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

---

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 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 determined 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 to either 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

<table>
<thead>
<tr>
<th>QUARTER</th>
<th>TOTAL</th>
<th>PERCENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1,534</td>
<td>10%</td>
</tr>
<tr>
<td>2</td>
<td>$4,321</td>
<td>28%</td>
</tr>
<tr>
<td>3</td>
<td>$1,157</td>
<td>7%</td>
</tr>
<tr>
<td>4</td>
<td>$8,532</td>
<td>55%</td>
</tr>
</tbody>
</table>

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

PROC REPORT DATA=QUARTER NOWD CENTER SPLIT="*" out=Three;
WHERE (year=98);
COLUMN quarter ('QUARTER' sales sales=pct);
DEFINE quarter /GROUP FORMAT=8. 'QUARTER*(1)' CENTER;
DEFINE sales / ANALYSIS FORMAT=DOLLAR8. 'QUARTER*(1)'
('YEAR TO DATE' ytd cumpct);
DEFINE ytd / COMPUTED 'TOTAL*(4)' FORMAT=DOLLAR8.0;
DEFINE pct / ANALYSIS PCTSUM FORMAT=PERCENT8.0 'PERCENT*(3)';
DEFINE cumpct / COMPUTED 'TOTAL*(4)' FORMAT=PERCENT8.0;
COMPUTE BEFORE; total+sales.sum;
cumtot=0;
ENDCOMP;
COMPUTE ytd;
cumtot+sales.sum;
ytd=cumtot;
IF _BREAK_="RBREAK" THEN ytd=.;
ENDCOMP;
COMPUTE cumpct;
cumctc=ystd/total;
ENDCOMP;
RBREAK AFTER /SUMMARIZE UL OL;
RUN;

Things To Notice In Table 2 Code
Several things to notice in this generic PROC REPORT code are:

1. The use of the split character "*" to produce the column headers.
2. The use of an alias to produce the column percent, and
3. 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, and (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

<table>
<thead>
<tr>
<th>QUARTER</th>
<th>Total</th>
<th>Percent</th>
<th>YEAR TO DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1,534</td>
<td>10%</td>
<td>$1,534</td>
</tr>
<tr>
<td>2</td>
<td>$4,321</td>
<td>28%</td>
<td>$5,855</td>
</tr>
<tr>
<td>3</td>
<td>$1,157</td>
<td>7%</td>
<td>$7,012</td>
</tr>
<tr>
<td>4</td>
<td>$8,532</td>
<td>55%</td>
<td>$15,544</td>
</tr>
</tbody>
</table>

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;
WHERE (year=98);
COLUMN quarter ('QUARTER' sales sales=pct);
DEFINE quarter /GROUP FORMAT=8. 'QUARTER*(1)' CENTER;
DEFINE sales / ANALYSIS FORMAT=DOLLAR8. 'QUARTER*(1)'
('YEAR TO DATE' ytd cumpct);
DEFINE ytd / COMPUTED 'TOTAL*(4)' FORMAT=DOLLAR8.0;
DEFINE pct / ANALYSIS PCTSUM FORMAT=PERCENT8.0 'PERCENT*(3)';
DEFINE cumpct / COMPUTED 'TOTAL*(4)' FORMAT=PERCENT8.0;
COMPUTE BEFORE; total+sales.sum;
cumtot=0;
ENDCOMP;
COMPUTE ytd;
cumtot+sales.sum;
ytd=cumtot;
IF _BREAK_="RBREAK" THEN ytd=.;
ENDCOMP;
COMPUTE cumpct;
cumctc=ystd/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.

1. 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 cumulative percent.

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

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

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

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.

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).

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 missing this statement.

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.

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.

<table>
<thead>
<tr>
<th>Obs</th>
<th>QTR*</th>
<th>SALES</th>
<th>PCT</th>
<th>YTD</th>
<th>CUMCPT</th>
<th>BREAK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1534</td>
<td>0.09869</td>
<td>1534</td>
<td>0.09869</td>
<td><em>RBREAK</em></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4321</td>
<td>0.27799</td>
<td>5855</td>
<td>0.37667</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1157</td>
<td>0.07443</td>
<td>7012</td>
<td>0.45111</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>8532</td>
<td>0.54889</td>
<td>15544</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>15544</td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

### SUGI 27 Data Presentation

**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>
<thead>
<tr>
<th>YEAR</th>
<th>QUARTER</th>
<th>BY QUARTER</th>
<th>YEAR TO DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TOTAL</td>
<td>PERCENT</td>
<td>TOTAL</td>
</tr>
<tr>
<td>97</td>
<td>1</td>
<td>$1,658</td>
<td>11%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>$4,068</td>
<td>27%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>$1,311</td>
<td>9%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>$8,038</td>
<td>53%</td>
</tr>
<tr>
<td>98</td>
<td>1</td>
<td>$1,534</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>$4,321</td>
<td>28%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>$1,157</td>
<td>7%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>$8,532</td>
<td>55%</td>
</tr>
</tbody>
</table>

| 99   | 1       | $1,554     | 100%        | .          | .          |
|      | 2       | $3,815     | 26%         | $5,415     | 37%        |
|      | 3       | $1,218     | 8%          | $6,633     | 45%        |
|      | 4       | $8,079     | 55%         | $14,712    | 100%       |

| 97   | 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%       |

| 98   | 1       | $14,712    | 100%        | .          | .          |
|      | 2       | $3,815     | 26%         | $5,415     | 37%        |
|      | 3       | $1,218     | 8%          | $6,633     | 45%        |
|      | 4       | $8,079     | 55%         | $14,712    | 100%       |

| 99   | 1       | $14,712    | 100%        | .          | .          |

**SAS Code**

The SAS code to produce table 4 is given below.

```sas
PROC REPORT DATA=QUARTER nowd CENTER SPLIT="**" out=Four ;
COLUMN year quarter ('SALES ' ('BY QUARTER ' sales pct) ('YEAR TO DATE " ytd cumpct));
DEFINE year / GROUP FORMAT=5. 'YEAR*' * CENTER; ` ' ' * (2)*' 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=percent8.0 'PERCENT*'(6)';
COMPUTE BEFORE year; 1 total=sales.SUM; 2 cumtot=0; 3 ENDCOMP;
COMPUTE ytd ; 1 cumtot=sales.SUM; 2 ytd=cumtot; 3 IF _BREAK_="year" THEN ytd=.; 4 IF _BREAK_="year" THEN total=0;
ENDCOMP;
COMPUTE cumpct; 1 cumpct=ytd/total; 2 ENDCOMP;
COMPUTE pct; 1 pct=sales.SUM/total ; 2 ENDCOMP;
```
SUGI 27

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:

1. The COMPUTE BEFORE block makes the cumulative total for year available before each quarter record is created.
2. Creates the data variable "total" and sets it equal to the value of "sales.sum".
3. 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".
4. For each record the value of "sales" is added to "cumtot" to make "cumtot" the cumulative total for the year.
5. The report variable – "ytd" – is set equal to the data variable "cumtot".
6. 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.
7. The cumulative percent is calculated for each quarter using the cumulative total for the year ("cumtot") and the yearly total ("total").
8. The quarterly percent is calculated by dividing the sum of "sales" for the quarter ("sales.sum") by the total sales for the year ("total").
9. 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 and they are used by PROC REPORT to calculate the percent and cumulative percent column.

Output Data Set Four
<table>
<thead>
<tr>
<th>Obs</th>
<th>Quarters</th>
<th>Sales</th>
<th>Pct</th>
<th>YTD</th>
<th>Cumulative</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>97</td>
<td>15075</td>
<td>0.10998</td>
<td>1558</td>
<td>0.10998</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>97</td>
<td>1658</td>
<td>0.10998</td>
<td>1558</td>
<td>0.10998</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>97</td>
<td>2068</td>
<td>0.12985</td>
<td>9726</td>
<td>0.12985</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>97</td>
<td>1311</td>
<td>0.08697</td>
<td>7037</td>
<td>0.08697</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>97</td>
<td>8038</td>
<td>0.53320</td>
<td>15075</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>97</td>
<td>15075</td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>98</td>
<td>15544</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>98</td>
<td>15534</td>
<td>0.09869</td>
<td>1534</td>
<td>0.09869</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>98</td>
<td>4321</td>
<td>0.22799</td>
<td>5855</td>
<td>0.36766</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>98</td>
<td>1157</td>
<td>0.07443</td>
<td>7012</td>
<td>0.45111</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>98</td>
<td>4321</td>
<td>0.22799</td>
<td>5855</td>
<td>0.36766</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>98</td>
<td>1534</td>
<td>0.10998</td>
<td>1534</td>
<td>0.10998</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>98</td>
<td>14712</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>98</td>
<td>1600</td>
<td>0.10875</td>
<td>1600</td>
<td>0.10875</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>98</td>
<td>3815</td>
<td>0.25931</td>
<td>5415</td>
<td>0.36807</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>98</td>
<td>1218</td>
<td>0.08279</td>
<td>6633</td>
<td>0.45086</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>98</td>
<td>8079</td>
<td>0.54914</td>
<td>14712</td>
<td>1.00000</td>
<td></td>
</tr>
</tbody>
</table>

Things To Notice In Table 5 Code

1. The data variable, "Y", is a line counter used to detect the first summary line so it will have a missing value for the difference.
2. Sets the row difference to missing when it is first summary line.
3. Detects whether it is a summary line or a grand total of summary lines created by a RBREAK statement.

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 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.

Table 5: SIMPLE DIFFERENCE TABLE

<table>
<thead>
<tr>
<th>Quarters</th>
<th>Sales</th>
<th>Percent</th>
<th>Change</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(2)</td>
<td>(3)</td>
<td>(4)</td>
<td>(5)</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>---------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>1</td>
<td>$1,534</td>
<td>10%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$4,321</td>
<td>28%</td>
<td>$2,787</td>
<td>1817%</td>
</tr>
<tr>
<td>3</td>
<td>$1,157</td>
<td>7%</td>
<td>$-3,164</td>
<td>(73%)</td>
</tr>
<tr>
<td>4</td>
<td>$8,532</td>
<td>55%</td>
<td>$7,375</td>
<td>637%</td>
</tr>
<tr>
<td></td>
<td>$15,544</td>
<td>100%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

PROC REPORT DATA=quarter NOWD OUT=five 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;
  IF r=1 THEN diff= ;
  IF r>1 THEN DO;
  last=sales.SUM;
  IF BREAK EQ "" THEN DO;
  diff=sales.SUM-py;
  last=sales.SUM ;
  END;
  IF BREAK NE "" THEN
  diff = . ;
  END;
  py=sales.SUM;
ENDCOMP;
COMPUTE diff_pct;
  diff_pct = (diff / (last - diff) ) ;
ENDCOMP;
RBREAK AFTER / SUMMARIZE SKIP UL OL ;RUN;

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

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Total</th>
<th>Percent</th>
<th>Change</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(2)</td>
<td>(3)</td>
<td>(4)</td>
<td>(5)</td>
</tr>
<tr>
<td>1</td>
<td>$1,534</td>
<td>10%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$4,321</td>
<td>28%</td>
<td>$2,787</td>
<td>1817%</td>
</tr>
<tr>
<td>3</td>
<td>$1,157</td>
<td>7%</td>
<td>$-3,164</td>
<td>(73%)</td>
</tr>
<tr>
<td>4</td>
<td>$8,532</td>
<td>55%</td>
<td>$7,375</td>
<td>637%</td>
</tr>
<tr>
<td></td>
<td>$15,544</td>
<td>100%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

PROC REPORT DATA=QUARTER NOWD OUT=five 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;
  IF r=1 THEN diff= ;
  IF r>1 THEN DO;
  last=sales.SUM;
  if BREAK EQ "" THEN DO;
  diff=sales.SUM-py;
  last=sales.SUM ;
  END;
  if BREAK NE "" THEN
  diff = . ;
  END;
  py=sales.SUM;
ENDCOMP;
COMPUTE diff_pct;
  diff_pct = (diff / (last - diff) ) ;
ENDCOMP;
RBREAK AFTER / SUMMARIZE SKIP UL OL ;RUN;
Calculates the difference between the current row and the
previous row. The previous row total is save in _RBREAK_.

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

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

Saves the current row total in the data variable "py".

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.

Produces the output data set given below.

---

**TABLE 6: 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.

---

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="*" COLUMN;
COLUMN
{ year quarter }
{* _SALES_ sales sales=pct }
{* _YEAR-TO-DATE_ ytd cumptc }
{ _QUARTERLY CHANGE_ diff diff_pct }
{ "PERCENT*(8)" 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;
  diff=sales.SUM-last;
  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;
```

---

SAS Code

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

---
### 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.

```sas
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) ); 
      sales=int(normal(123)*(20)+quarter*7); 
      IF QUARTER=3 THEN SALES=SALES-15; 
      OUTPUT; 
    END; 
  END; 
END; 
RUN;
```

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