Paper 120-27

Using PROC REPORT To Produce Tables With Cumulative Totals and Row Differences

David D. Chapman, US Census Bureau, Washington, DC

ABSTRACT

PROC REPORT is a tool for tabulating and reporting the contents of SAS® data sets. PROC REPORT's strength is its ability to create table columns, modify individual cell values, and use information from different rows of the report. This flexibility allows users to create tables that show cumulative totals and row differences. Cumulative totals are used to report values such as the number of responses year-to-date, percent of a total for all periods to date, and cumulative distribution functions. Row differences show changes between row. Two PROC REPORT features that allow this are the COMPUTE block, and the distinction that PROC REPORT makes between report and data variables. This paper discusses how PROC REPORT builds a report, COMPUTE blocks, and what report and data variables are and how they work. The code to produce one and two level tables containing both cumulative row totals and row differences are shown with explanatory comments.

Key word: PROC REPORT, Cumulative Total, Cumulative Percent, Row Difference

INTRODUCTION

PROC REPORT is an excellent tool for preparing standard reports. Not all reports however fit the standard. An advantage of PROC REPORT over PROC TABULATE is its ability to define columns and manipulate the rows of a report. One important report used by statisticians is a table of cumulative totals or percent. Cumulative totals are needed when displaying both the number of responses to a survey and the cumulative responses to date by month. Often it is also important to show the change from one row to the next.

This paper discusses the following PROC REPORT features: (1) the difference between a report variable and a data variable, (2) how PROC REPORT builds a report, and (3) the use of the compute statement. PROC REPORT is flexible enough to allow the calculation a cumulative total and row change. A complete two level table showing cumulative totals and row differences is given in Table 1. The code for this final table is illustrated by starting with a simple table and progressively adding more complexity. For each table the code is given and annotated to explain what is being done and why. The code to generate the artificial data used in the example is given in the Appendix. The purpose of this paper is to show how to produce Table 1.

FINAL TABLE

Table 1 is the final table. This PROC REPORT table contains both cumulative totals and row differences. The first four columns of this tables are standard PROC REPORT columns. Columns (5) and (6) show the absolute and relative difference between two row totals. Columns (7) and (8) show the cumulative total and percent associated with row totals. Each of these columns is discussed in detail in later sections. Using Table 1 you can determine a quarter's current sales, the change in sales from the previous quarter, and the total sales for the year to date. The SAS code to produce this table is given later.

HOW PROC REPORT BUILDS A TABLE

PROC REPORT builds a table in a two step process. It first constructs a temporary data set. This temporary data set is then used along with compute blocks and data variables to create and display the table. A detailed description of the process is given in "The PROC REPORT Procedure", Chapter 32 of Version 8, SAS Procedures Guide."

```
                  TABLE 1: FINAL TABLE
          ___SALES__ QUARTER CHANGE YEAR-TO-DATE
YEAR QTR Total Percent TOTAL PERCENT YTD PERCENT
(1)  (2)   (3)     (4)    (5)     (6)     (7)    (8)
------------------------------------------------
97   1  $1,658  11%    (na)   (na)   $1,658  11%
     2  $4,068  27%  $2,410   145%   $5,726  38%
     3  $1,311   9%  $-2757   (68%)  $7,037  47%
     4  $8,038  53%  $6,727   513%  $15,075 100%
-----   ------ ---  ------   -----  ------ ----
97      $15075 100%    (na)   (na)        .    .
-----   ------ ---  ------   -----  ------ ----

98   1  $1,534  10%   (na)    (na)   $1,534  10%
     2  $4,321  28%  $2,787   182%   $5,855  38%
     3  $1,157   7%  $-3164   (73%)  $7,012  45%
     4  $8,532  55%  $7,375   637%  $15,544 100%
-----   ------ ---  ------   -----  ------ ----
98      $15544   100%     (na)   (na)      .      .
-----   ------ ---  ------   -----  ------ ----

99   1  $1,600  11%    (na)   (na)   $1,600  11%
     2  $3,815  26%  $2,215   138%   $5,415  37%
     3  $1,218   8%  $-2597   (68%)  $6,633  45%
     4  $8,079  55%  $6,861   563%  $14,712 100%
-----   ------ ---  ------   -----  ------ ----
99      $14712 100%    (na)   (na)        .    .
-----   ------ ---  ------   -----  ------ ----
```

Creating The Temporary Data Set

PROC REPORT summarizes the input data set for group, order, and across variables. As it does this, it also calculates statistics needed in both summary and detail table lines. It creates the records associated with RBREAK, BREAK, and COMPUTE BEFORE | AFTER statements for report variables. A record can be written to the temporary data set for lines not published in the report. Creation of a preliminary data set gives the PROC REPORT access to group totals before the first line of the report or a group is printed. The important thing to realize is that the temporary data set has one record for each line in the displayed table. Depending on the PROC REPORT code used, it can also have additional records that are used but not displayed. Creation of tables with a cumulative total or row difference depends on using records in the temporary data set that are not displayed.

Using The Temporary Data Set To Build The Report

Once a temporary data set is created, PROC REPORT creates the report line by line. It initializes all data step variables to missing and then sequentially constructs each row of the report. All report variables are initialized to missing at the beginning of each line of the report. The value of each report variable is determine from left-to-right; values for computed variables come from executing statements in the compute block. Only report variables to the left of a report variable being computed can be used in the calculation. All other values are obtained from the temporary file created at the start. A key feature of PROC REPORT is using data variables and compute blocks.

Data Variables Versus Report Variables

A PROC REPORT variable is either a data variable or a report variable. Data variables do not appear in the COLUMN statement; they appear only in a COMPUTE block. Data variables are set to missing initially and remain missing until specifically assigned a value. Once assigned a value, they keep that value until it is changed. Data variables hold their value between COMPUTE blocks for the same report line and between different report lines. Data variables behave like data step variables when a retain statement is used. Report variables appear in a COLUMN statement. They appear in the report unless a NOPRINT option is used in the DEFINE statement. Report variable appear in the report in the order they occur on the COLUMN statement. There are two varieties of report variables: (1) variables that come from the input data set (e.g. Sales) and (2) COMPUTED variables that are assigned values in a COMPUTE block (e.g. YTD). A report variable may or may not appear in a compute block. Report variables are initialized to missing at the beginning of each row of the report; this is similar to how data set variables are handled in the data step with the program data vector. Computed variables are created from left to right as they appear on the COLUMN statement.

Compute Blocks

A compute block is one or more programming statements that appear between the COMPUTE and ENDCOMP statements. A COMPUTE block can be associated with either a report item (data set variable, statistic, or computed variable), a location (before or after a group of observations, or at the top or bottom of a page or of the complete report). A compute block with a report variable can be used either to define the computed variable (a variable not on the input data set), to modify a report variable, or to change the display characteristics of a report variable. An example of a compute block to create a new variable is given below.

```
COMPUTE PROFIT;
        PROFIT = REVENUE.SUM  - EXPENSES.SUM
ENDCOMP;
```

BASIC TABLE

The final table starts with a simple Table 2 and adds different levels and columns to become more complex. This basic table is column two, three and four of a single level of Table 1.

The Basic Table

Table 2 is the basis for Table 1.

```
         Table 2:  1998 SALES BY QUARTER
                             SALES
                      ____________________
           QUARTER     TOTAL    PERCENT
             (1)        (2)       (3)
              1       $1,534      10%
              2       $4,321      28%
              3       $1,157       7%
              4       $8,532      55%
                     --------   ------
                     $15,544    100.0%
                     --------   ------
```

SAS Code

The code to produce this table is given below.

```
   PROC REPORT DATA=QUARTER nowd SPLIT="*"; ❶
   WHERE(year=98);
   COLUMN quarter ('_SALES_' sales
                                 sales=pct)❷;
   DEFINE quarter /GROUP FORMAT=8.
                                QUARTER*(1)';
   DEFINE sales   /ANALYSIS SUM
                   FORMAT=DOLLAR8. 'TOTAL*(2)';
   DEFINE pct     /ANALYSIS PCTSUM
                FORMAT=PERCENT8. 'PERCENT*(3)';
   RBREAK  AFTER  /SUMMARIZE UL OL;   ❸
   RUN;
```

Things To Notice In Table 2 Code

Several things to notice in this generic PROC REPORT code are:

❶ The use of the split character "*" to produce the column headers,
❷ The use of an alias to produce the column percent, and
❸ The use of the RBREAK command to produce the grand total for all groups.

SIMPLE TABLE WITH CUMULATIVE TOTALS

The basic characteristic of a cumulative total column are: (1) the cumulative total for the first row is always equal to the sum from the first row, (2) the cumulative total from the last row is always equal to the sum of all row totals, (3) the cumulative total of a row is equal to the row total plus the cumulative total of the previous row, and (4) when row totals are positive the cumulative total for any row is greater than the cumulative total of the previous row.

Simple Cumulative Total Table

Table 3 is Table 2 with a cumulative total and percent column added using data variables and a COMPUTE BEFORE block.

```
TABLE 3: 1998 SALES BY QUARTER and YEAR TO DATE
       _______________SALES________________
        ___BY QUARTER___    __YEAR TO DATE__
QUARTER  Total   Percent     TOTAL   PERCENT
  (1)     (2)      (3)        (4)      (5)
   1    $1,534     10%      $1,534     10%
   2    $4,321     28%      $5,855     38%
   3    $1,157      7%      $7,012     45%
   4    $8,532     55%     $15,544    100%
      --------  --------  --------  --------

       $15,544    100%        .         .
      --------  --------  --------  --------
```

SAS Code

Table 3 is produced by modifying and adding to the code used to produce Table 2.

```
   PROC REPORT DATA=QUARTER nowd  CENTER
                          SPLIT="*" out=Three  ❾;
   WHERE(year=98);
   COLUMN quarter ('_SALES_'
             ("_BY QUARTER_" sales sales=pct)
             ('_YEAR TO DATE_' ytd  cumpct ) ) ;
   DEFINE quarter / GROUP  FORMAT=8.
                         'QUARTER*(1)' CENTER;
   DEFINE sales  / ANALYSIS FORMAT=DOLLAR8.
                               SUM 'Total*(2)';
   DEFINE pct    / ANALYSIS PCTSUM
                FORMAT=PERCENT8.0 'Percent*(3)';
   DEFINE ytd    / COMPUTED "TOTAL*(4)"
                           FORMAT= DOLLAR8.0  ;
   DEFINE cumpct / COMPUTED 'PERCENT*(5)'
                            FORMAT=percent8.0;
   COMPUTE BEFORE;  ❶
       total+sales.SUM;  ❷
       cumtot=0;  ❸
   ENDCOMP;
   COMPUTE ytd ;  ❹
       cumtot+sales.SUM;  ❺
       ytd=cumtot;  ❻
       IF _BREAK_="_RBREAK_" THEN ytd=.; ❼
   ENDCOMP;
   COMPUTE cumpct;  ❽
       cumpct=ytd/total;
   ENDCOMP;
   RBREAK  AFTER /SUMMARIZE UL OL;
   RUN;
```

Things To Notice In Table 3 Code

The key to calculating cumulative totals and percent is to save the grand total at the beginning and to calculate the cumulative total correctly. The grand total is needed to calculate the cumulative percent. To do this requires the use of data variables and a COMPUTE BEFORE block.

❶ The initial COMPUTE BEFORE block creates a summary record on the temporary data set that contains the total for all report variables on the input data set. This is the source of the information on total sales used as the denominator in calculating the ❽cumulative percent.

❷ The variable “total” is a data variable that is set to the value of sales for all records included in the table.

❸ The variable “cumtot” is a data variable that is initialized to zero. It is a variable that will later contain the cumulative total.

❹ The COMPUTE YTD block calculates the value of the report variable that holds the cumulative total.

❺ For each record on the temporary data set, the value of sales for the current record (“sales.sum”) is added to the data variable “cumtot” that holds the sum of all previous records.

❻ The value of the report variable “ytd” is set equal to the value of “cumtot”, the cumulative total sales of all records processed. The value of the report variable “ytd” is set equal to the current value of the data variable “cumtot”. Notice that the cumulative total for the last record in the group will always be equal to the grand total for column (2).

❼ When summary records are created either by a COMPUTE BLOCK, BREAK, or RBREAK statement, a SAS variable called _BREAK_ is set. When the summary record is for the entire data set and created by a RBREAK statement ,the value of _BREAK_ is “_RBREAK_”. When the summary record is created because of either a BREAK or COMPUTE BLOCK associated with a variable, the value of _BREAK_ is the name of the variable. The value of the _BREAK_ variable is used to recognize a summary line of table. Since the sum of the cumulative total and percent of values of the individual rows has no meaning, the sum and percent are set to missing with this statement.

❽ This compute block calculates the cumulative percent. It uses the report variable “ytd” containing the cumulative total and the data variable “total” containing the grand total. Notice that the cumulative total for that last record is always equal to 100 percent.

❾ The output data set is the temporary data set used to create the final PROC REPORT table after the report variables have been created or modified. A PROC PRINT of that file is given below.

```
               Output Data Set Three
OBS  QTR  SALES     PCT     YTD    CUMPCT   _BREAK_
 1    .   15544  1.00000     .         .    _RBREAK_
 2    1    1534  0.09869   1534   0.09869
 3    2    4321  0.27799   5855   0.37667
 4    3    1157  0.07443   7012   0.45111
 5    4    8532  0.54889  15544   1.00000
 6    .   15544  1.00000     .         .    _RBREAK_
```

Notice Table 3 has five rows and data set three above has six records. Records 2,3,4, and 5 correspond to the four quarter totals and were produced by the “DEFINE quarter/ GROUP ...” statement. Record 6 is the year total and was produced by the RBREAK statement. The first record is also the total for all records on the data set and was created by the COMPUTE BEFORE statement but is not displayed by the PROC REPORT.

TWO LEVEL CUMULATIVE TOTAL TABLE

When Table 3 is expanded to include multiple years, you get Table 4. Almost the same table could have been created by using the code for Table 3 with a BY statement. This would however have given each group as a separate table rather that all groups as part of the same table.

Two Level Table With Cumulative Totals
The basic change to the SAS code to produce table 4 is the addition of the “year” report variable, the replacing of the RBREAK statement with a BREAK statement, and adding a COMPUTE BEFORE YEAR block.

```
   TABLE 4: SALES BY QUARTER and YEAR TO DATE
          ______________SALES_______________
YEAR QUARTER __BY QUARTER__  __YEAR TO DATE__
              TOTAL  PERCENT  TOTAL   PERCENT
 (1)    (2)    (3)     (4)     (5)      (6)
97       1   $1,658    11%   $1,658     11%
         2   $4,068    27%   $5,726     38%
         3   $1,311     9%   $7,037     47%
         4   $8,038    53%  $15,075    100%
-----       -------  ------  -------   ------
97          $15,075   100%      .        .
-----       -------  ------  -------   ------

98       1   $1,534    10%   $1,534     10%
         2   $4,321    28%   $5,855     38%
         3   $1,157     7%   $7,012     45%
         4   $8,532    55%  $15,544    100%
-----       -------  ------  -------   ------
98          $15,544   100%      .        .
-----       -------  ------  -------   ------

99       1   $1,600    11%   $1,600     11%
         2   $3,815    26%   $5,415     37%
         3   $1,218     8%   $6,633     45%
         4   $8,079    55%  $14,712    100%
-----       -------  ------  -------   ------
99           $14,712  100%      .        .
-----       -------  ------  -------   ------
```

SAS Code
The SAS code to produce table 4 is given below.

```
PROC REPORT DATA=QUARTER nowd  CENTER
SPLIT="*" out=Four  ❾;
COLUMN year quarter
('_SALES_'
('_BY QUARTER_' sales pct)
("_YEAR TO DATE_" ytd cumpct));
DEFINE year    / GROUP FORMAT=5. 'YEAR*' '*
                                   (1) *' CENTER;
DEFINE quarter / GROUP FORMAT=8. 'QUARTER ' *
' ' * (2)*' CENTER;
DEFINE sales   / ANALYSIS FORMAT=DOLLAR8.
                          SUM 'TOTAL*(3)';
DEFINE pct     / COMPUTED
       FORMAT=PERCENT8.0 'PERCENT*(4)';
DEFINE ytd     / COMPUTED "TOTAL*(5)" FORMAT=
DOLLAR8.0 ;
DEFINE cumpct  / COMPUTED FORMAT=percent8.0
'PERCENT*(6)';
COMPUTE BEFORE year;  ❶
    total+sales.SUM;  ❷
    cumtot=0;  ❸
ENDCOMP;
COMPUTE ytd ;
    cumtot+sales.SUM;  ❹
    ytd=cumtot;  ❺
    IF _BREAK_="year" THEN ytd=.;  ❻
    if _break_="year" THEN total=0;
ENDCOMP;
COMPUTE cumpct;
    cumpct=ytd/total;  ❼
ENDCOMP;
COMPUTE pct;
    pct=sales.SUM/total ;  ❽
ENDCOMP;
```

```
   BREAK AFTER year / SUMMARIZE SKIP  UL OL ;
   RUN;
```

Things To Notice in Table 4 Code
Calculating cumulative totals and percent separately for different groups in a PROC REPORT table is almost the same as doing it for one group. The key is to initialize the cumulative total data variables at the start of each group and not to compute a grand total. Key features in the code are:

❶ The COMPUTE BEFORE block makes the cumulative total for year available before each quarter record is created.

❷ Creates the data variable "total" and sets it equal to the value of "sales.sum".

❸ Creates the data variable "cumtot" and sets it equal to zero. The value of "cumtot" is set to zero every time there is a new value for "year".

❹ For each record the value of "sales" is added to "cumtot" to make "cumtot" the cumulative total for the year.

❺ The report variable – "ytd" – is set equal to the data variable "cumtot".

❻ When the report line is for the yearly total, the value of _BREAK_ will be "year" and the sum of the cumulative totals and percent for the year are set to missing because the answer is meaningless.

❼ The cumulative percent is calculated for each quarter using the cumulative total for the year ("cumtot") and the yearly total ("total").

❽ The quarterly percent is calculated by dividing the sum of "sales" for the quarter ("sales.sum") by the total sales for the year ("total").

❾ This creates the output data set for Table 4. Note that Table 4 has 15 rows of data and the output data set has 18 rows of data. The three extra rows of data in the output data set were generated by the COMPUTE BEFORE YEAR block ❶ and they are used by PROC REPORT to calculate the percent and cumulative percent column

```
                  Output Data Set Four
Obs year QTR sales      pct       ytd     cumpct   _BREAK_
 1   97   .  15075  .             .          .       year
 2   97   1   1658  0.10998     1658    0.10998
 3   97   2   4068  0.26985     5726    0.37983
 4   97   3   1311  0.08697     7037    0.46680
 5   97   4   8038  0.53320    15075    1.00000
 6   97   .  15075  1.00000       .          .       year
 7   98   .  15544   .            .          .       year
 8   98   1   1534  0.09869     1534    0.09869
 9   98   2   4321  0.27799     5855    0.37667
10   98   3   1157  0.07443     7012    0.45111
11   98   4   8532  0.54889    15544    1.00000
12   98   .  15544  1.00000       .          .       year
13   99   .  14712   .            .          .       year
14   99   1   1600  0.10875     1600   0.10875
15   99   2   3815  0.25931     5415   0.36807
16   99   3   1218  0.08279     6633   0.45086
17   99   4    8079 0.54914    14712   1.00000
18   99   .   14712 1.00000       .          .       year
```

SIMPLE TABLE WITH ROW DIFFERENCES
A difference table is a summary table where values in at least one column equal a current row total minus the previous row total. The difference shows the change in the current row from the previous row. When rows are time periods, differences show either the absolute or relative change in a population from one time period to the next. In Table 5 the difference represents the change in sales for a particular quarter. On the first row, the value is set to missing. Subsequent rows are the difference between that row and the previous row.

Simple Difference Table
Table 5 is a simple difference table showing the change in sales from one quarter to the next. Column (4) shows the absolute change in the current row from the previous row. Column (5) shows the percent or relative change.

```
         Table 5: SIMPLE DIFFERENCE TABLE
        ____SALES ____        _QUARTER CHANGE_
QUARTER TOTAL    PERCENT      CHANGE   PERCENT
  (1)     (2)      (3)         (4)       (5)
------------------------------------------------
   1    $1,534     10%           .         .
   2    $4,321     28%       $2,787     1817%
   3    $1,157      7%      $-3,164     ( 73%)
   4    $8,532     55%       $7,375      637%
      --------  --------  --------  --------
       $15,544    100%           .         .
      --------  --------  --------  --------
```

SAS Code
The SAS code that produces table 5 is given below.

```
PROC REPORT DATA=QUARTER NOWD OUT=Five❾
                    CENTER HEADSKIP HEADLINE;
WHERE(YEAR=98);
COLUMN
  ("_QUARTER_"  quarter )
  ("_SALES_" sales sales=pct )
  ("_QUARTER CHANGE_" diff diff_pct) ;
DEFINE quarter  / GROUP  FORMAT=12. '(1)'
                                      CENTER;
DEFINE sales    / ANALYSIS FORMAT=DOLLAR8.
                     SUM 'TOTAL*(2)';
DEFINE pct      / ANALYSIS PCTSUM
            FORMAT=PERCENT8.1  'PERCENT*(3)';
DEFINE diff     / COMPUTED FORMAT= DOLLAR8.0
                             'CHANGE*(4)' ;
DEFINE diff_pct / COMPUTED FORMAT=percent8.0
                           'PERCENT*(5)' ;
COMPUTE diff;
  r+1;  ❶
  IF r=1 THEN diff=. ;  ❷
  IF r>1 THEN DO;
    last=sales.SUM;
    if _BREAK_ EQ "     " THEN DO;  ❸
        diff=sales.SUM-py;  ❹
        last=sales.SUM ;  ❻
        END;
    if _BREAK_ NE "      " THEN
      diff = .  ;  ❺
  END;
  py=sales.SUM;  ❼
ENDCOMP;
COMPUTE diff_pct;
  diff_pct = (diff / (last - diff) );  ❽
ENDCOMP;
RBREAK AFTER  / SUMMARIZE SKIP OL UL ;RUN;
```

Things To Notice In Table 5 Code
❶ The data variable, "r" , is a line counter used to detect the first summary line so it will have a missing value for the difference.

❷ Sets the row difference to missing when it is first summary line.

❸ Detects whether it is a summary line or a grand total of summary lines created by a RBREAK statement.

❹ Calculates the difference between the current row and the previous row. The previous row total is save in ❼.

❺ Sets the difference to missing when it is not a summary line

❻ Puts the value of current summary line into a data variable called “last” needed to calculate the row difference.

❼ Saves the current row total in the data variable “py”.

❽ Calculates the percent change between the current and prior row. The variable “diff” contains the change; “last” now contains the current value of the row. .

❾ Produces the output data set given below.

```
                  Output Data Set Five
OBS QTR SALES  PCT      DIFF    PCTDIF _BREAK_
1     1   1534 0.09869      .         .
2     2   4321 0.27799   2787   1.81682
3     3   1157 0.07443  -3164  -0.73224
4     4   8532 0.54889   7375   6.37424
5     .  15544 1.00000     .        .        _RBREAK_
```

TWO LEVEL DIFFERENCE TABLE

Like the two level table with cumulative totals, the two level difference table is an extension of the one level table plus some straightforward changes. In Table 6, the simple difference table could have been used with a BY statement to create the table.

Two Level Difference Table

The two level difference table is given in Table 6. It is just like Table 5 except there is a separate set of rows for each of the three years.

```
         TABLE 6: Two Level Difference Table


                ____SALES_____   _QUARTERLY CHANGE_
        QUARTER  TOTAL   PERCENT  TOTAL      PERCENT
YEAR     (1)      (2)     (3)      (4)         (5)
-----------------------------------------------------------
97         1    $1,693    11%        .           .
           2    $4,140    27%     $2,448       145%
           3    $1,413     9%    $-2,728     ( 66%)
           4    $8,167    53%     $6,754       478%
---------      -------  -------  --------  ---------
97             $15,413   100%        .         .
---------      -------  -------  --------  ---------

98         1    $1,560    10%        .         .
           2    $4,405    28%     $2,845      182%
           3    $1,264     8%    $-3,141    ( 71%)
           4    $8,669    55%     $7,405      586%
---------      -------  -------  --------  ---------
98             $15,897   100%        .           .

---------      -------  -------  --------  ---------
99         1    $1,631    11%        .         .
           2    $3,879    26%     $2,248      138%
           3    $1,317     9%    $-2,561    ( 66%)
           4    $8,201    55%     $6,883      522%
---------      -------  -------  --------  ---------
99             $15,028   100%       .          .
---------      -------  -------  --------  ---------
```

SAS Code

The SAS code used to produce Table 6 is similar to the code use to produce Table 5.

```
PROC REPORT DATA=QUARTER NOWD OUT=Six
           SPLIT="*" CENTER HEADSKIP HEADLINE;
COLUMN
  ( year quarter )
  ("_SALES_" sales sales=pct)
  ("CHANGE FROM PREVIOUS QUARTER_" diff
                                  diff_pct) ;
DEFINE YEAR / GROUP;
DEFINE quarter / GROUP  FORMAT=8.
                          'QUARTER*(1)' CENTER;
DEFINE sales   / COMPUTED FORMAT=DOLLAR8.
                                SUM 'Total*(2)';
DEFINE pct     / ANALYSIS PCTSUM
              FORMAT=PERCENT8.0 'Percent*(3)';
DEFINE diff    / COMPUTED "TOTAL*(4)"
                             FORMAT= DOLLAR8.0 ;
DEFINE diff_pct / COMPUTED 'PERCENT*(5)'
                              FORMAT=percent9.0;
COMPUTE BEFORE year;
  r=0;
  last=0;
  total=sales.SUM;
ENDCOMP;
COMPUTE pct;
  pct=sales.SUM/total;
ENDCOMP;
COMPUTE diff;
  r+1;
  IF r=1 THEN diff=. ;
  else DO;
    if _BREAK_ EQ "     " THEN
       diff=sales.SUM-last ;
    else  diff = .  ;
  end;
  last = sales.SUM;
ENDCOMP;
COMPUTE diff_pct;
  diff_pct= (diff/(last-diff) );
ENDCOMP;
BREAK AFTER year / SUMMARIZE SKIP OL UL ;
 RUN;
```

Things To Notice in Table 6 Code

The “COMPUTE BEFORE” block is used to created a record in the temporary data set needed to assign a value fo data variable “total”. The variable “total” is the denominator in calculating the variable “pct.” The “COMPUTE BEFORE” block also is used to initialize the line counter , “r”, to zero and the value of the last row, “last”, to zero.

FINAL TABLE

The final table is given in Table 1. The code below produces that table and is just a combination of the code used to make Table 4 and Table 6. It uses all the techniques discussed earlier.

```
PROC REPORT DATA=QUARTER NOWD SPLIT="*"
headline;
COLUMN
  ( year quarter )
  ("_SALES_" sales pct )
  ("_QUARTERLY CHANGE_" diff diff_pct )
  ("_YEAR-TO-DATE_"  ytd cumpct );
DEFINE year        /GROUP    'YEAR*(1)';
DEFINE quarter   /GROUP    FORMAT=8.
              'QUARTER*(2)' CENTER;
DEFINE sales      /COMPUTED FORMAT=DOLLAR8.
                         SUM 'Total*(3)';
DEFINE PCT       /COMPUTED FORMAT=PERCENT8.0
                            'Percent*(4)';
DEFINE diff         /COMPUTED FORMAT=
       DOLLARA.0 "TOTAL*(5)" center;
DEFINE diff_pct/COMPUTED
FORMAT=pctA.'PERCENT*(6)';
DEFINE ytd       /COMPUTED FORMAT=DOLLAR8.
                             'YTD*(7)'  ;
DEFINE cumpct   /COMPUTED FORMAT=PERCENT8.0
                            'PERCENT*(8)';
COMPUTE BEFORE year;
       r=0;
       last=0;
       total=sales.SUM;
       cumtot=0;
ENDCOMP;
COMPUTE diff;
      r+1;
      IF r=1 THEN diff=. ;
      IF r>1 THEN DO;
          last=sales.SUM;
```

```
              if _BREAK_ EQ "     " THEN
                   diff = sales.SUM-py;
              if _BREAK_ EQ "    " THEN
                  last = sales.SUM ;
              if _BREAK_ NE "      " THEN
                  diff = .  ;
          END;
          py = sales.SUM;
     ENDCOMP;
     COMPUTE diff_pct;
          diff_pct = (diff / (last - diff) );
     ENDCOMP;
     BREAK AFTER year / SUMMARIZE SKIP OL UL ;
     COMPUTE ytd ;
         cumtot+sales.SUM;
         ytd=cumtot;
         IF _BREAK_="year" THEN ytd  =.;
         if _BREAK_="year" THEN total=0;
     ENDCOMP;
     COMPUTE cumpct;
         cumpct=ytd/total;
     ENDCOMP;
     COMPUTE pct;
         pct=sales.SUM/total ;
     ENDCOMP;
     BREAK AFTER year / SUMMARIZE OL UL ;
     RUN;
```

CONCLUSION

PROC REPORT can be used to construct complex and complicated tables by making use of data variables and the COMPUTE block and by knowing how PROC REPORT uses the temporary data set to construct the report. Use of the additional capabilities added by the Output Display System (ODS) will allow even more complex tables to be produced in the future.

TRADEMARKS

SAS® is a registered trademark of SAS Institute, Inc.

REFERENCES

Pass, Ray (2000) PROC REPORT – Land of the Missing OBS Column, NESUG 2000 Proceedings, Philadelphia, Pennsylvania, September 24 - 26, 2000.

The Report Procedure, Chapter 32, SAS PROCEDURES GUIDE, Version 8 (1999) SAS Institute, Cary, NC

DISCLAIMER

This paper reports the results of research and analysis undertaken by Census Bureau staff. It has undergone a more limited review by the Census Bureau than its official publications. This report is released to inform interested parties and to encourage discussion.

CONTACT INFORMATION

I am interested in your comments, questions, and any problems you have applying this approach.

David D. Chapman
Chief, Frame Development Staff
Economic Statistical Methods and Programming Division
U.S. Census Bureau
Washington, DC, 20233-6200
Work Phone: (301) 457-4904
Fax:(301)-457-1382
Email:david.d.chapman@census.gov

APPENDIX

This code generates the SAS data set used to produce the examples in this paper. With this data set and included code you can reproduce all tables. The PROC FORMAT code below creates the formats used to add “(na)” symbols into table 1.

```
   DATA quarter;
   DO year=97 TO 99;
     DO j=1 TO 12;
       IF j=1 THEN xx='1dec1997'd;
       QUARTER=QTR( intnx('month',xx,J) );
       DO n=1 to 100;
       sales=int(normal(123)*(20)+quarter*7);
          IF QUARTER=3 THEN SALES=SALES-15;
          OUTPUT;
       END;
     END;
   END;
   RUN;

   PROC FORMAT ;
   VALUE PCTA
           .='(na)'
           OTHER=[PERCENT8.0];
   VALUE DOLLARA
           .='(na)'
           OTHER=[DOLLAR8.0];
   RUN;
```