656	27.49%
Shoe Zoo: We've got everything from boots to bunny slippers!						

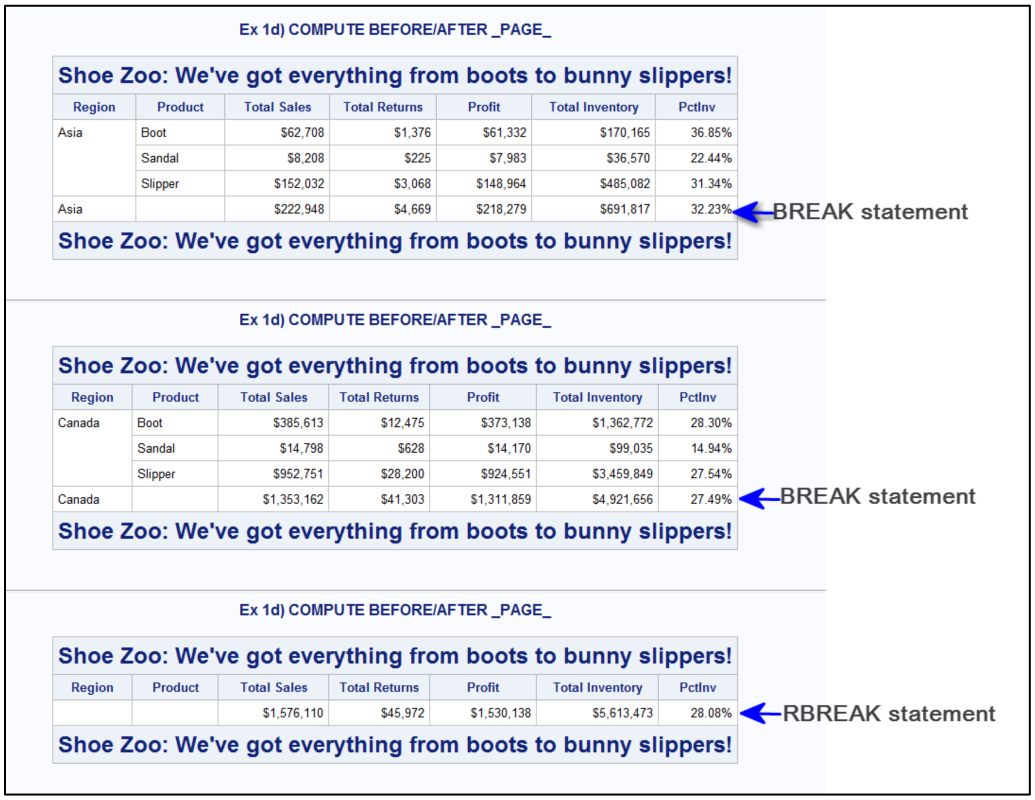
Output 9: Results with the BY Statement

When you use BY-group processing, note how each BY group is treated as a separate entity. The BREAK statement and the RBREAK statements are both executed for each BY group. Usually, if you are implementing BY-group processing, you would use either the BREAK statement or the RBREAK statement, but not both, depending on how you wanted the summary line to look. Another feature of the report is that, by default, PROC REPORT inserts the BYLINE information beneath the TITLE and above each logical page of output. In an HTML file, there is a logical page delineated by the gray horizontal rule line that appears between each BY-group's table. In RTF and PDF files, there would be a physical page between the tables.

When you use the PAGE option in the BREAK statement, you take advantage of PAGE processing, whereby PROC REPORT inserts a page break between the GROUP or ORDER variables. PROC REPORT puts a logical break in HTML output and a physical page break in RTF and PDF output. In the following code, the BY statement was removed and the PAGE option was used instead (Note that only the changes from the previous program are shown.)

```
proc report data=shoes spanrows;
  ** same WHERE statement for 2 regions and 3 products**;
  title 'Ex 1d) COMPUTE BEFORE/AFTER _PAGE_';
  ** same code as previous **;
  break after region/summarize page;
  ** same code as previous **;
run;
```

When you review Output 10, you will see that only the BREAK statement is executed for every value of the GROUP item, REGION. In addition, the RBREAK statement is correctly executed one time at the end of the report, and the grand total appears on a logical page by itself.



Ex 1d) COMPUTE BEFORE/AFTER _PAGE_

Shoe Zoo: We've got everything from boots to bunny slippers!

Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Asia	Boot	$62,708	$1,376	$61,332	$170,165	36.85%
	Sandal	$8,208	$225	$7,983	$36,570	22.44%
	Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
Asia		$222,948	$4,669	$218,279	$691,817	32.23%

Shoe Zoo: We've got everything from boots to bunny slippers!

Ex 1d) COMPUTE BEFORE/AFTER _PAGE_

Shoe Zoo: We've got everything from boots to bunny slippers!

Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Canada	Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
	Sandal	$14,798	$628	$14,170	$99,035	14.94%
	Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
Canada		$1,353,162	$41,303	$1,311,859	$4,921,656	27.49%

Shoe Zoo: We've got everything from boots to bunny slippers!

Ex 1d) COMPUTE BEFORE/AFTER _PAGE_

Shoe Zoo: We've got everything from boots to bunny slippers!

Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
		$1,576,110	$45,972	$1,530,138	$5,613,473	28.08%

Shoe Zoo: We've got everything from boots to bunny slippers!

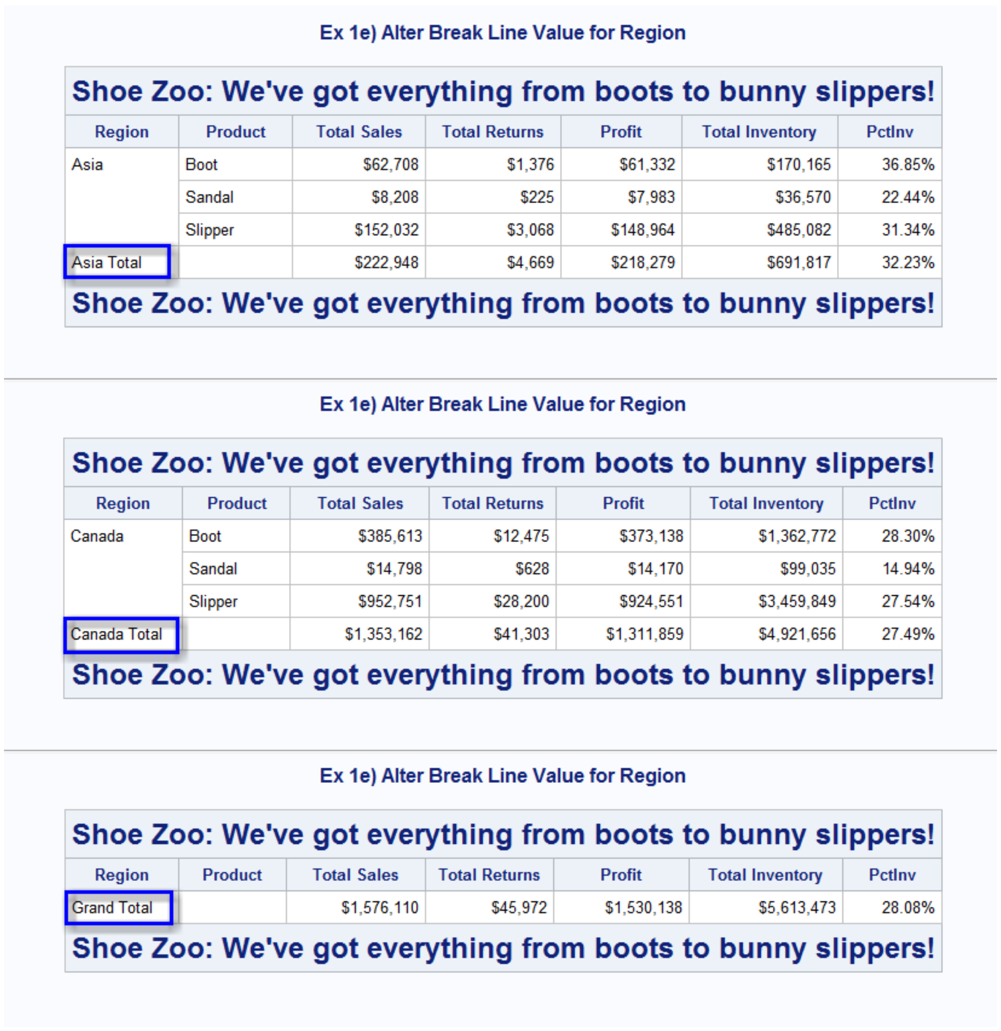
Output 10: Results with the PAGE Option in the BREAK Statement

One final modification will add a simple COMPUTE block to the code. But this time, instead of calculating a new variable, the desired change will be to have "Asia Total", "Canada Total", and "Grand Total" as the values that are displayed in the REGION cell on the summary line. This means that we want to add a simple COMPUTE block for REGION to our previous code with the PAGE option.

```
compute region;
   if upcase(_break_) = 'REGION' then do;
      region = catx(' ',region, 'Total');
     end;
     else if _break_ in ('_RBREAK_') then do;
       region = 'Grand Total';
     end;
endcomp;
```

Note how the CATX function is used to concatenate the value of REGION with the string 'Total'. At the _RBREAK_ value for the _BREAK_ automatic variable, the value that is assigned to REGION will be 'Grand Total' (instead of the usual blank value), as shown in Output 10). The point of this example is that you can use a COMPUTE block to alter the value that is displayed for an existing report item and that is based on a variable. Under usual circumstances, you would not alter the data cells on the report rows. But changing the value that is displayed at a break is one of the most common reasons for using a COMPUTE block to assign a new value to an item on the BREAK or RBREAK location.

The results of using the simple COMPUTE target are shown in Output 11.



Ex 1e) Alter Break Line Value for Region

Shoe Zoo: We've got everything from boots to bunny slippers!

Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Asia	Boot	$62,708	$1,376	$61,332	$170,165	36.85%
	Sandal	$8,208	$225	$7,983	$36,570	22.44%
	Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
Asia Total		$222,948	$4,669	$218,279	$691,817	32.23%

Shoe Zoo: We've got everything from boots to bunny slippers!

Ex 1e) Alter Break Line Value for Region

Shoe Zoo: We've got everything from boots to bunny slippers!

Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Canada	Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
	Sandal	$14,798	$628	$14,170	$99,035	14.94%
	Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
Canada Total		$1,353,162	$41,303	$1,311,859	$4,921,656	27.49%

Shoe Zoo: We've got everything from boots to bunny slippers!

Ex 1e) Alter Break Line Value for Region

Shoe Zoo: We've got everything from boots to bunny slippers!

Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Grand Total		$1,576,110	$45,972	$1,530,138	$5,613,473	28.08%

Shoe Zoo: We've got everything from boots to bunny slippers!

Output 11: Adding a COMPUTE REGION Block to Change the Summary Line

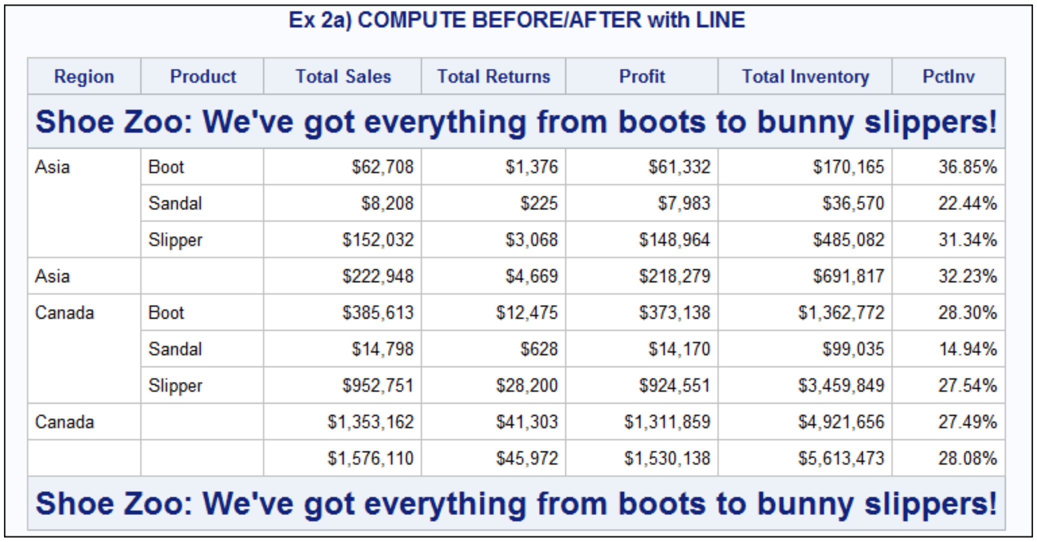
The next example will explore COMPUTE BEFORE and COMPUTE AFTER without using the _PAGE_ value as the target location.

EXAMPLE 2: COMPUTE BEFORE/AFTER

If you refer back to Output 6, the type of breaks in this section will be the ones numbered 2 and 5. The biggest difference is that the COMPUTE BEFORE _PAGE_ text is placed above the column headers, but the text from the COMPUTE BEFORE is beneath the column headers. This placement of COMPUTE BEFORE text makes this type of COMPUTE block fall out of favor quickly. Most people tend to use COMPUTE AFTER more than COMPUTE BEFORE. To show what will happen with COMPUTE BEFORE and COMPUTE AFTER, consider this code:

```
proc report data=shoes spanrows;
  title 'Ex 2a) COMPUTE BEFORE/AFTER with LINE';
  column region product sales returns Profit inventory PctInv ;
  define region / group;
  define product / group;
  define profit / computed f=dollar14.;
  define pctinv / computed f=percent9.2;
  break after region/summarize;
  rbreak after / summarize;
  compute pctinv;
    profit = sum(sales.sum,-1*returns.sum);
    pctinv = sales.sum / inventory.sum;
  endcomp;
  compute before  / style=Header{fontweight=bold fontsize=18pt};
     line "Shoe Zoo: We've got everything from boots to bunny slippers!";
  endcomp;
  compute after  / style=Header{fontweight=bold fontsize=18pt};
     line "Shoe Zoo: We've got everything from boots to bunny slippers!";
  endcomp;
run;
```

This program uses the same WHERE statement as the previous example to filter the data for two regions and three products. The results are shown in Output 12. In the output, the LINE output from the COMPUTE BEFORE statement looks out of place when it is between the column headers and the first report row. But the COMPUTE AFTER text looks identical to the Output 8 results for the COMPUTE AFTER _PAGE_ location.



Ex 2a) COMPUTE BEFORE/AFTER with LINE

Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Shoe Zoo: We've got everything from boots to bunny slippers!						
Asia	Boot	$62,708	$1,376	$61,332	$170,165	36.85%
	Sandal	$8,208	$225	$7,983	$36,570	22.44%
	Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
Asia		$222,948	$4,669	$218,279	$691,817	32.23%
Canada	Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
	Sandal	$14,798	$628	$14,170	$99,035	14.94%
	Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
Canada		$1,353,162	$41,303	$1,311,859	$4,921,656	27.49%
		$1,576,110	$45,972	$1,530,138	$5,613,473	28.08%
Shoe Zoo: We've got everything from boots to bunny slippers!						

Output 12: Using COMPUTE BEFORE and COMPUTE AFTER without _PAGE_

A useful feature of COMPUTE AFTER is the ability to insert custom rows of information, based on the data. What if you wanted to see the overall averages for all the regions that appeared on the report, whether there were two regions or ten regions? What if you still wanted the grand total break line to appear at the bottom of the report rows?

Both types of break information can be added to the report. But to do this, you'll need to modify the code slightly. To start, the report will need some helper variables, which declared as alias items in the COLUMN statement:

```
column region product sales returns Profit inventory PctInv
       sales=salesmn returns=retmn inventory=invmn;
```

In this COLUMN statement, the variables for SALES, RETURNS, and INVENTORY are used a second time on the report and are assigned aliases because the MEAN statistic will be requested for each of these three items. The results that are created with this revised COLUMN statement appear in Output 13.

Each of these three report items has been defined as an analysis item with a statistic of MEAN. Then the NOPRINT option has also been specified in each DEFINE statement to prevent the columns from appearing on the final report, but still allowing their values to be used in the COMPUTE AFTER break processing. Only the changed code or the new code is shown for this program, as follows:

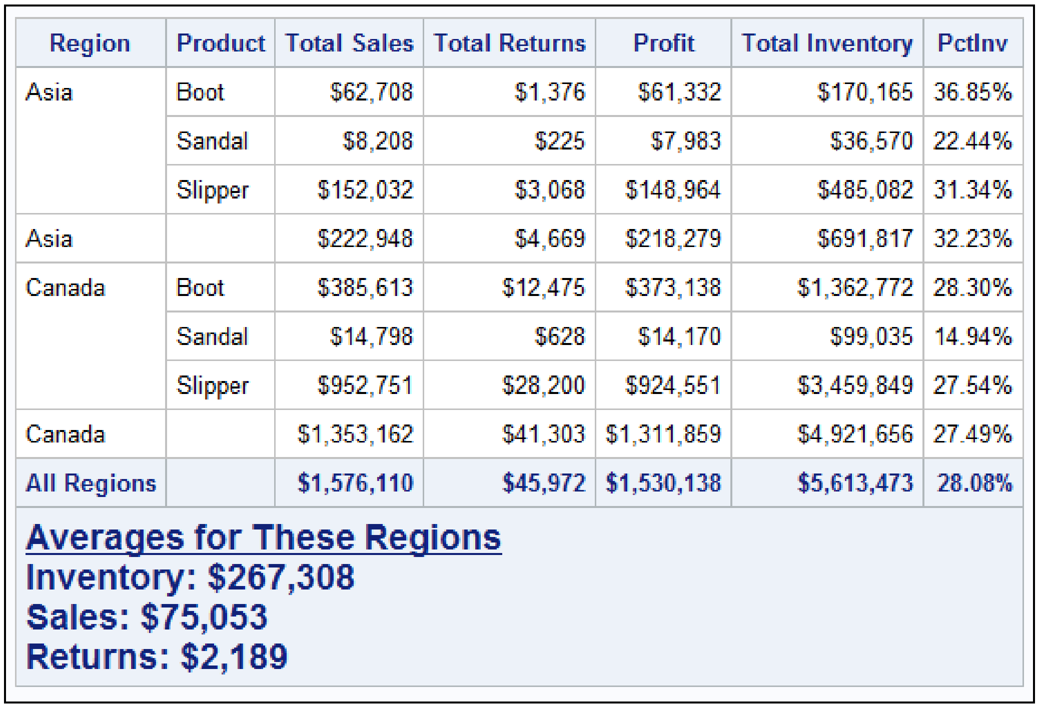
```
ods escapechar='^';
proc report data=shoes spanrows;
  column region product sales returns Profit inventory PctInv
         sales=salesmn returns=retmn inventory=invmn;
  *** same code as previous program ***;
  define salesmn / mean noprint;
  define retmn / mean noprint;
  define invmn / mean noprint;
  break after region/summarize;
  rbreak after / summarize  style=Header;
  *** same COMPUTE block for PCTINV ***;
  compute after  / style=Header{just=l fontweight=bold fontsize=14pt};
    region = 'All Regions';
    cinv = left(put(invmn,dollar10.));
    csal = left(put(salesmn,dollar10.));
    cret = left(put(retmn,dollar10.));
    line "^{style[textdecoration=underline]Averages for These Regions}";
    line "Inventory: " cinv $varying50.;
    line "Sales: " csal $varying50.;
    line "Returns: " cret $varying50.;
  endcomp;
run;
```

Notice how the text "All Regions" has been assigned to the value for REGION at the report break. In addition, the style for the summary line is set to a header in the RBREAK statement.

In the COMPUTE AFTER statement, the style of the COMPUTE AFTER line text is changed after the slash (/). A default value of Header is specified for these lines, with overrides for the justification, font weight, and font size attributes. Underlining for the text "Averages for These Regions" was accomplished through the use of ODS ESCAPECHAR syntax and the TEXTDECORATION style attribute.

Finally, temporary character variables are created to be formatted character versions of the SALESMN, RETMN, and INVMN aliases. The character strings that result from the PUT function are left-justified so that, when written to the results file, the formatted dollar values will be padded on the right with blanks instead of being padded on the left.

The benefit of using the aliases and the NOPRINT techniques means that a prior summarization step with PROC MEANS or PROC TABULATE is not necessary in order to calculate the averages for the final LINE statements. Because the summary values for the aliases are available at the COMPUTE AFTER location, they can be written, formatted, or used in other calculations, at the end of the report, as desired.



Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Asia	Boot	$62,708	$1,376	$61,332	$170,165	36.85%
	Sandal	$8,208	$225	$7,983	$36,570	22.44%
	Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
Asia		$222,948	$4,669	$218,279	$691,817	32.23%
Canada	Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
	Sandal	$14,798	$628	$14,170	$99,035	14.94%
	Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
Canada		$1,353,162	$41,303	$1,311,859	$4,921,656	27.49%
All Regions		**$1,576,110**	**$45,972**	**$1,530,138**	**$5,613,473**	**28.08%**

Averages for These Regions
Inventory: $267,308
Sales: $75,053
Returns: $2,189

Output 13: Using COMPUTE AFTER with Customized Text

Here is one final note about using COMPUTE AFTER and either BY-group processing or PAGE processing in the BREAK statement. Consider this BREAK statement:

```
break after region / summarize page;
```

The COMPUTE AFTER line output will appear only one time on the report: at the end of the report on a logical page by itself. However, with BY-group processing, every BY group is treated as a complete entity, which means that for every BY group, the COMPUTE AFTER line output will appear at the bottom of the BY group, as shown in Output 9. This is useful information to keep in mind when you are trying to decide whether to use PAGE or BY.

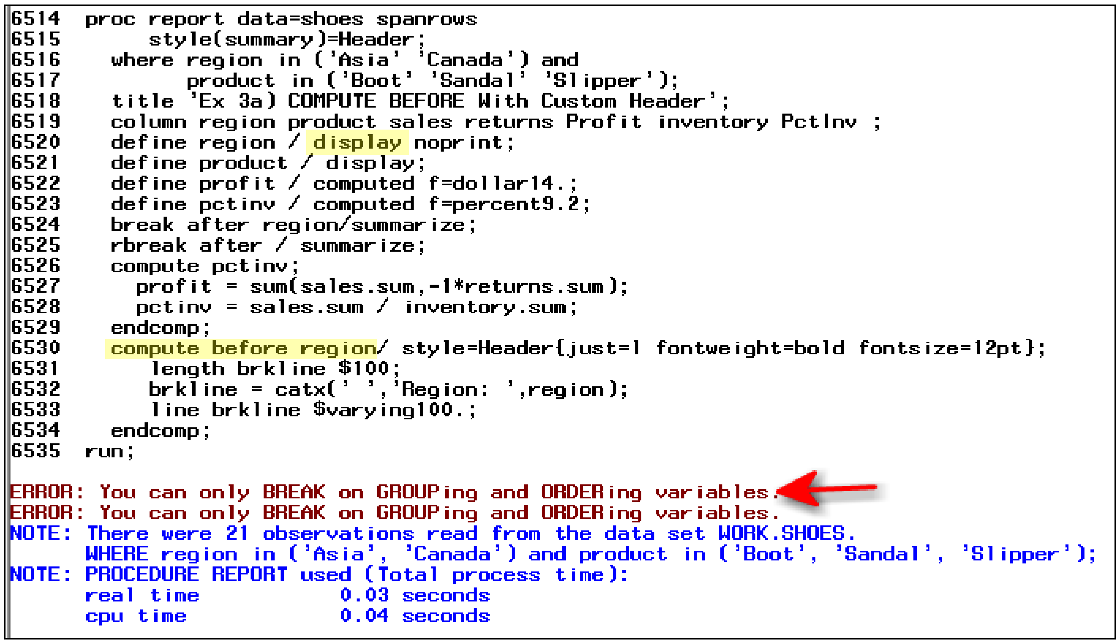
Next, we'll move to a discussion about the most commonly used form of BREAK processing and how to write extra lines with a *target-variable* that is used with both COMPUTE BEFORE and COMPUTE AFTER.

EXAMPLE 3: COMPUTE BEFORE AND COMPUTE AFTER *TARGET-VARIABLE*

When you use a target variable in your COMPUTE statement, you must also have defined the target variable as either a GROUP or ORDER usage. Break processing before or after a report group in PROC REPORT requires one of these usages. If you do not specify a usage for a character variable, the default is DISPLAY. Consider this combination of code in a full program

```
define region / display noprint;
define region /  noprint;
compute before region;
compute after region;
```

You would get an error message in the log, as shown in Output 14. To avoid this error, I always recommend that you explicitly list a usage for your DEFINE statements instead of taking the default. Considering usage (DISPLAY, ORDER, GROUP, ACROSS, COMPUTED)and explicitly coding the default or an alternative usage indicates that you've planned your report and that you understand the impact of usage on the resulting report. It also indicates that you understand the options for BREAK processing and COMPUTE processing that are available for that program.



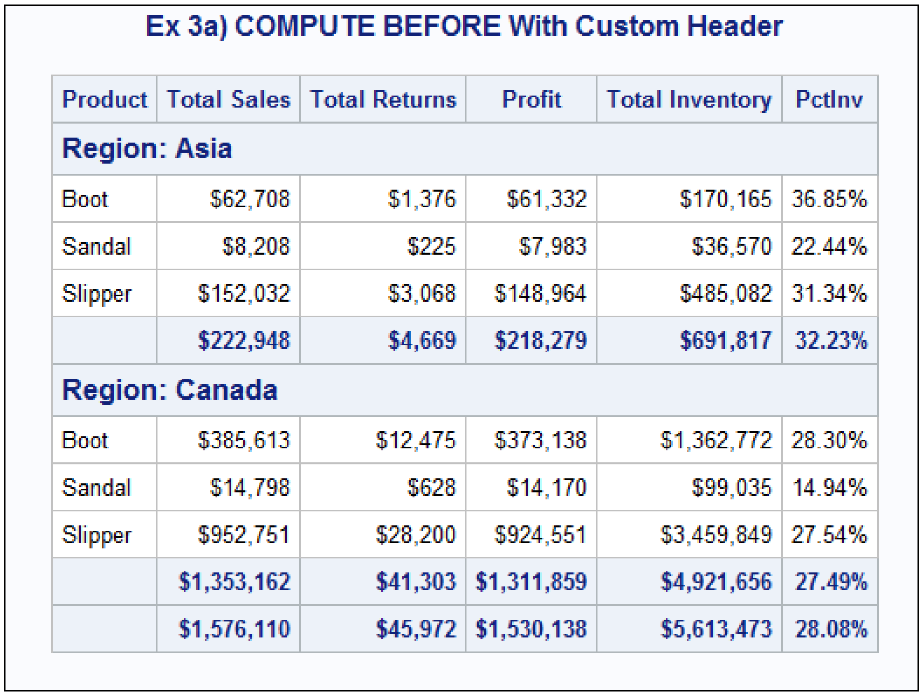
```
6514  proc report data=shoes spanrows
6515       style(summary)=Header;
6516    where region in ('Asia' 'Canada') and
6517          product in ('Boot' 'Sandal' 'Slipper');
6518    title 'Ex 3a) COMPUTE BEFORE With Custom Header';
6519    column region product sales returns Profit inventory PctInv ;
6520    define region / display noprint;
6521    define product / display;
6522    define profit / computed f=dollar14.;
6523    define pctinv / computed f=percent9.2;
6524    break after region/summarize;
6525    rbreak after / summarize;
6526    compute pctinv;
6527      profit = sum(sales.sum,-1*returns.sum);
6528      pctinv = sales.sum / inventory.sum;
6529    endcomp;
6530    compute before region/ style=Header{just=l fontweight=bold fontsize=12pt};
6531       length brkline $100;
6532       brkline = catx(' ','Region: ',region);
6533       line brkline $varying100.;
6534    endcomp;
6535  run;

ERROR: You can only BREAK on GROUPing and ORDERing variables.
ERROR: You can only BREAK on GROUPing and ORDERing variables.
NOTE: There were 21 observations read from the data set WORK.SHOES.
      WHERE region in ('Asia', 'Canada') and product in ('Boot', 'Sandal', 'Slipper');
NOTE: PROCEDURE REPORT used (Total process time):
      real time           0.03 seconds
      cpu time            0.04 seconds
```

Output 14: Log Displays Error Message with Wrong Usage

So consider this code that uses REGION appropriately as a GROUP item on the report:

```
proc report data=shoes spanrows
     style(summary)=Header;
  column region product sales returns Profit inventory PctInv ;
  define region / group noprint;
  define product / group;
  define profit / computed f=dollar14.;
  define pctinv / computed f=percent9.2;
  break after region/summarize;
  rbreak after / summarize;
  compute pctinv;
    profit = sum(sales.sum,-1*returns.sum);
    pctinv = sales.sum / inventory.sum;
  endcomp;
  compute before region/
          style=Header{just=l fontweight=bold fontsize=12pt};
     length brkline $100;
     brkline = catx(' ','Region: ',region);
     line brkline $varying100.;
  endcomp;
run;
```

Results are shown in Output 15: Notice how the COMPUTE block for REGION is executed before the first report row of the group is placed on the report. The COMPUTE BEFORE statement for the first group will always appear between the column headers and the first row of the group. In this case, in addition to being declared as a GROUP usage, the REGION variable is declared as a NOPRINT item, as well. This means that REGION can be used, such as to create the temporary report variable BRKLINE, but the REGION value will not be displayed on every report row. The temporary variable BRKLINE is used in the COMPUTE block to hold the string "Region: Asia" or "Region: Canada", which will be displayed at the top of each REGION group. Notice that a LENGTH statement is used to explicitly set the length of the temporary variable. Since we do not know what the length of each region is, ahead of time, it is considered a best practice to provide a LENGTH statement for any character variables that are used as temporary variables in PROC REPORT Compute blocks.



Ex 3a) COMPUTE BEFORE With Custom Header

Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Region: Asia					
Boot	$62,708	$1,376	$61,332	$170,165	36.85%
Sandal	$8,208	$225	$7,983	$36,570	22.44%
Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
	$222,948	**$4,669**	**$218,279**	**$691,817**	**32.23%**
Region: Canada					
Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
Sandal	$14,798	$628	$14,170	$99,035	14.94%
Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
	$1,353,162	**$41,303**	**$1,311,859**	**$4,921,656**	**27.49%**
	$1,576,110	**$45,972**	**$1,530,138**	**$5,613,473**	**28.08%**

Output 15: Results with COMPUTE BEFORE REGION Using a Target Variable

Because the PROC REPORT statement had a style override for the SUMMARY location on the report, both of the subtotal rows and the grand total row are given the same style characteristics as the column headers. The custom line above the group comes from the COMPUTE BEFORE REGION COMPUTE block and the LINE statement. The subtotal and grand total rows come from the BREAK AFTER REGION statement. At this point, COMPUTE AFTER is not being used.

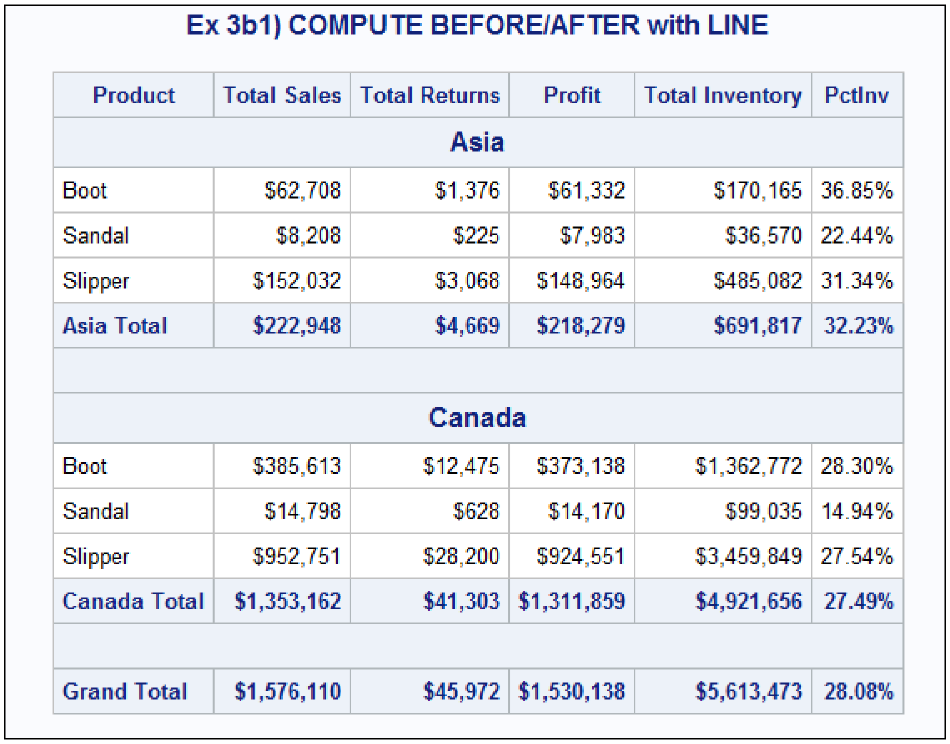
The next example will introduce COMPUTE AFTER REGION to the report. When PROC REPORT was originally written for the LISTING destination, the BREAK statement used a SKIP option to insert a blank line between each GROUP or ORDER section on the report. However, the SKIP option is not supported by ODS destinations such as RTF, PDF, or HTML. Instead, the method that is used to replicate the SKIP behavior for all destinations, except LISTING, is the COMPUTE AFTER *target-variable* construct. Here is an example:

```
compute after region/ style=Header;
   line ' ';
endcomp;
```

The results of this COMPUTE AFTER block, in conjunction with the BREAK statement, produces the same output as SKIP for the LISTING destination. In the code, the COMPUTE BEFORE block was changed as well. Instead of putting "Region:" in front of the value for each region, you can use a simple LINE statement to write out the value of REGION at the break with this code:

```
compute before region/
        style=Header{just=c fontweight=bold fontsize=12pt};
   line region $varying100.;
endcomp;
```

The results of using this code that was changed from the previous program is shown in Output 16.



Ex 3b1) COMPUTE BEFORE/AFTER with LINE

Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Asia					
Boot	$62,708	$1,376	$61,332	$170,165	36.85%
Sandal	$8,208	$225	$7,983	$36,570	22.44%
Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
Asia Total	**$222,948**	**$4,669**	**$218,279**	**$691,817**	**32.23%**
Canada					
Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
Sandal	$14,798	$628	$14,170	$99,035	14.94%
Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
Canada Total	**$1,353,162**	**$41,303**	**$1,311,859**	**$4,921,656**	**27.49%**
Grand Total	**$1,576,110**	**$45,972**	**$1,530,138**	**$5,613,473**	**28.08%**

Output 16: COMPUTE AFTER Inserts Blank Line between Groups

How do you feel about the blank line between the “Canada Total" and the “Grand Total” row? Some people like it. Some people don't like it. Output 16 shows how you can replicate the SKIP behavior if you want.

Output 17 shows that it is possible to suppress LINE output when you want to do that. The key to suppressing LINE output is to understand that PROC REPORT always writes LINE output. It does not execute LINE statements conditionally. But when you tell PROC REPORT to write the LINE with a formatted length of 0, PROC REPORT is smart enough to suppress writing the LINE. This is accomplished for Output 17.

Output 17 uses the same WORK.SHOES subset for Asia and Canada as the previous example. In this subset of the data, the only blank line to be suppressed is the one for the Canada group, which is the last group. This means that for Canada rows, when break processing happens, the length of the line needs to be assigned a value of 0. This changed code will accomplish that goal:

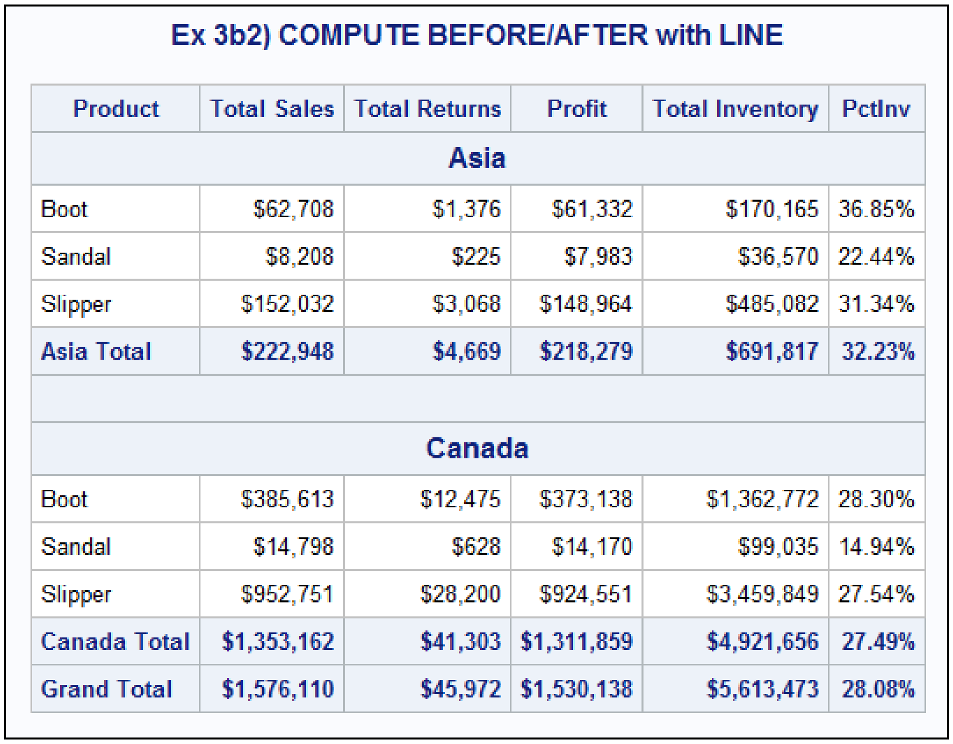
```
proc report data=shoes spanrows
     style(summary)=Header;
  *** column and define statements same as previous code ***;
  *** compute block for pctinv same as previous code ***;
  break after region/summarize;
  rbreak after / summarize;
  compute before region/
          style=Header{just=c fontweight=bold fontsize=12pt};
     line region $varying100.;
     brktxt = ' ';
     if region = 'Canada' then reglg = 0;
     else reglg = 1;
  endcomp;
  compute after region/ style=Header;
     line brktxt $varying. reglg;
  endcomp;
```

```
    compute product;
      if upcase(_break_) = 'REGION' then product=catx(' ',region,'Total');
      else if _break_ = '_RBREAK_' then product = 'Grand Total';
    endcomp;
  run;
```

Notice how COMPUTE BEFORE REGION is doing some extra work. Although the LINE statement is the same as shown previously, there are some added statements in the COMPUTE block that test the value of REGION. First, the temporary variable, BRKTXT, is created to hold the single blank. Then, the value of REGION is tested. If the value of REGION is CANADA, which is the last REGION for this subset, then the length of the temporary variable, REGLG, is assigned a value of 0. Otherwise, the temporary variable, REGLG, is assigned a length of 1. For this program, the test for the last REGION value was hardcoded. But SAS macro programmers know that you can create a macro variable to test the value instead.

In the program that produced Output 16, the $VARYING format was used with a hard-coded length of 100 ($VARYING100.). The syntax for the $VARYING format allows you to provide a length that can be considered a "maximum number of characters" (such as 100), when you always want to write a value, but are not sure ahead of time what the maximum length of the variable is. However, you can also provide a variable name with $VARYING., as shown above. This gives you more flexibility to control the length of what is written at the break point and even more flexibility because a length variable with a value of 0 will cause the LINE statement output to be suppressed.

Notice how the COMPUTE AFTER REGION block is now simplified. The LINE statement will always execute after the BREAK has been performed for REGION. But since the text and the length of the text were set in the COMPUTE BEFORE REGION block, the COMPUTE AFTER REGION needs a LINE statement along with the $VARYING. format and the REGLG variable for providing the length to write. As described, if the length to write is 0, then the LINE output will be suppressed. Notice that the LINE output spans the width of the entire table. The ODS destinations, except LISTING, do not provide accurate alignment of information beneath specific columns because they use proportional fonts to write output. Also, LINE commands with pointer controls such as @15 or @30 are impractical because they do not act the same for proportionally spaced fonts as they do for the LISTING destination.



Ex 3b2) COMPUTE BEFORE/AFTER with LINE

Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Asia					
Boot	$62,708	$1,376	$61,332	$170,165	36.85%
Sandal	$8,208	$225	$7,983	$36,570	22.44%
Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
Asia Total	**$222,948**	**$4,669**	**$218,279**	**$691,817**	**32.23%**
Canada					
Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
Sandal	$14,798	$628	$14,170	$99,035	14.94%
Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
Canada Total	**$1,353,162**	**$41,303**	**$1,311,859**	**$4,921,656**	**27.49%**
Grand Total	**$1,576,110**	**$45,972**	**$1,530,138**	**$5,613,473**	**28.08%**

Output 17: Suppressing LINE Statement Output

The last piece of business is to add some identifying text on the summary lines. In Output 13, this was done in the COMPUTE AFTER block. However, in Output 17, it looks better with the text for PRODUCT changed to be either “Asia Total” or “Canada Total” for the BREAK AFTER REGION rows and if the text for PRODUCT was changed to “Grand Total” for the RBREAK AFTER row. Both changes are accomplished in the COMPUTE block for PRODUCT with a test for the value of _BREAK_, which is the automatic variable that PROC REPORT creates when BREAK processing is performed in a PROC REPORT step. Just remember that the length of the variable that is assigned a value, in this case, PRODUCT, must be wide enough to accommodate the text string for display at the breakpoint.

The ability to create a temporary report variable in the COMPUTE BEFORE *target-variable* in the COMPUTE block opens up more possibilities for reporting than what was illustrated in Output 17. Because these temporary variables retain their values until you reset the values, more complex break processing is possible. The next section shows some examples that use temporary variables.

EXAMPLE 4: MORE COMPLEX BREAK PROCESSING

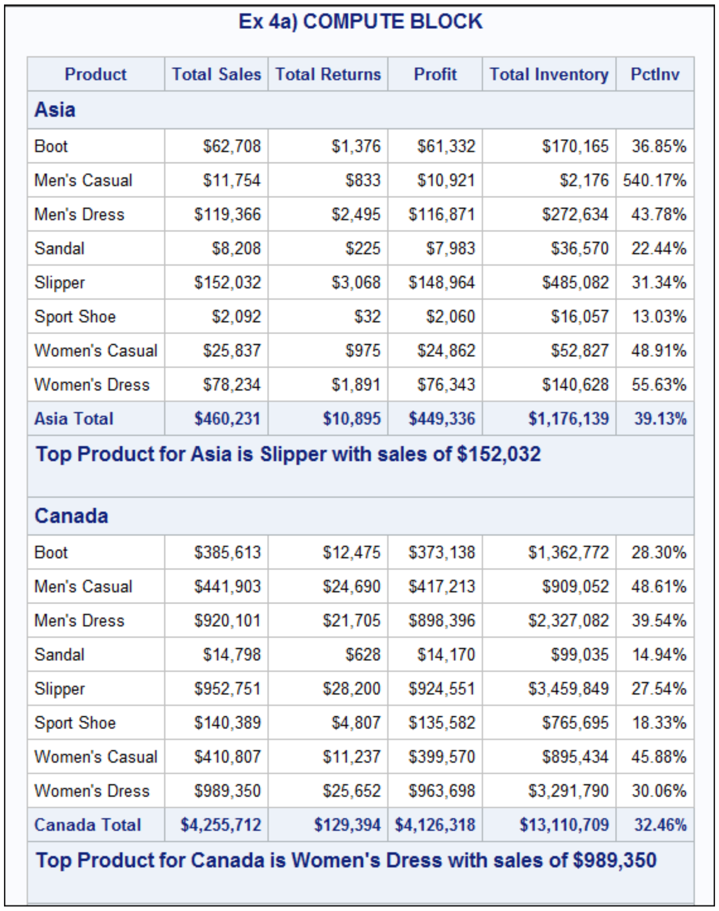
What if the desired report uses data-driven information at a break? For example, you need to generate a summary report showing sales for each PRODUCT and REGION. But for every region, you need to write a summary line that essentially highlights the top product that is sold for the region. One way to approach this task would be to presummarize the data using PROC MEANS or PROC SQL and then to create macro variables for the top product for each region. There is no need to do this preprocessing if you tap into the power of the COMPUTE block and the ability to use temporary variables.

Consider this code:

```
proc report data=shoes spanrows
     style(summary)=Header;
  *** same COLUMN and DEFINE statements as previous ***;
  break after region/summarize;
  rbreak after / summarize;
  compute pctinv;
    profit = sum(sales.sum,-1*returns.sum);
    pctinv = sales.sum / inventory.sum;
  endcomp;
  compute before region  /
          style=Header{just=l fontweight=bold fontsize=12pt};
      topprod = .;
      thisregion = region;
      line Region $varying100.;
  endcomp;
  compute sales;
    if _break_ = ' ' then do;
      if sales.sum gt topprod then do;
         topprod = sales.sum;
         holdprod = Product;
      end;
    end;
  endcomp;
  compute product;
    if upcase(_break_) = 'REGION' then product=catx(' ',region,'Total');
    else if _break_ = '_RBREAK_' then product = 'Grand Total';
  endcomp;
  compute after region  /
          style=Header{just=l fontweight=bold fontsize=12pt};
     brkline=catx(' ','Top Product for', thisregion, 'is',holdprod,
                  'with sales of',put(topprod,dollar14.));
     line brkline $varying100.;
     line ' ';
  endcomp;
run;
```

Partial output is shown in Output 18. In the COMPUTE BEFORE REGION block, the temporary variable TOPPROD is initialized to missing at the start (BEFORE) of each region. TOPPROD will hold the highest SALES amount for one PRODUCT for this region. HOLDPROD will hold the name of the PRODUCT with the highest sales. So, in the COMPUTE SALES block, the current value of SALES.SUM is compared to the retained value of TOPPROD. The comparison is done only on the report rows that are based on data rather than on the BREAK rows. If the value for SALES.SUM on the current report row is greater than the currently held value for TOPPROD, then the name of the Product on this report row is assigned to another temporary variable called HOLDPROD. The test for the _BREAK_ variable prevents the grand total or the region total from being considered the highest value.

This comparison will be done on every product for a single region. By the end of the group for one region, the variable TOPPROD will hold the highest SALES value, and the variable THISREGION will hold the name of the current REGION group that is being processed and HOLDPROD will hold the name of the highest PRODUCT sales for this region. This is the information that is used in the COMPUTE AFTER REGION block when the temporary BRKLINE is created using the CATX function. Once the BRKLINE variable is created, the last thing that happens in the COMPUTE AFTER REGION block is the LINE statement for BRKLINE and the LINE statement put a blank line beneath the custom text under the REGION summary.



Ex 4a) COMPUTE BLOCK

Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Asia					
Boot	$62,708	$1,376	$61,332	$170,165	36.85%
Men's Casual	$11,754	$833	$10,921	$2,176	540.17%
Men's Dress	$119,366	$2,495	$116,871	$272,634	43.78%
Sandal	$8,208	$225	$7,983	$36,570	22.44%
Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
Sport Shoe	$2,092	$32	$2,060	$16,057	13.03%
Women's Casual	$25,837	$975	$24,862	$52,827	48.91%
Women's Dress	$78,234	$1,891	$76,343	$140,628	55.63%
Asia Total	**$460,231**	**$10,895**	**$449,336**	**$1,176,139**	**39.13%**
Top Product for Asia is Slipper with sales of $152,032					
Canada					
Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
Men's Casual	$441,903	$24,690	$417,213	$909,052	48.61%
Men's Dress	$920,101	$21,705	$898,396	$2,327,082	39.54%
Sandal	$14,798	$628	$14,170	$99,035	14.94%
Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
Sport Shoe	$140,389	$4,807	$135,582	$765,695	18.33%
Women's Casual	$410,807	$11,237	$399,570	$895,434	45.88%
Women's Dress	$989,350	$25,652	$963,698	$3,291,790	30.06%
Canada Total	**$4,255,712**	**$129,394**	**$4,126,318**	**$13,110,709**	**32.46%**
Top Product for Canada is Women's Dress with sales of $989,350					

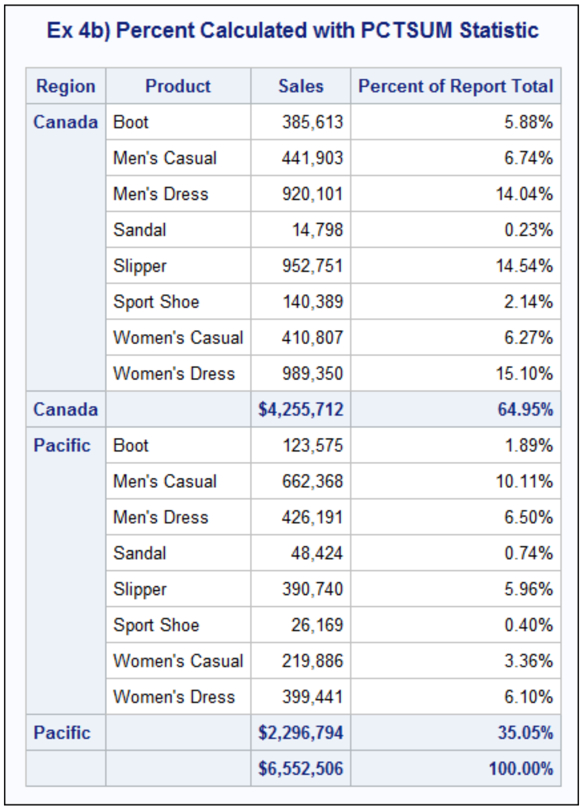
Output 18: Custom Break Line Added after Each Region

Throughout this paper, so far, the PCTINV report item has been a COMPUTED item with a custom percentage. Although PROC TABULATE wins the award for having the most choices for calculating percents, PROC REPORT runs a close second because PROC REPORT supports the PCTN and PCTSUM statistics. But the ability to perform custom calculations enables you to generate values, such as custom percentages or the percent of a group.

Before jumping into a custom calculation for a percent, let's see the code that uses PCTSUM as the statistic for a numeric variable:

```
proc report data=pctshoes nowd spanrows
  style(summary)=Header;
  column region product Sales Sales=psal;
  define region / group 'Region' f=$25.
         style(column)=Header;
  define product / group 'Product';
  define Sales / sum 'Sales' f=comma14.;
  define psal / pctsum 'Percent of Report Total' f=percent9.2;
  break after region / summarize;
  rbreak after / summarize;
  compute sales;
     if upcase(_break_) in ('REGION' '_RBREAK_') then do;
       call define(_col_,'format','dollar15.');
     end;
  endcomp;
run;
```

This code is very straightforward. The alias item, PSAL, uses SALES a second time on the report, requesting the PCTSUM statistic for the column. PROC REPORT does not multiply percents by 100, so the regular PERCENT format is specified for that column. The COMMA14. format is used for the SALES column, but a CALL DEFINE statement changes the format to a DOLLAR format in the BREAK lines.



Ex 4b) Percent Calculated with PCTSUM Statistic

Region	Product	Sales	Percent of Report Total
Canada	Boot	385,613	5.88%
	Men's Casual	441,903	6.74%
	Men's Dress	920,101	14.04%
	Sandal	14,798	0.23%
	Slipper	952,751	14.54%
	Sport Shoe	140,389	2.14%
	Women's Casual	410,807	6.27%
	Women's Dress	989,350	15.10%
Canada		$4,255,712	64.95%
Pacific	Boot	123,575	1.89%
	Men's Casual	662,368	10.11%
	Men's Dress	426,191	6.50%
	Sandal	48,424	0.74%
	Slipper	390,740	5.96%
	Sport Shoe	26,169	0.40%
	Women's Casual	219,886	3.36%
	Women's Dress	399,441	6.10%
Pacific		$2,296,794	35.05%
		$6,552,506	100.00%

Output 19: Results Using PCTSUM for the SALES Variable

If you want to show two percents on the report, you use the simple PCTSUM, as shown in Output 19, as well as the percent of the REGION. This means that on every row, the SALES value needs to be divided by the REGION group's total sales. Again, temporary variables will help do this calculation.

Here's the code that illustrates how to calculate the percents in a custom COMPUTE block:

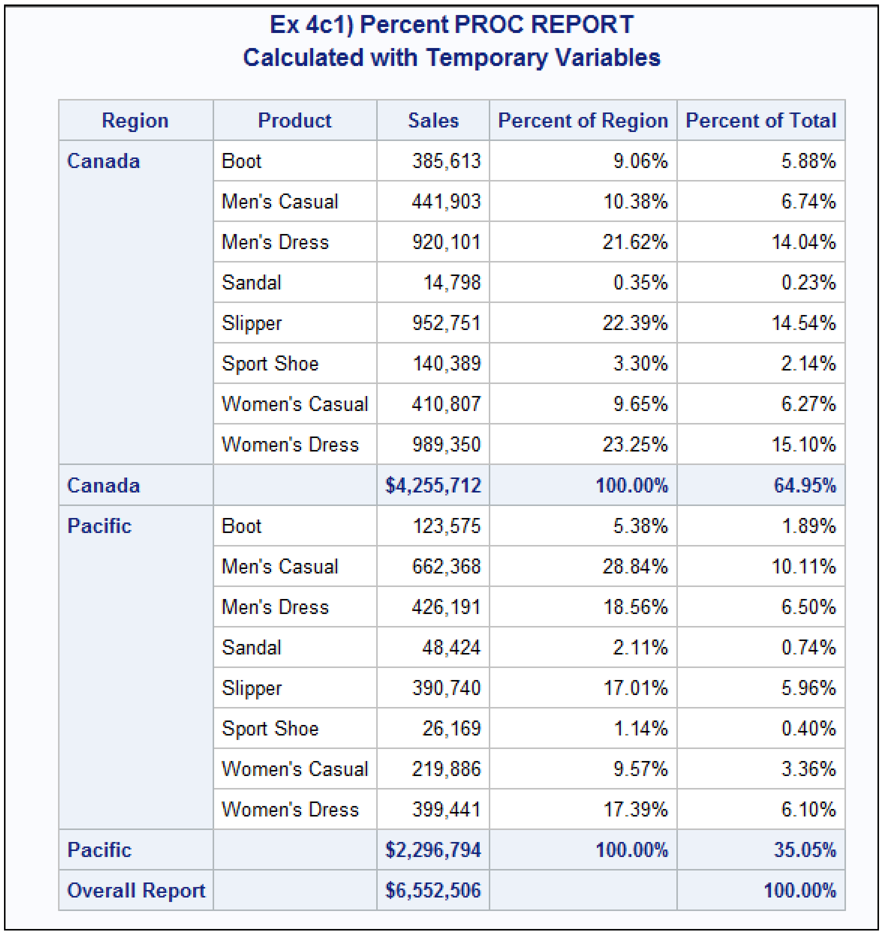
```
proc report data=pctshoes nowd spanrows
  style(summary)=Header;
  title 'Ex 4c1) Percent PROC REPORT';
  title2 'Calculated with Temporary Variables';
  column region product Sales PctGrp Sales=psal;
  define region / group 'Region' f=$25.
         style(column)=Header;
  define product / group 'Product';
  define Sales / sum 'Sales' f=comma14.;
  define pctgrp / computed 'Percent of Region' f=percent9.2;
  define psal / pctsum 'Percent of Total' f=percent9.2;

  break after region / summarize;
  rbreak after / summarize;

  compute before region;
     grptot=Sales.sum;
  endcomp;

  compute PctGrp;
     PctGrp=Sales.sum / grptot;
  endcomp;

  compute sales;
     if upcase(_break_) in ('REGION' '_RBREAK_') then do;
       call define(_col_,'format','dollar15.');
     end;
  endcomp;
  compute after;
     PctGrp = .;
     region = 'Overall Report';
  endcomp;
run;
```

In this code, there are only a few changes that are needed in order to calculate the custom percent of each sale for a region. In a COMPUTE BEFORE REGION block, a temporary variable named GRPTOT is created and is assigned the summary value for SALES.SUM. Because PROC REPORT summarizes the data before it starts writing the report rows, the summary for each REGION is available at the beginning of the group rows and at the end of the group rows, too.

Next, there is a COMPUTE block for the PCTGRP report item, which appears in the COLUMN statement and also in the DEFINE statement with a usage of COMPUTED. The value for PCTGRP is calculated as the SALES.SUM divided by the temporary variable GRPTOT.

In the COMPUTE SALES block, a CALL DEFINE statement will override the default format for the SALES value. On most of the report rows, the COMMA14 format will be used. But for the summary rows, which are identified by testing the automatic _BREAK_ variable, the format will be changed to the DOLLAR15 format. This provides the flexibility to have one format for most of the rows, and then to change the format at the BREAK location according to some other criterion.

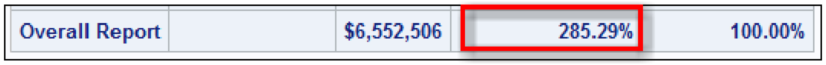
On the "grand total" row or on the final row written by the RBREAK statement, the value for REGION, which is normally a blank for the RBREAK row, will be "Overall Report". Also, the PCTGRP column will be assigned a missing value. The MISSING=' ' system option was set before this code ran, and so instead of seeing a '.' on the final row for PCTGRP, the cell just appears as a blank, as shown in Output 20. If PCTGRP were not assigned a missing value, the calculation for PCTGRP would be performed at the end of the report, resulting in an undesirable percentage value, as shown in Output 21.



Ex 4c1) Percent PROC REPORT
Calculated with Temporary Variables

Region	Product	Sales	Percent of Region	Percent of Total
Canada	Boot	385,613	9.06%	5.88%
	Men's Casual	441,903	10.38%	6.74%
	Men's Dress	920,101	21.62%	14.04%
	Sandal	14,798	0.35%	0.23%
	Slipper	952,751	22.39%	14.54%
	Sport Shoe	140,389	3.30%	2.14%
	Women's Casual	410,807	9.65%	6.27%
	Women's Dress	989,350	23.25%	15.10%
Canada		$4,255,712	100.00%	64.95%
Pacific	Boot	123,575	5.38%	1.89%
	Men's Casual	662,368	28.84%	10.11%
	Men's Dress	426,191	18.56%	6.50%
	Sandal	48,424	2.11%	0.74%
	Slipper	390,740	17.01%	5.96%
	Sport Shoe	26,169	1.14%	0.40%
	Women's Casual	219,886	9.57%	3.36%
	Women's Dress	399,441	17.39%	6.10%
Pacific		$2,296,794	100.00%	35.05%
Overall Report		$6,552,506		100.00%

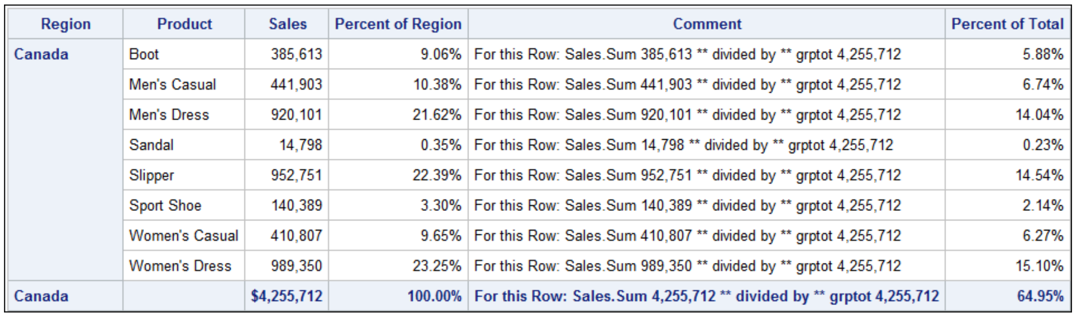
Output 20: Custom Percent of GROUP Variable



Overall Report		$6,552,506	285.29%	100.00%

Output 21: Final Row without Adjustment Shows Undesirable Percent

In the ZIP file of programs for this paper, you will find a bonus program in the Example 4 code that includes a comment on every row to show that the correct value of GRPTOT is being used to divide the value for SALES.SUM on every row, as shown in Output 22 for the Canada region.



Region	Product	Sales	Percent of Region	Comment	Percent of Total
Canada	Boot	385,613	9.06%	For this Row: Sales.Sum 385,613 ** divided by ** grptot 4,255,712	5.88%
	Men's Casual	441,903	10.38%	For this Row: Sales.Sum 441,903 ** divided by ** grptot 4,255,712	6.74%
	Men's Dress	920,101	21.62%	For this Row: Sales.Sum 920,101 ** divided by ** grptot 4,255,712	14.04%
	Sandal	14,798	0.35%	For this Row: Sales.Sum 14,798 ** divided by ** grptot 4,255,712	0.23%
	Slipper	952,751	22.39%	For this Row: Sales.Sum 952,751 ** divided by ** grptot 4,255,712	14.54%
	Sport Shoe	140,389	3.30%	For this Row: Sales.Sum 140,389 ** divided by ** grptot 4,255,712	2.14%
	Women's Casual	410,807	9.65%	For this Row: Sales.Sum 410,807 ** divided by ** grptot 4,255,712	6.27%
	Women's Dress	989,350	23.25%	For this Row: Sales.Sum 989,350 ** divided by ** grptot 4,255,712	15.10%
Canada		$4,255,712	100.00%	For this Row: Sales.Sum 4,255,712 ** divided by ** grptot 4,255,712	64.95%

Output 22: Bonus Program Shows Exactly How PCTGRP Is Calculated

EXAMPLE 5: MULTIPLE BREAK LINES

One of the limitations of the technique shown in Output 13 is that the LINE statement output spans the entire table and is not in columnar format. Sometimes you need to add extra information into break lines, but you want to see the report data aligned in columns and you want to control the extra lines.

The concept of using helper variables to insert extra break lines has already been discussed in other papers. Allison McMahill Booth had an example in her 2007 paper, which is listed in the References section.

For every extra summary line to be added to the report, you need a helper variable that is available for break processing. If you want four extra summary lines, as shown in Output 23, you need four helper variables. The following DATA step program contains four helper variables:

```
data newshoes;
  set shoes;
  brka = '1';
  brkb = '2';
  brkc = '3';
  brkd = '4';
  output;
run;
```

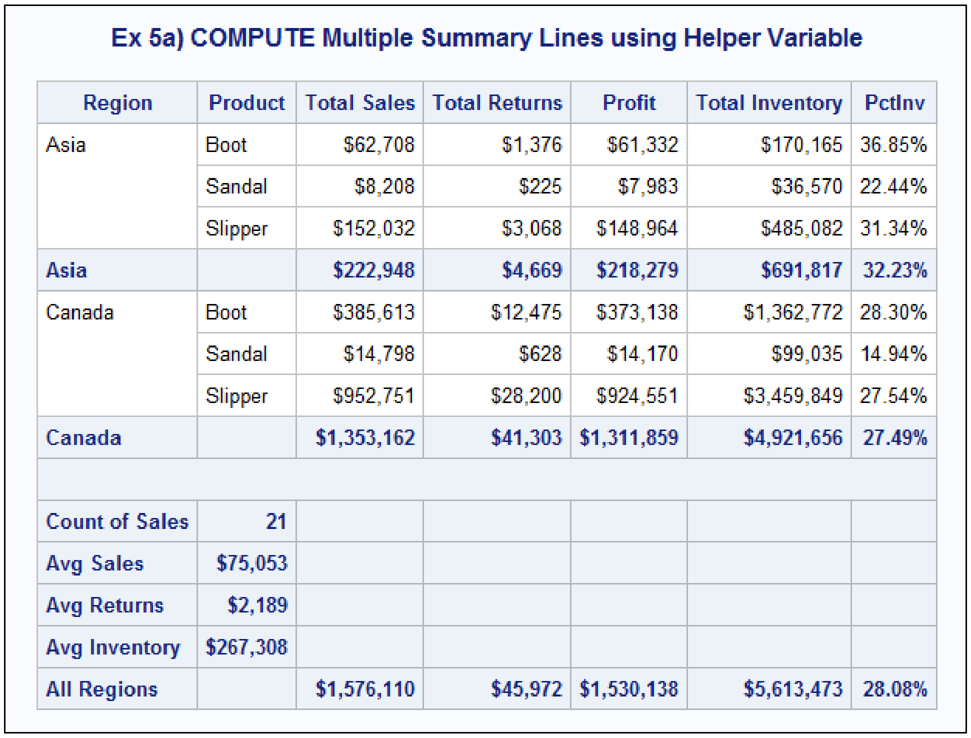
The value of these variables will be the same for all the rows in the input data, assuming that you want to have the extra summary lines only at the bottom of the report. Standard BREAK statements will be used in the PROC REPORT code, and since the values are the same for all the observations, the BREAK AFTER processing will execute only one time for each helper variable. The code is similar to the code that produced Output 13. The following code contains the relevant statements. Refer to the ZIP file for the complete program:

```
proc report data=newshoes spanrows;
  column brka brkb brkc brkd
         region product sales returns Profit inventory PctInv
         sales=salescnt sales=salesmn returns=retmn inventory=invmn;
  define brka / group noprint;
  define brkb / group noprint;
  define brkc / group noprint;
  define brkd / group noprint;
  *** DEFINE for Region, Product, Profile, PctInv same as previous **;
  define salescnt / N "Sales Count" f=comma6. noprint;
  define salesmn / mean "Avg Sales" noprint;
  define retmn / mean "Avg Returns" noprint;
  define invmn / mean "Avg Inventory" noprint;
  break after region/summarize style=Header;
  break after brka / summarize style=Header;
  break after brkb / summarize style=Header;
  break after brkc / summarize style=Header;
  break after brkd / summarize style=Header;
  rbreak after / summarize  style=Header;
  compute region;
    if upcase(_break_) = 'BRKD' then Region = 'Count of Sales';
    else if upcase(_break_) = 'BRKC' then Region = 'Avg Sales';
    else if upcase(_break_) = 'BRKB' then Region = 'Avg Returns';
    else if upcase(_break_) = 'BRKA' then Region = 'Avg Inventory ';
  endcomp;
  compute after region;
     brkline = ' ';
     if region = 'Canada' then lgbrk = 1;
     else lgbrk = 0;
```

```
       line brkline $varying. lgbrk;
    endcomp;
    compute invmn;
      if upcase(_break_) = 'BRKD' then product = put(salescnt,comma6.);
     else if upcase(_break_) = 'BRKC' then product = put(salesmn,dollar14.);
     else if upcase(_break_) = 'BRKB' then product = put(retmn,dollar14.);
     else if upcase(_break_) = 'BRKA' then product = put(invmn,dollar14.);
      if upcase(_break_) in ('BRKA', 'BRKB', 'BRKC', 'BRKD') then do;
         sales.sum=.;
         pctinv = .;
         profit = .;
         returns.sum=.;
         inventory.sum = .;
         call define('product','style','style=Header{just=r}');
     end;
    endcomp;
  run;
```

Without the extra COMPUTE blocks, the BREAK statements for BRKA, BRKB, BRKC and BRKD will produce the same numbers as shown on the RBREAK summary line. This means that the values for the alias analysis (NOPRINT) variables must be moved into the correct cells on the new summary lines.

Remember the left-to-right rule of PROC REPORT. The COMPUTE block for PRODUCT or REGION cannot access the values for the four alias variables SALESCNT, SALESMN, RETMN, and INVMN. But a COMPUTE block for INVMN can access all the values that appear to its left in the COLUMN statement.



Ex 5a) COMPUTE Multiple Summary Lines using Helper Variable

Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv
Asia	Boot	$62,708	$1,376	$61,332	$170,165	36.85%
	Sandal	$8,208	$225	$7,983	$36,570	22.44%
	Slipper	$152,032	$3,068	$148,964	$485,082	31.34%
Asia		**$222,948**	**$4,669**	**$218,279**	**$691,817**	**32.23%**
Canada	Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%
	Sandal	$14,798	$628	$14,170	$99,035	14.94%
	Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%
Canada		**$1,353,162**	**$41,303**	**$1,311,859**	**$4,921,656**	**27.49%**
Count of Sales	**21**					
Avg Sales	**$75,053**					
Avg Returns	**$2,189**					
Avg Inventory	**$267,308**					
All Regions		**$1,576,110**	**$45,972**	**$1,530,138**	**$5,613,473**	**28.08%**

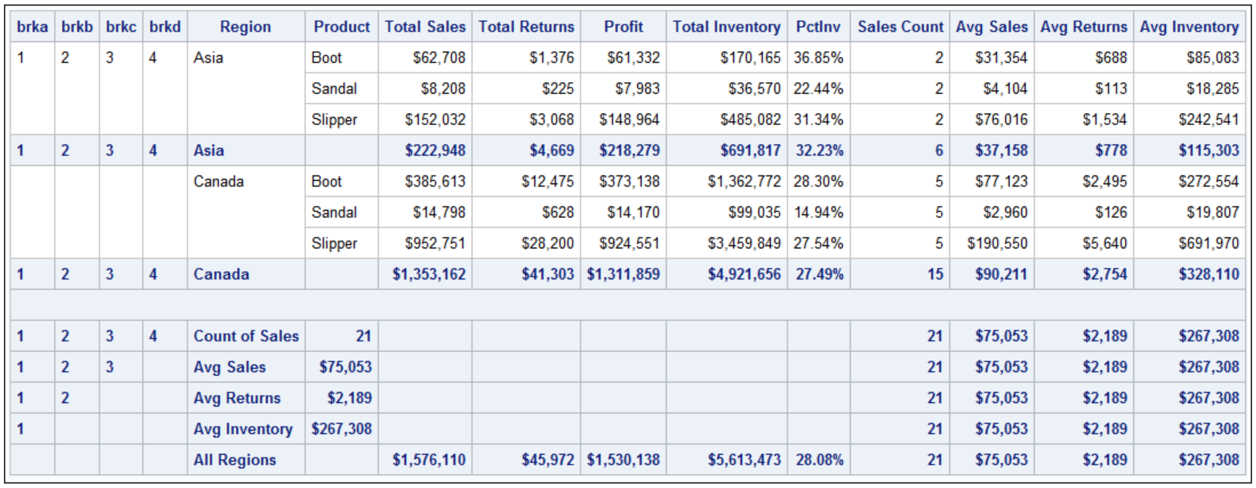
Output 23: Using Helper Variables to Add Extra Summary Lines

In the COMPUTE block for REGION, the identifying label for each summary is assigned based on the value of the automatic _BREAK_ variable. Then, in the COMPUTE block for INVMN, a PUT function is used to write the alias column values into the PRODUCT cell on every row. But the other visible cells on these four rows are assigned a value of missing so that the duplicated values do not appear. Finally, a CALL DEFINE statement changes the justification of the four PRODUCT cells to right-justified.

To place a single blank line between the summary for Canada and the extra summary lines, the technique of setting a variable, in this case, LGBRK, to 0 is used in the COMPUTE AFTER REGION block. The BRKLINE temporary variable is set to a blank value, and the temporary variable LGBRK is assigned a value of 0 for every REGION value except for Canada. This has the effect of displaying only one blank line as a separator, where desired.

Notice that BRKA is the first variable that is listed in the COLUMN statement, but it is the last BREAK line that is inserted before the RBREAK line. In a similar manner, BRKD is the final helper variable that is listed in the COLUMN statement, but it is the first of the four BREAK lines that is inserted after the last REGION.

To understand how this works, all you have to do is remove the NOPRINT option from the DEFINE statements, as illustrated by the results shown in Output 24.



brka	brkb	brkc	brkd	Region	Product	Total Sales	Total Returns	Profit	Total Inventory	PctInv	Sales Count	Avg Sales	Avg Returns	Avg Inventory
1	2	3	4	Asia	Boot	$62,708	$1,376	$61,332	$170,165	36.85%	2	$31,354	$688	$85,083
					Sandal	$8,208	$225	$7,983	$36,570	22.44%	2	$4,104	$113	$18,285
					Slipper	$152,032	$3,068	$148,964	$485,082	31.34%	2	$76,016	$1,534	$242,541
1	**2**	**3**	**4**	**Asia**		**$222,948**	**$4,669**	**$218,279**	**$691,817**	**32.23%**	**6**	**$37,158**	**$778**	**$115,303**
				Canada	Boot	$385,613	$12,475	$373,138	$1,362,772	28.30%	5	$77,123	$2,495	$272,554
					Sandal	$14,798	$628	$14,170	$99,035	14.94%	5	$2,960	$126	$19,807
					Slipper	$952,751	$28,200	$924,551	$3,459,849	27.54%	5	$190,550	$5,640	$691,970
1	**2**	**3**	**4**	**Canada**		**$1,353,162**	**$41,303**	**$1,311,859**	**$4,921,656**	**27.49%**	**15**	**$90,211**	**$2,754**	**$328,110**
1	**2**	**3**	**4**	**Count of Sales**	**21**						**21**	**$75,053**	**$2,189**	**$267,308**
1	**2**	**3**		**Avg Sales**	**$75,053**						**21**	**$75,053**	**$2,189**	**$267,308**
1	**2**			**Avg Returns**	**$2,189**						**21**	**$75,053**	**$2,189**	**$267,308**
1				**Avg Inventory**	**$267,308**						**21**	**$75,053**	**$2,189**	**$267,308**
				All Regions		**$1,576,110**	**$45,972**	**$1,530,138**	**$5,613,473**	**28.08%**	**21**	**$75,053**	**$2,189**	**$267,308**

Output 24: Results Displaying All Values Including the NOPRINT Items

There are other features of PROC REPORT, such as the CALL DEFINE and STYLE overrides, that were used in this paper, but were not emphasized in great detail. I hope this paper has motivated you to learn more about how PROC REPORT can help you generate production-quality reports.

CONCLUSION

Don't break up with BREAK processing and the COMPUTE block! Once you understand the "breakdown" of the relationship between BREAK and RBREAK processing and the ability of the COMPUTE block to write custom text and use temporary variables, the COMPUTE block will become one of the most useful techniques in your SAS and PROC REPORT toolbox. The ZIP file with all the programs used in this paper will be available for download from the SAS Global Forum 2017 website.

REFERENCES

Booth, Allison McMahill. 2010. "Evolve from a Carpenter's Apprentice to a Master Woodworker: Creating a Plan for Your Reports and Avoiding Common Pitfalls in REPORT Procedure Coding." *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute Inc.
Available http://support.sas.com/resources/papers/proceedings10/133-2010.pdf.

Booth, Allison McMahill. 2011. "Beyond the Basics: Advanced REPORT Procedure Tips and Tricks Updated for SAS® 9.2." *Proceedings of the SAS Global Forum 2011 Conference*. Cary, NC: SAS Institute Inc.
Available http://support.sas.com/resources/papers/proceedings11/246-2011.pdf.

Booth, Allison McMahill. 2012. "PROC REPORT Unwrapped: Exploring the Secrets behind One of the Most Popular Procedures in Base SAS® Software." *Proceedings of the Pharmaceutical Industry 2012*

SAS Users Group Conference. Cary, NC: SAS Institute Inc.
Available http://www.pharmasug.org/proceedings/2012/TF/PharmaSUG-2012-TF20-SAS.pdf.

Booth, Allison McMahill. 2007. "Beyond the Basics: Advanced PROC REPORT Tips and Tricks." *Proceedings of the SAS Global Forum 2007 Conference*. Cary, NC: SAS Institute Inc.
Available http://support.sas.com/rnd/papers/sgf07/sgf2007-report.pdf.

Eslinger, Jane. 2015. "The REPORT Procedure: A Primer for the Compute Block." *Proceedings of the SAS Global 2015 Conference*, Cary, NC: SAS Institute Inc.
Available https://support.sas.com/resources/papers/proceedings15/SAS1642-2015.pdf.

Zender, Cynthia L. 2008. "Creating Complex Reports." *Proceedings of the SAS Global Forum 2008 Conference*. Cary, NC: SAS Institute Inc.
Available http://www2.sas.com/proceedings/forum2008/173-2008.pdf.

Zender, Cynthia L., and Allison M. Booth. 2013. "Turn Your Plain Report into a Painted Report Using ODS Styles." *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute Inc.
Available http://support.sas.com/resources/papers/proceedings13/366-2013.pdf.

Zender, Cynthia L. 2014. "Paper SAS388-2014, Sailing Over the ACROSS Hurdle in PROC REPORT." *Proceedings of the SAS Global Forum 2014 Conference*. Cary, NC: SAS Institute Inc.
Available http://support.sas.com/resources/papers/proceedings14/SAS388-2014.pdf.

ACKNOWLEDGMENTS

The author wishes to thank Michele Ensor, Jane Eslinger, Chevell Parker, and Bari Lawhorn who generously reviewed the paper. Thanks are also due to the paper editor, Vicki Leary, who improved the readability of this paper.

CONTACT INFORMATION

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

Cynthia Zender
SAS Institute Inc.
Cynthia.Zender@sas.com