Chapter 9
The COMPUTAB Procedure

Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: COMPUTAB Procedure</td>
<td>464</td>
</tr>
<tr>
<td>Getting Started: COMPUTAB Procedure</td>
<td>464</td>
</tr>
<tr>
<td>Producing a Simple Report</td>
<td>464</td>
</tr>
<tr>
<td>Using PROC COMPUTAB</td>
<td>466</td>
</tr>
<tr>
<td>Defining Report Layout</td>
<td>466</td>
</tr>
<tr>
<td>Adding Computed Rows and Columns</td>
<td>467</td>
</tr>
<tr>
<td>Enhancing the Report</td>
<td>468</td>
</tr>
<tr>
<td>Syntax: COMPUTAB Procedure</td>
<td>469</td>
</tr>
<tr>
<td>Functional Summary</td>
<td>470</td>
</tr>
<tr>
<td>PROC COMPUTAB Statement</td>
<td>471</td>
</tr>
<tr>
<td>BY Statement</td>
<td>473</td>
</tr>
<tr>
<td>CELL Statement</td>
<td>473</td>
</tr>
<tr>
<td>COLUMNS Statement</td>
<td>473</td>
</tr>
<tr>
<td>INIT Statement</td>
<td>475</td>
</tr>
<tr>
<td>ROWS Statement</td>
<td>476</td>
</tr>
<tr>
<td>SUMBY Statement</td>
<td>478</td>
</tr>
<tr>
<td>Programming Statements</td>
<td>479</td>
</tr>
<tr>
<td>Details: COMPUTAB Procedure</td>
<td>480</td>
</tr>
<tr>
<td>Program Flow Example</td>
<td>480</td>
</tr>
<tr>
<td>Order of Calculations</td>
<td>482</td>
</tr>
<tr>
<td>Column Selection</td>
<td>484</td>
</tr>
<tr>
<td>Controlling Execution within Row and Column Blocks</td>
<td>485</td>
</tr>
<tr>
<td>Program Flow</td>
<td>486</td>
</tr>
<tr>
<td>Direct Access to Table Cells</td>
<td>487</td>
</tr>
<tr>
<td>Reserved Words</td>
<td>488</td>
</tr>
<tr>
<td>Missing Values</td>
<td>488</td>
</tr>
<tr>
<td>OUT= Data Set</td>
<td>488</td>
</tr>
<tr>
<td>NOTRANS Option</td>
<td>489</td>
</tr>
<tr>
<td>Examples: COMPUTAB Procedure</td>
<td>489</td>
</tr>
<tr>
<td>Example 9.1: Using Programming Statements</td>
<td>489</td>
</tr>
<tr>
<td>Example 9.2: Enhancing a Report</td>
<td>492</td>
</tr>
<tr>
<td>Example 9.3: Comparison of Actual and Budget</td>
<td>497</td>
</tr>
<tr>
<td>Example 9.4: Consolidations</td>
<td>500</td>
</tr>
<tr>
<td>Example 9.5: Creating an Output Data Set</td>
<td>505</td>
</tr>
<tr>
<td>Example 9.6: A What-If Market Analysis</td>
<td>507</td>
</tr>
<tr>
<td>Example 9.7: Cash Flows</td>
<td>512</td>
</tr>
</tbody>
</table>
Overview: COMPUTAB Procedure

The COMPUTAB (computing and tabular reporting) procedure produces tabular reports generated using a programmable data table.

The COMPUTAB procedure is especially useful when you need both the power of a programmable spreadsheet and a report generation system, but you want to set up a program to run in a batch mode and generate routine reports.

With PROC COMPUTAB, you can select a subset of observations from the input data set, define the format of a table, operate on its row and column values, and create new columns and rows. Access to individual table values is available when needed.

The COMPUTAB procedure can tailor reports to almost any desired specification and provide consolidation reports over summarization variables. The generated report values can be stored in an output data set. PROC COMPUTAB is especially useful in creating tabular reports such as income statements, balance sheets, and other row and column reports.

Getting Started: COMPUTAB Procedure

The following example shows the different types of reports that can be generated by PROC COMPUTAB.

Suppose a company has monthly expense data on three of its divisions and wants to produce the year-to-date expense report shown in Figure 9.1. This section starts out with the default report produced by the COMPUTAB procedure and modifies it until the desired report is achieved.

![Figure 9.1 Year-to-Date Expense Report](image)

### Year to Date Expenses

<table>
<thead>
<tr>
<th>Division</th>
<th>Division</th>
<th>Division</th>
<th>Division</th>
<th>All Divisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Travel Expenses within U.S.</td>
<td>18700</td>
<td>211000</td>
<td>12800</td>
<td>$242,500</td>
</tr>
<tr>
<td>Advertising</td>
<td>18500</td>
<td>176000</td>
<td>34500</td>
<td>$229,000</td>
</tr>
<tr>
<td>Permanent Staff Salaries</td>
<td>186000</td>
<td>1270000</td>
<td>201000</td>
<td>$1,657,000</td>
</tr>
<tr>
<td>Benefits Including Insurance</td>
<td>3900</td>
<td>11100</td>
<td>17500</td>
<td>$32,500</td>
</tr>
<tr>
<td>Total</td>
<td>227100</td>
<td>1668100</td>
<td>265800</td>
<td>$2,161,000</td>
</tr>
</tbody>
</table>

Producing a Simple Report

Without any specifications, the COMPUTAB procedure transposes and prints the input data set. The variables in the input data set become rows in the report, and the observations in the input data set become columns. The variable names are used as the row titles. The column headings default to COL1 through COLn. For example, the following input data set contains the monthly expenses reported by different divisions of the company:
data report;
    length compdiv $ 1;
    input compdiv $ date:date7. salary travel insure advrtise;
    format date date7.;
    label travel = 'Travel Expenses within U.S.'
                advrtise = 'Advertising'
                salary = 'Permanent Staff Salaries'
                insure = 'Benefits Including Insurance';
datalines;
A 31JAN1989 95000 10500 2000 6500
B 31JAN1989 668000 112000 5600 90000
C 31JAN1989 105000 6800 9000 18500
A 28FEB1989 91000 8200 1900 12000
B 28FEB1989 602000 99000 5500 86000
C 28FEB1989 96000 6000 8500 16000
;
You can get a listing of the data set by using the PRINT procedure, as follows:

title 'Listing of Monthly Divisional Expense Data';
proc print data=report;
run;

Figure 9.2 Listing of Data Set by PROC PRINT

Listing of Monthly Divisional Expense Data

<table>
<thead>
<tr>
<th>Obs</th>
<th>compdiv</th>
<th>date</th>
<th>salary</th>
<th>travel</th>
<th>insure</th>
<th>advrtise</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>31JAN89</td>
<td>95000</td>
<td>10500</td>
<td>2000</td>
<td>6500</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>31JAN89</td>
<td>668000</td>
<td>112000</td>
<td>5600</td>
<td>90000</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>31JAN89</td>
<td>105000</td>
<td>6800</td>
<td>9000</td>
<td>18500</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>28FEB89</td>
<td>91000</td>
<td>8200</td>
<td>1900</td>
<td>12000</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>28FEB89</td>
<td>602000</td>
<td>99000</td>
<td>5500</td>
<td>86000</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>28FEB89</td>
<td>96000</td>
<td>6000</td>
<td>8500</td>
<td>16000</td>
</tr>
</tbody>
</table>

To get a simple, transposed report of the same data set, use the following PROC COMPUTAB statement:

title 'Monthly Divisional Expense Report';
proc computab data=report;
run;

Figure 9.3 Listing of Data Set by PROC COMPUTAB

Monthly Divisional Expense Report

<table>
<thead>
<tr>
<th>compdiv</th>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
<th>COL4</th>
<th>COL5</th>
<th>COL6</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>31JAN89</td>
<td>31JAN89</td>
<td>31JAN89</td>
<td>28FEB89</td>
<td>28FEB89</td>
<td>28FEB89</td>
</tr>
<tr>
<td>salary</td>
<td>95000.00</td>
<td>668000.00</td>
<td>105000.00</td>
<td>91000.00</td>
<td>602000.00</td>
<td>96000.00</td>
</tr>
<tr>
<td>travel</td>
<td>10500.00</td>
<td>112000.00</td>
<td>6800.00</td>
<td>8200.00</td>
<td>99000.00</td>
<td>6000.00</td>
</tr>
<tr>
<td>insure</td>
<td>2000.00</td>
<td>5600.00</td>
<td>9000.00</td>
<td>1900.00</td>
<td>5500.00</td>
<td>8500.00</td>
</tr>
<tr>
<td>advrtise</td>
<td>6500.00</td>
<td>90000.00</td>
<td>18500.00</td>
<td>12000.00</td>
<td>86000.00</td>
<td>16000.00</td>
</tr>
</tbody>
</table>
**Using PROC COMPUTAB**

The COMPUTAB procedure is best understood by examining the following features:

- definition of the report layout with ROWS and COLUMNS statements
- input block
- row blocks
- column blocks

PROC COMPUTAB builds a table according to the specifications in the ROWS and COLUMNS statements. Row names and column names define the rows and columns of the table. Options in the ROWS and COLUMNS statements control titles, spacing, and formatting.

The input block places input observations into the appropriate columns of the report. It consists of programming statements used to select observations to be included in the report, to determine the column into which the observation should be placed, and to calculate row and column values that are not in the input data set.

Row blocks and column blocks perform operations on the values of rows and columns of the report after the input block has executed. Row blocks are a block of programming statements labeled ROWxxxx: that create or modify row values; column blocks are a block of programming statements labeled COLxxxx: that create or modify column values. Row and column blocks can make multiple passes through the report for final calculations.

For most reports, these features are sufficient. More complicated applications might require knowledge of the program data vector and the COMPUTAB data table. These topics are discussed in the section “Details: COMPUTAB Procedure” on page 480.

**Defining Report Layout**

ROWS and COLUMNS statements define the rows and columns of the report. The order of row and column names in these statements determines the order of rows and columns in the report. Additional ROWS and COLUMNS statements can be used to specify row and column formatting options.

The following statements select and order the variables from the input data set and produce the report in Figure 9.4:

```plaintext
proc computab data=report;
   rows travel advertise salary;
run;
```
Adding Computed Rows and Columns

In addition to the variables and observations in the input data set, you can create additional rows or columns by using SAS programming statements in PROC COMPUTAB. You can do the following:

- modify input data and select columns in the input block
- create or modify columns in column blocks
• create or modify rows in row blocks

The following statements add one computed row (SUM) and one computed column (TOTAL) to the report in Figure 9.5. In the input block the logical operators indicate the observations that correspond to each column of the report. After the input block reads in the values from the input data set, the column block creates the column variable TOTAL by summing the columns A, B, and C. The additional row variable, SUM, is calculated as the sum of the other rows. The result is shown in Figure 9.6.

```plaintext
proc computab data= report;
   rows travel advertise salary insure sum;
   columns a b c total;
   a = compdiv = 'A';
   b = compdiv = 'B';
   c = compdiv = 'C';
   colblk: total = a + b + c;
   rowblk: sum = travel + advertise + salary + insure;
run;
```

**Figure 9.6** Report Produced Using Row and Column Blocks

**Monthly Divisional Expense Report**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAVEL</td>
<td>18700.00</td>
<td>211000.00</td>
<td>12800.00</td>
<td>242500.00</td>
</tr>
<tr>
<td>ADVERTISE</td>
<td>18500.00</td>
<td>176000.00</td>
<td>34500.00</td>
<td>229000.00</td>
</tr>
<tr>
<td>SALARY</td>
<td>186000.00</td>
<td>1270000.00</td>
<td>201000.00</td>
<td>1657000.00</td>
</tr>
<tr>
<td>INSURE</td>
<td>3900.00</td>
<td>11100.00</td>
<td>17500.00</td>
<td>32500.00</td>
</tr>
<tr>
<td>SUM</td>
<td>227100.00</td>
<td>1668100.00</td>
<td>265800.00</td>
<td>2161000.00</td>
</tr>
</tbody>
</table>

**Enhancing the Report**

To enhance the appearance of the final report, you can use the following:

• TITLE and LABEL statements
• column headings
• row titles
• row and column spacing control
• overlining and underlining
• formats

The following example enhances the report in the previous example. The enhanced report is shown in Figure 9.7.

The TITLE statement assigns the report title. The column headings in Figure 9.7 (Division A, Division B, and Division C) are assigned in the first COLUMNS statement by “Division” _name_ specification. The
second COLUMNS statement assigns the column heading ("All" "Divisions"), sets the spacing (+4), and formats the values in the TOTAL column.

Similarly, the first ROWS statement uses previously assigned variable labels for row labels by specifying the _LABEL_ option. The DUL option in the second ROWS statement double-underlines the INSURE row. The third ROWS statement assigns the row label TOTAL to the SUM row.

title 'Year to Date Expenses';
proc computab cwidth=8 cdec=0;
    columns a b c / 'Division' _name_;
    columns total / 'All' 'Divisions' +4 f=dollar10.0;
    rows travel advrtise salary insure / _label_;
    rows insure / dul;
    rows sum / 'Total';
    a = compdiv = 'A';
    b = compdiv = 'B';
    c = compdiv = 'C';
    colblk: total = a + b + c;
    rowblk: sum = travel + advrtise + salary + insure;
run;

Figure 9.7 Report Produced by PROC COMPUTAB Using Enhancements

Year to Date Expenses

<table>
<thead>
<tr>
<th></th>
<th>Division</th>
<th>Division</th>
<th>Division</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>Divisions</td>
</tr>
<tr>
<td>Travel Expenses within U.S.</td>
<td>18700</td>
<td>21100</td>
<td>12800</td>
<td>$242,500</td>
</tr>
<tr>
<td>Advertising</td>
<td>18500</td>
<td>176000</td>
<td>34500</td>
<td>$229,000</td>
</tr>
<tr>
<td>Permanent Staff Salaries</td>
<td>186000</td>
<td>1270000</td>
<td>201000</td>
<td>$1,657,000</td>
</tr>
<tr>
<td>Benefits Including Insurance</td>
<td>3900</td>
<td>11100</td>
<td>17500</td>
<td>$32,500</td>
</tr>
<tr>
<td>Total</td>
<td>227100</td>
<td>1668100</td>
<td>265800</td>
<td>$2,161,000</td>
</tr>
</tbody>
</table>

Syntax: COMPUTAB Procedure

The following statements are used with the COMPUTAB procedure:

PROC COMPUTAB options ;
    BY variables ;
    COLUMNS column-list / options ;
    ROWS row-list / options ;
    CELL cell-names / FORMAT= format ;
    INIT anchor-name locator-name values locator-name values ;
    programming statements ;
    SUMBY variables ;
The PROC COMPUTAB statement is the only required statement. The COLUMNS, ROWS, and CELL statements define the COMPUTAB table. The INIT statement initializes the COMPUTAB table values. Programming statements process COMPUTAB table values. The BY and SUMBY statements provide BY-group processing and consolidation (roll up) tables.

### Functional Summary

Table 9.1 summarizes the COMPUTAB procedure statements and options.

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Statements</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify BY-group processing</td>
<td>BY</td>
<td></td>
</tr>
<tr>
<td>Specify the format for printing a particular cell</td>
<td>CELL</td>
<td></td>
</tr>
<tr>
<td>Define columns of the report</td>
<td>COLUMNS</td>
<td></td>
</tr>
<tr>
<td>Initialize values in the COMPUTAB data table</td>
<td>INIT</td>
<td></td>
</tr>
<tr>
<td>Define rows of the report</td>
<td>ROWS</td>
<td></td>
</tr>
<tr>
<td>Produce consolidation tables</td>
<td>SUMBY</td>
<td></td>
</tr>
<tr>
<td><strong>Data Set Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify the input data set</td>
<td>COMPUTAB</td>
<td>DATA=</td>
</tr>
<tr>
<td>Specify an output data set</td>
<td>COMPUTAB</td>
<td>OUT=</td>
</tr>
<tr>
<td><strong>Input Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify a value to use when testing for 0</td>
<td>COMPUTAB</td>
<td>FUZZ=</td>
</tr>
<tr>
<td>Initialize the data table to missing</td>
<td>COMPUTAB</td>
<td>INITMISS</td>
</tr>
<tr>
<td>Prevent the transposition of the input data set</td>
<td>COMPUTAB</td>
<td>NOTRANS</td>
</tr>
<tr>
<td><strong>Printing Control Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suppress printing of the listed columns</td>
<td>COLUMNS</td>
<td>NOPRINT</td>
</tr>
<tr>
<td>Suppress all printed output</td>
<td>COMPUTAB</td>
<td>NOPRINT</td>
</tr>
<tr>
<td>Suppress printing of the listed rows</td>
<td>ROWS</td>
<td>NOPRINT</td>
</tr>
<tr>
<td>Suppress columns with all 0 or missing values</td>
<td>COLUMNS</td>
<td>NOZERO</td>
</tr>
<tr>
<td>Suppress rows with all 0 or missing values</td>
<td>ROWS</td>
<td>NOZERO</td>
</tr>
<tr>
<td>List option values</td>
<td>COMPUTAB</td>
<td>OPTIONS</td>
</tr>
<tr>
<td>Overprint titles, values, overlining, and underlining associated with listed rows</td>
<td>ROWS</td>
<td>OVERPRINT</td>
</tr>
<tr>
<td>Print only consolidation tables</td>
<td>COMPUTAB</td>
<td>SUMONLY</td>
</tr>
<tr>
<td><strong>Report Formatting Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify number of decimal places to print</td>
<td>COMPUTAB</td>
<td>CDEC=</td>
</tr>
<tr>
<td>Specify number of spaces between columns</td>
<td>COMPUTAB</td>
<td>CSPACE=</td>
</tr>
<tr>
<td>Specify column width for the report</td>
<td>COMPUTAB</td>
<td>CWIDTH=</td>
</tr>
<tr>
<td>Overline the listed rows with double lines</td>
<td>ROWS</td>
<td>DOL</td>
</tr>
<tr>
<td>Underline the listed rows with double lines</td>
<td>ROWS</td>
<td>DUL</td>
</tr>
</tbody>
</table>
### Table 9.1 continued

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify a format for printing the cell values</td>
<td>CELL</td>
<td>FORMAT=</td>
</tr>
<tr>
<td>Specify a format for printing column values</td>
<td>COLUMNS</td>
<td>FORMAT=</td>
</tr>
<tr>
<td>Specify a format for printing the row values</td>
<td>ROWS</td>
<td>FORMAT=</td>
</tr>
<tr>
<td>Left-align the column headings</td>
<td>COLUMNS</td>
<td>LJC</td>
</tr>
<tr>
<td>Left-justify character rows in each column</td>
<td>ROWS</td>
<td>LJC</td>
</tr>
<tr>
<td>Specify indentation from the margin</td>
<td>ROWS</td>
<td>+n</td>
</tr>
<tr>
<td>Suppress printing of row titles on later pages</td>
<td>COMPUTAB</td>
<td>NORTR</td>
</tr>
<tr>
<td>Overline the listed rows with a single line</td>
<td>ROWS</td>
<td>OL</td>
</tr>
<tr>
<td>Start a new page before printing the listed rows</td>
<td>ROWS</td>
<td><em>PAGE</em></td>
</tr>
<tr>
<td>Specify number of spaces before row titles</td>
<td>COMPUTAB</td>
<td>RTS=</td>
</tr>
<tr>
<td>Print a blank row</td>
<td>ROWS</td>
<td>SKIP</td>
</tr>
<tr>
<td>Underline the listed rows with a single line</td>
<td>ROWS</td>
<td>UL</td>
</tr>
<tr>
<td>Specify text to print if column is 0 or missing</td>
<td>COLUMNS</td>
<td>ZERO=</td>
</tr>
<tr>
<td>Specify text to print if row is 0 or missing</td>
<td>ROWS</td>
<td>ZERO=</td>
</tr>
</tbody>
</table>

#### Row and Column Type Options
- Specify that columns contain character data: COLUMNS CHAR
- Specify that rows contain character data: ROWS CHAR

#### Options for Column Headings
- Specify literal column headings: COLUMNS ‘column heading’
- Use variable labels in column headings: COLUMNS _LABEL_
- Specify a master title centered over columns: COLUMNS MTITLE= 
- Use column names in column headings: COLUMNS _NAME_

#### Options for Row Titling
- Use labels in row titles: ROWS _LABEL_
- Use row names in row titles: ROWS _NAME_
- Specify literal row titles: ROWS ‘row title’

The following sections describe the PROC COMPUTAB statement and then describe the other statements in alphabetical order.

### PROC COMPUTAB Statement

```plaintext
PROC COMPUTAB options ;
```

The following options can be used in the PROC COMPUTAB statement.
Chapter 9: The COMPUTAB Procedure

Input Options

**DATA=SAS-data-set**

names the SAS data set that contains the input data. If this option is not specified, the last created data set is used. If you are not reading a data set, use DATA=_NULL_.

**FUZZ=value**

specifies the criterion to use when testing for 0. If a number is within the FUZZ= value of 0, the number is set to 0.

**INITMISS**

initializes the COMPUTAB data table to missing rather than to 0. The COMPUTAB data table is discussed further in the section “Details: COMPUTAB Procedure” on page 480.

**NOTRANSPOSE**

**NOTRANS**

prevents the transposition of the input data set in building the COMPUTAB report tables. The NOTRANS option causes input data set variables to appear among the columns of the report rather than among the rows.

Report Formatting Options

The formatting options specify default values. Many of the formatting options can be modified for specific columns in COLUMNS statements and for rows in ROWS statements.

**CDEC=d**

specifies the default number of decimal places for printing. The default is CDEC=2. See the FORMAT= option in the sections on the COLUMN, ROWS, and CELL statements later in this chapter.

**CSPACE=n**

specifies the default number of spaces to insert between columns. The value of the CSPACE= option is used as the default value for the +n option in the COLUMNS statement. The default is CSPACE=2.

**CWIDTH=w**

specifies a default column width for the report. The default is CWIDTH=9. The width must be in the range of 1–32.

**NORTR**

suppresses the printing of row titles on each page. The NORTR (no row-title repeat) option is useful to suppress row titles when report pages are to be joined together in a larger report.

**RTS=n**

specifies the default number of spaces to be inserted before row titles when row titles appear after the first printed column. The default row-title spacing is RTS=2.

Output Options

**NOPRINT**

suppresses all printed output. Use the NOPRINT option with the OUT= option to produce an output data set but no printed reports.
OPTIONS
lists PROC COMPUTAB option values. The option values appear on a separate page preceding the procedure’s normal output.

OUT=SAS-data-set
names the SAS data set to contain the output data. For a description of the structure of the output data set, see the section “Details: COMPUTAB Procedure” on page 480.

SUMONLY
suppresses printing of detailed reports. When the SUMONLY option is used, PROC COMPUTAB generates and prints only consolidation tables as specified in the SUMBY statement.

BY Statement

BY variables;

A BY statement can be used with PROC COMPUTAB to obtain separate reports for observations in groups defined by the BY variables. At the beginning of each BY group, before PROC COMPUTAB reads any observations, all table values are set to 0 unless the INITMISS option or an INIT statement is specified.

CELL Statement

CELL cell-names / FORMAT=format;

The CELL statement specifies the format for printing a particular cell in the COMPUTAB data table. Cell variable names are compound SAS names of the form name1.name2, where name1 is the name of a row variable and name2 is the name of a column variable. Formats specified with the FORMAT= option in CELL statements override formats specified in ROWS and COLUMNS statements.

COLUMNS Statement

COLUMNS column-list / options;

COLUMNS statements define the columns of the report. The COLUMNS statement can be abbreviated COLUMN, COLS, or COL.

The specified column names must be valid SAS names. Abbreviated lists, as described in SAS Language: Reference, can also be used.

You can use as many COLUMNS statements as you need. A COLUMNS statement can describe more than one column, and one column of the report can be described with several different COLUMNS statements. The order of the columns on the report is determined by the order of appearance of column names in COLUMNS statements. The first occurrence of the name determines where in the sequence of columns a particular column is located.

The following options can be used in the COLUMNS statement.
**Option for Column Type**

**CHAR**

indicates that the columns contain character data.

**Options for Column Headings**

You can specify as many lines of column headings as needed. If no options are specified, the column names from the COLUMNS statement are used as column headings. Any or all of the following options can be used in a column heading:

"column heading"

specifies that the characters enclosed in quotes are to be used in the column heading for the variable or variables listed in the COLUMNS statement. Each quoted string appears on a separate line of the heading.

_LABEL_

uses labels, if provided, in the heading for the column or columns listed in the COLUMNS statement. If a label has not been provided, the name of the column is used. For information about the LABEL statement, see *SAS Language: Reference*.

**MTITLE=** "text"

specifies that the string of characters enclosed in quotes is a master title to be centered over all the columns listed in the COLUMNS statement. The list of columns must be consecutive. Special characters ("+", "*", "=" , and so forth) placed on either side of the text expand to fill the space. The MTITLE= option can be abbreviated M=.

_NAME_

uses column names in column headings for the columns listed in the COLUMNS statement. This option allows headings ("text") and names to be combined in a heading.

**Options for Column Print Control**

+n

inserts n spaces before each column listed in the COLUMNS statement. The default spacing is given by the CSPACE= option in the PROC COMPUTAB statement.

NOPRINT

suppresses printing of columns listed in the COLUMNS statement. This option enables you to create columns to be used for intermediate calculations without having those columns printed.

NOZERO

suppresses printing of columns when all the values in a column are 0 or missing. Numbers within the FUZZ= value of 0 are treated as 0.

_PAGE_

starts a new page of the report before printing each of the columns in the list that follows.
_TITLES_
prints row titles before each column in the list. The _TITLES_ option can be abbreviated as _TITLE_.

Options for Column Formatting

Column formats override row formats for particular table cells only when the input data set is not transposed (when the NOTRANS option is specified).

FORMAT= \textit{format} \newline
specifies a format for printing the values of the columns listed in the COLUMNS statement. The FORMAT= option can be abbreviated F=.

LJC \newline
left-justifies the column headings for the columns listed. By default, columns are right-justified. When the LJC (left-justify character) option is used, any character row values in the column are also left-justified rather than right-justified.

ZERO= \textit{“text”} \newline
substitutes \textit{“text”} when the value in the column is 0 or missing.

\textbf{INIT Statement}

\texttt{INIT anchor-name [locator-name] values [locator-name values] ;}

The INIT statement initializes values in the COMPUTAB data table at the beginning of each execution of the procedure and at the beginning of each BY group if a BY statement is present.

The INIT statement in the COMPUTAB procedure is similar in function to the RETAIN statement in the DATA step, which initializes values in the program data vector. The INIT statement can be used at any point after the variable to which it refers has been defined in COLUMNS or ROWS statements. Each INIT statement initializes one row or column. Any number of INIT statements can be used.

The first term after the keyword \textit{INIT}, \textit{anchor-name}, anchors initialization to a row or column. If \textit{anchor-name} is a row name, then all \textit{locator-name} values in the statement are columns of that row. If \textit{anchor-name} is a column name, then all \textit{locator-name} values in the statement are rows of that column.

The following terms appear in the INIT statement:

\textit{anchor-name} \newline
names the row or column in which values are to be initialized. This term is required.

\textit{locator-name} \newline
identifies the starting column in the row (or starting row in the column) into which values are to be placed. For example, in a table with a row SALES and a column for each month of the year, the following statement initializes values for columns JAN, FEB, and JUN:

\texttt{init sales jan 500 feb 600 jun 800;}

If you do not specify \textit{locator-name} values, the first value is placed into the first row or column, the second value into the second row or column, and so on. For example, the following statement assigns 500 to column JAN, 600 to FEB, and 450 to MAR:
### Chapter 9: The COMPUTAB Procedure

```
init sales 500 600 450;
```

The `+n` option specifies the number of columns in a row (or rows in a column) that are to be skipped when initializing values. For example, the following statement assigns 500 to JAN and 900 to JUL:

```
init sales jan 500 +5 900;
```

The `n*value` option assigns `value` to `n` columns in the row (or rows in the column). For example, both of the following statements assign 500 to columns JAN through JUN and 1000 to JUL through DEC:

```
init sales jan 6*500 jul 6*1000;
```

```
init sales 6*500 6*1000;
```

---

### ROWS Statement

The `ROWS` statement defines the rows of the report. The `ROWS` statement can be abbreviated `ROW`.

The specified row names must be valid SAS names. Abbreviated lists, as described in *SAS Language: Reference*, can also be used.

You can use as many `ROWS` statements as you need. A `ROWS` statement can describe more than one row, and one row of the report can be described with several different `ROWS` statements. The order of the rows in the report is determined by the order of appearance of row names in `ROWS` statements. The first occurrence of the name determines where the row is located.

The following options can be used in the `ROWS` statement.

#### Option for Row Type

- **CHAR**
  - Indicates that the rows contain character data.

#### Options for Row Titling

You can specify as many lines of row titles as needed. If no options are specified, the names from the `ROWS` statement are used as row titles. Any or all of the following options can be used in a row title:

- **_LABEL_**
  - Uses labels as row titles for the row or rows listed in the `ROWS` statement. If a label is not provided, the name of the row is substituted. For more information about the `LABEL` statement, see *SAS Language: Reference*. 

---

![Image](image-url)
_NAME_
uses row names in row titles for the row or rows listed in the ROWS statement.

"row title"
specifies that the string of characters enclosed in quotes is to be used in the row title for the row or rows listed in the ROWS statement. Each quoted string appears on a separate line of the heading.

**Options for Row Print Control**

+n
indents \( n \) spaces from the margin for the rows in the ROWS statement.

DOL
overlines the rows listed in the ROWS statement with double lines. Overlines are printed on the line before any row titles or data for the row.

DUL
underlines the rows listed in the ROWS statement with double lines. Underlines are printed on the line after the data for the row. A row can have both an underline and an overline option.

NOPRINT
suppresses printing of the rows listed in the ROWS statement. This option enables you to create rows to be used for intermediate calculations without having those rows printed.

NOZERO
suppresses the printing of a row when all the values are 0 or missing.

OL
overlines the rows listed in the ROWS statement with a single line. Overlines are printed on the line before any row titles or data for the row.

OVERPRINT
overprints titles, values, overlining, and underlining associated with rows listed in the ROWS statement. The OVERPRINT option can be abbreviated OVP. This option is valid only when the system option OVP is in effect. For more information about the OVP option, see [SAS Language: Reference](#).

_PAGE_
starts a new page of the report before printing these rows.

SKIP
prints a blank line after the data lines for these rows.

UL
underlines the rows listed in the ROWS statement with a single line. Underlines are printed on the line after the data for the row. A row can have both an underline and an overline option.

**Options for Row Formatting**

Row formatting options take precedence over column-formatting options when the input data set is transposed. Row print width can never be wider than column width. Character values are truncated on the right.
Chapter 9: The COMPUTAB Procedure

**FORMAT=format**
specifies a format for printing the values of the rows listed in the ROWS statement. The **FORMAT=** option can be abbreviated as **F=**.

**LJC**
left-justifies character rows in each column.

**ZERO=“text”**
substitutes “text” when the value in the row is 0 or missing.

---

**SUMBY Statement**

**SUMBY variables ;**

The **SUMBY** statement produces consolidation tables for variables whose names are in the **SUMBY** list. Only one **SUMBY** statement can be used.

To use a **SUMBY** statement, you must use a **BY** statement. The **SUMBY** and **BY** variables must be in the same relative order in both statements. For example:

```
by a b c;
sumby a b;
```

This **SUMBY** statement produces tables that consolidate over values of **C** within levels of **B** and over values of **B** within levels of **A**. Suppose **A** has values 1, 2; **B** has values 1, 2; and **C** has values 1, 2, 3. Table 9.2 indicates the consolidation tables produced by the **SUMBY** statement.

**Table 9.2** Consolidation Tables Produced by the **SUMBY** Statement

<table>
<thead>
<tr>
<th>SUMBY Consolidations</th>
<th>Consolidated BY Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=1, B=1</td>
<td>C=1</td>
</tr>
<tr>
<td>A=1, B=2</td>
<td>C=1</td>
</tr>
<tr>
<td>A=1</td>
<td>B=1, C=1</td>
</tr>
<tr>
<td>A=2, B=1</td>
<td>C=1</td>
</tr>
<tr>
<td>A=2, B=2</td>
<td>C=1</td>
</tr>
<tr>
<td>A=2</td>
<td>B=1, C=1</td>
</tr>
</tbody>
</table>

Two consolidation tables for **B** are produced for each value of **A**. The first table consolidates the three tables produced for the values of **C** while **B** is 1; the second table consolidates the three tables produced for **C** while **B** is 2.

Tables are similarly produced for values of **A**. Nested consolidation tables are produced for **B** (as described previously) for each value of **A**. Thus, this **SUMBY** statement produces a total of six consolidation tables in addition to the tables produced for each **BY** group.
To produce a table that consolidates the entire data set (the equivalent of using PROC COMPUTAB with neither BY nor SUMBY statements), use the special name _TOTAL_ as the first entry in the SUMBY variable list. For example:

```
sumby _total_ a b;
```

PROC COMPUTAB then produces consolidation tables for SUMBY variables as well as a consolidation table for all observations.

To produce only consolidation tables, use the SUMONLY option in the PROC COMPUTAB statement.

---

**Programme Statements**

You can use most SAS programming statements the same way you use them in the DATA step. Also, all DATA step functions can be used in the COMPUTAB procedure.

Lines written by the PUT statement are not integrated with the COMPUTAB report. PUT statement output is written to the SAS log.

The automatic variable _N_ can be used; its value is the number of observations read or the number read in the current BY group, if a BY statement is used. FIRST.variable and LAST.variable references cannot be used.

The following statements are also available in PROC COMPUTAB:

- `ABORT`  - `FORMAT`
- `ARRAY`  - `GOTO`
- `ATTRIB`  - `IF-THEN/ELSE`
- `assignment statement`  - `LABEL`
- `CALL`  - `LINK`
- `DELETE`  - `PUT`
- `DO`  - `RETAINT`
- `iterative DO`  - `SELECT`
- `DO UNTIL`  - `STOP`
- `DO WHILE`  - `sum statement`
- `END`  - `TITLE`
- `FOOTNOTE`

The programming statements can be assigned labels `ROWxxxx` or `COLxxxx` to indicate the start of a row and column block, respectively. Statements in a row block create or change values in all the columns in the specified rows. Similarly, statements in a column block create or change values in all the rows in the specified columns.

There is an implied RETURN statement before each new row or column block. Thus, the flow of execution does not leave the current row (column) block before the block repeats for all columns (rows.) Row and column variables and nonretained variables are initialized prior to each execution of the block.

The next `COLxxxx` label, `ROWxxxx` label, or the end of the PROC COMPUTAB step signals the end of a row (column) block. Column blocks and row blocks can be mixed in any order. In some cases, performing calculations in different orders can lead to different results.

For more information, see the sections “Program Flow Example” on page 480, “Order of Calculations” on page 482, and “Controlling Execution within Row and Column Blocks” on page 485.
Details: COMPUTAB Procedure

Program Flow Example

This example shows how the COMPUTAB procedure processes observations in the program working storage and the COMPUTAB data table (CDT).

Assume you have three years of figures for sales and cost of goods sold (CGS), and you want to determine total sales and cost of goods sold and calculate gross profit and the profit margin.

```plaintext
data example;
   input year sales cgs;
datalines;
  1988 83 52
  1989 106 85
  1990 120 114
;

proc computab data=example;

   columns c88 c89 c90 total;
   rows sales cgs gprofit pctmarg;

   /* calculate gross profit */
   gprofit = sales - cgs;

   /* select a column */
   c88 = year = 1988;
   c89 = year = 1989;
   c90 = year = 1990;

   /* calculate row totals for sales */
   /* and cost of goods sold */
   col: total = c88 + c89 + c90;

   /* calculate profit margin */
   row: pctmarg = gprofit / cgs * 100;
run;
```

Table 9.3 shows the CDT before any observation is read in. All the columns and rows are defined with the values initialized to 0.

<table>
<thead>
<tr>
<th></th>
<th>C88</th>
<th>C89</th>
<th>C90</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALES</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CGS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GPROFIT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PCTMARG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
When the first input is read in (YEAR=1988, SALES=83, and CGS=52), the input block puts the values for SALES and CGS in the C88 column since year=1988. Also the value for the gross profit for that year (GPROFIT) is calculated as indicated in the following statements:

\[
\text{gprofit} = \text{sales-cgs}; \\
c88 = \text{year} = 1988; \\
c89 = \text{year} = 1989; \\
c90 = \text{year} = 1990;
\]

Table 9.4 shows the CDT after the first observation is input.

**Table 9.4  CDT after First Observation Input (C88=1)**

<table>
<thead>
<tr>
<th></th>
<th>C88</th>
<th>C89</th>
<th>C90</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALES</td>
<td>83</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CGS</td>
<td>52</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GPROFIT</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PCTMARG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Similarly, the second observation (YEAR=1989, SALES=106, CGS=85) is put in the second column, and the GPROFIT is calculated to be 21. The third observation (YEAR=1990, SALES=120, CGS=114) is put in the third column, and the GPROFIT is calculated to be 6. Table 9.5 shows the CDT after all observations are input.

**Table 9.5  CDT after All Observations Input**

<table>
<thead>
<tr>
<th></th>
<th>C88</th>
<th>C89</th>
<th>C90</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALES</td>
<td>83</td>
<td>106</td>
<td>120</td>
<td>0</td>
</tr>
<tr>
<td>CGS</td>
<td>52</td>
<td>85</td>
<td>114</td>
<td>0</td>
</tr>
<tr>
<td>GPROFIT</td>
<td>31</td>
<td>21</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>PCTMARG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

After the input block is executed for each observation in the input data set, the first row or column block is processed. In this case, the column block is

\[
\text{col: total} = \text{c88} + \text{c89} + \text{c90};
\]

The column block executes for each row, calculating the TOTAL column for each row. Table 9.6 shows the CDT after the column block has executed for the first row (TOTAL=83 + 106 + 120). The total sales for the three years is 309.

**Table 9.6  CDT after Column Block Executed for First Row**

<table>
<thead>
<tr>
<th></th>
<th>C88</th>
<th>C89</th>
<th>C90</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALES</td>
<td>83</td>
<td>106</td>
<td>120</td>
<td>309</td>
</tr>
<tr>
<td>CGS</td>
<td>52</td>
<td>85</td>
<td>114</td>
<td>0</td>
</tr>
<tr>
<td>GPROFIT</td>
<td>31</td>
<td>21</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>PCTMARG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 9.7 shows the CDT after the column block has executed for all rows and the values for total cost of goods sold and total gross profit have been calculated.

**Table 9.7**  CDT after Column Block Executed for All Rows

<table>
<thead>
<tr>
<th></th>
<th>C88</th>
<th>C89</th>
<th>C90</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALES</td>
<td>83</td>
<td>106</td>
<td>120</td>
<td>309</td>
</tr>
<tr>
<td>CGS</td>
<td>52</td>
<td>85</td>
<td>114</td>
<td>251</td>
</tr>
<tr>
<td>GPROFIT</td>
<td>31</td>
<td>21</td>
<td>6</td>
<td>58</td>
</tr>
<tr>
<td>PCTMARG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

After the column block has been executed for all rows, the next block is processed. The row block is

\[
\text{row: pctmarg} = \frac{\text{gprofit}}{\text{cgs}} \times 100;
\]

The row block executes for each column, calculating the PCTMARG for each year and the total (TOTAL column) for three years. Table 9.8 shows the CDT after the row block has executed for all columns.

**Table 9.8**  CDT after Row Block Executed for All Columns

<table>
<thead>
<tr>
<th></th>
<th>C88</th>
<th>C89</th>
<th>C90</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALES</td>
<td>83</td>
<td>106</td>
<td>120</td>
<td>309</td>
</tr>
<tr>
<td>CGS</td>
<td>52</td>
<td>85</td>
<td>114</td>
<td>251</td>
</tr>
<tr>
<td>GPROFIT</td>
<td>31</td>
<td>21</td>
<td>6</td>
<td>58</td>
</tr>
<tr>
<td>PCTMARG</td>
<td>59.62</td>
<td>24.71</td>
<td>5.26</td>
<td>23.11</td>
</tr>
</tbody>
</table>

---

**Order of Calculations**

The COMPUTAB procedure provides alternative programming methods for performing most calculations. New column and row values are formed by adding values from the input data set, directly or with modification, into existing columns or rows. New columns can be formed in the input block or in column blocks. New rows can be formed in the input block or in row blocks.

This example illustrates the different ways to collect totals. Table 9.9 is the total sales report for two products, SALES1 and SALES2, during the years 1988–1990. The values for SALES1 and SALES2 in columns C88, C89, and C90 come from the input data set.

**Table 9.9**  Total Sales Report

<table>
<thead>
<tr>
<th></th>
<th>C88</th>
<th>C89</th>
<th>C90</th>
<th>SALESTOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALES1</td>
<td>15</td>
<td>45</td>
<td>80</td>
<td>140</td>
</tr>
<tr>
<td>SALES2</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>120</td>
</tr>
<tr>
<td>YRTOT</td>
<td>45</td>
<td>85</td>
<td>130</td>
<td>260</td>
</tr>
</tbody>
</table>
The new column SALESTOT, which is the total sales for each product over three years, can be computed in several different ways:

- in the input block by selecting SALESTOT for each observation:

  \[ \text{salestot} = 1; \]

- in a column block:

  \[ \text{coltot: salestot} = c88 + c89 + c90; \]

In a similar fashion, the new row YRTOT, which is the total sales for each year, can be formed as follows:

- in the input block:

  \[ \text{yrtot} = \text{sales1} + \text{sales2}; \]

- in a row block:

  \[ \text{rowtot: yrtot} = \text{sales1} + \text{sales2}; \]

Performing some calculations in PROC COMPUTAB in different orders can yield different results, because many operations are not commutative. Be sure to perform calculations in the proper sequence. It might take several column and row blocks to produce the desired report values.

Notice that in the previous example, the grand total for all rows and columns is 260 and is the same whether it is calculated from row subtotals or column subtotals. It makes no difference in this case whether you compute the row block or the column block first.

However, consider the example in Table 9.10, where a new column and a new row are formed.

<table>
<thead>
<tr>
<th>STORE1</th>
<th>STORE2</th>
<th>STORE3</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT1</td>
<td>12</td>
<td>13</td>
<td>27</td>
</tr>
<tr>
<td>PRODUCT2</td>
<td>11</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>TOTAL</td>
<td>23</td>
<td>28</td>
<td>41</td>
</tr>
</tbody>
</table>

The new column MAX contains the maximum value in each row, and the new row TOTAL contains the column totals. MAX is calculated in a column block:
Chapter 9: The COMPUTAB Procedure

col: max = max(store1, store2, store3);
TOTAL is calculated in a row block:
row: total = product1 + product2;

Notice that either of two values, 41 or 42, is possible for the element in column MAX and row TOTAL. If the row block is first, the value is the maximum of the column totals (41). If the column block is first, the value is the sum of the MAX values (42). Whether to compute a column block before a row block can be a critical decision.

Column Selection

The following discussion assumes that the NOTRANS option has not been specified. When NOTRANS is specified, this section applies to rows rather than columns.

If a COLUMNS statement appears in PROC COMPUTAB, a target column must be selected for the incoming observation. If there is no COLUMNS statement, a new column is added for each observation. When a COLUMNS statement is present and the selection criteria fail to designate a column, the current observation is ignored. Faulty column selection can result in columns or entire tables of 0s (or missing values if the INITMISS option is specified).

During execution of the input block, when an observation is read, its values are copied into row variables in the program data vector (PDV).

To select columns, use either the column variable names themselves or the special variable _COL_. Use the column names by setting a column variable equal to some nonzero value. The example in the section “Getting Started: COMPUTAB Procedure” on page 464 uses the logical expression COMPDIV= value, and the result is assigned to the corresponding column variable.

```
a = compdiv = 'A';
b = compdiv = 'B';
c = compdiv = 'C';
```

IF statements can also be used to select columns. The following statements are equivalent to the preceding example:

```
if compdiv = 'A' then a = 1;
else if compdiv = 'B' then b = 1;
else if compdiv = 'C' then c = 1;
```

At the end of the input block for each observation, PROC COMPUTAB multiplies numeric input values by any nonzero selector values and adds the result to selected columns. Character values simply overwrite the contents already in the table. If more than one column is selected, the values are added to each of the selected columns.

Use the _COL_ variable to select a column by assigning the column number to it. The COMPUTAB procedure automatically initializes column variables and sets the _COL_ variable to 0 at the start of each execution of the input block. At the end of the input block for each observation, PROC COMPUTAB examines the value of _COL_. If the value is nonzero and within range, the row variable values are added to the CDT cells of the _COL_ th column. For example:
Controlling Execution within Row and Column Blocks

Row names, column names, and the special variables _ROW_ and _COL_ can be used to limit the execution of programming statements to selected rows or columns. A row block operates on all columns of the table for a specified row unless restricted in some way. Likewise, a column block operates on all rows for a specified column. Use column names or _COL_ in a row block to execute programming statements conditionally; use row names or _ROW_ in a column block.

For example, consider a simple column block that consists of only one statement:

```sas
col: total = qtr1 + qtr2 + qtr3 + qtr4;
```

This column block assigns a value to each row in the TOTAL column. As each row participates in the execution of a column block, the following changes occur:

- Its row variable in the program data vector is set to 1.
- The value of _ROW_ is the number of the participating row.
- The value from each column of the row is copied from the COMPUTAB data table to the program data vector.

To avoid calculating TOTAL on particular rows, use row names or _ROW_. For example:

```sas
col: if sales|cost then total = qtr1 + qtr2 + qtr3 + qtr4;
```

or

```sas
col: if _row_ < 3 then total = qtr1 + qtr2 + qtr3 + qtr4;
```

Row and column blocks can appear in any order, and rows and columns can be selected in each block.
Program Flow

This section describes in detail the different steps in PROC COMPUTAB execution.

Step 1: Define Report Organization and Set Up the COMPUTAB Data Table

Before the COMPUTAB procedure reads in data or executes programming statements, the columns list from the COLUMNS statements and the rows list from the ROWS statements are used to set up a matrix of all columns and rows in the report. This matrix is called the COMPUTAB data table (CDT). When you define columns and rows of the CDT, the COMPUTAB procedure also sets up corresponding variables in working storage called the program data vector (PDV) for programming statements. Data values reside in the CDT but are copied into the program data vector as they are needed for calculations.

Step 2: Select Input Data with Input Block Programming Statements

The input block copies input observations into rows or columns of the CDT. By default, observations go to columns; if the data set is not transposed (the NOTRANS option is specified), observations go to rows of the report table. The input block consists of all executable statements before any ROWxxxx: or COLxxxx: statement label. Use programming statements to perform calculations and select a given observation to be added into the report.

Input Block

The input block is executed once for each observation in the input data set. If there is no input data set, the input block is not executed. The program logic of the input block is as follows:

1. Determine which variables, row or column, are selector variables and which are data variables. Selector variables determine which rows or columns receive values at the end of the block. Data variables contain the values that the selected rows or columns receive. By default, column variables are selector variables and row variables are data variables. If the input data set is not transposed (the NOTRANS option is specified), the roles are reversed.

2. Initialize nonretained program variables (including selector variables) to 0 (or missing if the INITMISS option is specified). Selector variables are temporarily associated with a numeric data item supplied by the procedure. Using these variables to control row and column selection does not affect any other data values.

3. Transfer data from an observation in the data set to data variables in the PDV.

4. Execute the programming statements in the input block by using values from the PDV and storing results in the PDV.

5. Transfer data values from the PDV into the appropriate columns of the CDT. If a selector variable for a row or column has a nonmissing and nonzero value, multiply each PDV value for variables used in the report by the selector variable and add the results to the selected row or column of the CDT.
Step 3: Calculate Final Values by Using Column Blocks and Row Blocks

Column Blocks
A column block is executed once for each row of the CDT. The program logic of a column block is as follows:

1. Indicate the current row by setting the corresponding row variable in the PDV to 1 and the other row variables to missing. Assign the current row number to the special variable _ROW_.
2. Move values from the current row of the CDT to the respective column variables in the PDV.
3. Execute programming statements in the column block by using the column values in the PDV. Here new columns can be calculated and old ones adjusted.
4. Move the values back from the PDV to the current row of the CDT.

Row Blocks
A row block is executed once for each column of the CDT. The program logic of a row block is as follows:

1. Indicate the current column by setting the corresponding column variable in the PDV to 1 and the other column variables to missing. Assign the current column number to the special variable _COL_.
2. Move values from the current column of the CDT to the respective row variables in the PDV.
3. Execute programming statements in the row block by using the row values in the PDV. Here new rows can be calculated and old ones adjusted.
4. Move the values back from the PDV to the current column of the CDT.

See the section “Controlling Execution within Row and Column Blocks” on page 485.

Any number of column blocks and row blocks can be used. Each can include any number of programming statements.

The values of row variables and column variables are determined by the order in which different row-block and column-block programming statements are processed. These values can be modified throughout the COMPUTAB procedure, and final values are printed in the report.

Direct Access to Table Cells
You can insert or retrieve numeric values from specific table cells by using the special reserved name TABLE with row and column subscripts. References to the TABLE have the form

TABLE[ row-index, column-index ]

where row-index and column-index can be numbers, character literals, numeric variables, character variables, or expressions that produce a number or a name. If an index is numeric, it must be within range; if it is character, it must name a row or column.

References to TABLE elements can appear on either side of an equal sign in an assignment statement and can be used in a SAS expression.
Reserved Words

Certain words are reserved for special use by the COMPUTAB procedure, and using these words as variable names can lead to syntax errors or warnings. The reserved words are as follows:

- COLUMN
- COLUMNS
- COL
- COLS
- _COL_
- ROW
- ROWS
- _ROW_
- INIT
- _N_
- TABLE

Missing Values

Missing values for variables in programming statements are treated in the same way that missing values are treated in the DATA step; that is, missing values used in expressions propagate missing values to the result. For more information about missing values, see SAS Language: Reference.

Missing values in the input data are treated as follows in the COMPUTAB report table. At the end of the input block, either one or more rows or one or more columns can have been selected to receive values from the program data vector (PDV). Numeric data values from variables in the PDV are added into selected report table rows or columns. If a PDV value is missing, the values already in the selected rows or columns for that variable are unchanged by the current observation. Other values from the current observation are added to table values as usual.

OUT= Data Set

The output data set contains the following variables:

- BY variables
- a numeric variable _TYPE_
• a character variable _NAME_
• the column variables from the COMPUTAB data table

The BY variables contain values for the current BY group. For observations in the output data set from consolidation tables, the consolidated BY variables have missing values.

The special variable _TYPE_ is a numeric variable that can have one of three values: 1, 2, or 3. _TYPE_ = 1 indicates observations from the normal report table produced for each BY group; _TYPE_ = 2 indicates observations from the _TOTAL_ consolidation table; _TYPE_ = 3 indicates observations from other consolidation tables. _TYPE_ = 2 and _TYPE_ = 3 observations have one or more BY variables with missing values.

The special variable _NAME_ is a character variable of length 8 that contains the row or column name associated with the observation from the report table. If the input data set is transposed, _NAME_ contains column names; otherwise, _NAME_ contains row names.

If the input data set is transposed, the remaining variables in the output data set are row variables from the report table. They are column variables if the input data set is not transposed.

---

**NOTRANS Option**

The NOTRANS option in the PROC COMPUTAB statement prevents the transposition of the input data set. The NOTRANS option affects the input block, the precedence of row and column options, and the structure of the output data set if the OUT= option is specified.

When the input data set is transposed, input variables are among the rows of the COMPUTAB report, and observations compose columns. The reverse is true if the data set is not transposed; therefore, the input block must select rows to receive data values, and input variables are among the columns.

Variables from the input data set dominate the format specification and data type. When the input data set is transposed, input variables are among the rows of the report, and row options take precedence over column options. When the input data set is not transposed, input variables are among the columns, and column options take precedence over row options.

Variables for the output data set are taken from the dimension (row or column) that contains variables from the input data set. When the input data set is transposed, this dimension is the row dimension; otherwise, the output variables come from the column dimension.

---

**Examples: COMPUTAB Procedure**

**Example 9.1: Using Programming Statements**

This example illustrates two ways of operating on the same input variables and producing the same tabular report. To simplify the example, no report enhancements are shown.
The manager of a hotel chain wants a report that shows the number of bookings at its hotels in each of four cities, the total number of bookings in the current quarter, and the percentage of the total coming from each location for each quarter of the year. Input observations contain the following variables: REPORTDATE (report date), LA (number of bookings in Los Angeles), ATL (number of bookings in Atlanta), CH (number of bookings in Chicago), and NY (number of bookings in New York).

The following DATA step creates the SAS data set BOOKINGS:

```sas
data bookings;
  input reptdate date9. la atl ch ny;
datalines;
  01JAN1989 100 110 120 130
  01FEB1989 140 150 160 170
  01MAR1989 180 190 200 210
  01APR1989 220 230 240 250
  01MAY1989 260 270 280 290
  01JUN1989 300 310 320 330
  01JUL1989 340 350 360 370
  01AUG1989 380 390 400 410
  01SEP1989 420 430 440 450
  01OCT1989 460 470 480 490
  01NOV1989 500 510 520 530
  01DEC1989 540 550 560 570
;```

The following PROC COMPUTAB statements select columns by setting _COL_ to an appropriate value. The PCT1, PCT2, PCT3, and PCT4 columns represent the percentage contributed by each city to the total for the quarter. These statements produce Output 9.1.1.

```sas
proc computab data=bookings cspace=1 cwidth=6;
  columns qtr1 pct1 qtr2 pct2 qtr3 pct3 qtr4 pct4;
  columns qtr1-qtr4 / format=6.2;
  columns pct1-pct4 / format=6.2;
  rows la atl ch ny total;
  _col_ = qtr( reptdate ) * 2 - 1;
  pct1 = qtr1;
  pct2 = qtr2;
  pct3 = qtr3;
  pct4 = qtr4;
  total = la + atl + ch + ny;
  if mod( _col_, 2 ) = 0 then do;
    la = la / total * 100;
    atl = atl / total * 100;
    ch = ch / total * 100;
    ny = ny / total * 100;
run;
```
Example 9.1: Using Programming Statements

```
total = 100;
end;
run;
```

Output 9.1.1 Quarterly Report of Hotel Bookings

Year to Date Expenses

<table>
<thead>
<tr>
<th></th>
<th>QTR1</th>
<th>PCT1</th>
<th>QTR2</th>
<th>PCT2</th>
<th>QTR3</th>
<th>PCT3</th>
<th>QTR4</th>
<th>PCT4</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA</td>
<td>420</td>
<td>22.58</td>
<td>780</td>
<td>23.64</td>
<td>1140</td>
<td>24.05</td>
<td>1500</td>
<td>24.27</td>
</tr>
<tr>
<td>ATL</td>
<td>450</td>
<td>24.19</td>
<td>810</td>
<td>24.55</td>
<td>1170</td>
<td>24.68</td>
<td>1530</td>
<td>24.76</td>
</tr>
<tr>
<td>CH</td>
<td>480</td>
<td>25.81</td>
<td>840</td>
<td>25.45</td>
<td>1200</td>
<td>25.32</td>
<td>1560</td>
<td>25.24</td>
</tr>
<tr>
<td>NY</td>
<td>510</td>
<td>27.42</td>
<td>870</td>
<td>26.36</td>
<td>1230</td>
<td>25.95</td>
<td>1590</td>
<td>25.73</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1860</td>
<td>100.00</td>
<td>3300</td>
<td>100.00</td>
<td>4740</td>
<td>100.00</td>
<td>6180</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Using the same input data, the next set of statements shows the usefulness of arrays in allowing PROC COMPUTAB to work in two directions at once. Arrays in larger programs can both reduce the amount of program source code and simplify otherwise complex methods of referring to rows and columns. The same report as in Output 9.1.1 is produced.

```
proc computab data=bookings cspace=1 cwidth=6;
  columns qtr1 pct1 qtr2 pct2 qtr3 pct3 qtr4 pct4;
  columns qtr1-qtr4 / format=6.;
  columns pct1-pct4 / format=6.2;
  rows la atl ch ny total;
  array pct[4] pct1-pct4;
  array qt[4] qtr1-qtr4;
  array rowlist[5] la atl ch ny total;
  /* column selection */
  _col_ = qtr(reptdate) * 2 - 1;
  /* copy qtr column values temporarily into pct columns */
  colcopy:
    do i = 1 to 4;
      pct[i] = qt[i];
    end;
  /* calculate total row for all columns */
  /* calculate percentages for all rows in pct columns only */
  rowcalc:
    total = la + atl + ch + ny;
    if mod(_col_,2) = 0 then
      do i = 1 to 5;
        rowlist[i] = rowlist[i] / total * 100;
      end;
  run;
```
Example 9.2: Enhancing a Report

This example shows how a report can be enhanced from a simple listing to a complex report. The simplest COMPUTAB report is a transposed listing of the data in the SAS data set INCOMREP shown in Output 9.2.1. To produce this output, nothing is specified except the PROC COMPUTAB statement and a TITLE statement.

```sas
data incomrep;
  length type $ 8;
  input type :$8. date :monyy7. 
    sales retdis tcos selling randd 
    general admin deprec other taxes;
  format date monyy7.;
datalines;
BUDGET JAN1989 4600 300 2200 480 110 500 210 14 --8 510
BUDGET FEB1989 4700 330 2300 500 110 500 200 14 0 480
BUDGET MAR1989 4800 360 2600 500 120 600 250 15 2 520
ACTUAL JAN1989 4900 505 2100 430 130 410 200 14 --8 500
ACTUAL FEB1989 5100 480 2400 510 110 390 230 15 2 490
;
title 'Computab Report without Any Specifications';
proc computab data=incomrep;
run;
```

**Output 9.2.1** Simple Report

`Computab Report without Any Specifications`

<table>
<thead>
<tr>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
<th>COL4</th>
<th>COL5</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>BUDGET</td>
<td>BUDGET</td>
<td>BUDGET</td>
<td>ACTUAL</td>
</tr>
<tr>
<td>sales</td>
<td>4600.00</td>
<td>4700.00</td>
<td>4800.00</td>
<td>4900.00</td>
</tr>
<tr>
<td>retdis</td>
<td>300.00</td>
<td>330.00</td>
<td>360.00</td>
<td>505.00</td>
</tr>
<tr>
<td>tcos</td>
<td>2200.00</td>
<td>2300.00</td>
<td>2600.00</td>
<td>2100.00</td>
</tr>
<tr>
<td>selling</td>
<td>480.00</td>
<td>500.00</td>
<td>500.00</td>
<td>430.00</td>
</tr>
<tr>
<td>randd</td>
<td>110.00</td>
<td>110.00</td>
<td>120.00</td>
<td>130.00</td>
</tr>
<tr>
<td>general</td>
<td>500.00</td>
<td>500.00</td>
<td>600.00</td>
<td>410.00</td>
</tr>
<tr>
<td>admin</td>
<td>210.00</td>
<td>200.00</td>
<td>250.00</td>
<td>200.00</td>
</tr>
<tr>
<td>deprec</td>
<td>14.00</td>
<td>14.00</td>
<td>15.00</td>
<td>14.00</td>
</tr>
<tr>
<td>other</td>
<td>-8.00</td>
<td>0.00</td>
<td>2.00</td>
<td>-8.00</td>
</tr>
<tr>
<td>taxes</td>
<td>510.00</td>
<td>480.00</td>
<td>520.00</td>
<td>500.00</td>
</tr>
</tbody>
</table>

To exclude the budgeted values from your report, select columns for ACTUAL observations only. To remove unwanted variables, specify the variables you want in a ROWS statement.

```sas
title 'Column Selection by Month';

proc computab data=incomrep;
  rows sales--other;
  columns jana feba mara;
  mnth = month(date);
  if type = 'ACTUAL';
run;
```
Example 9.2: Enhancing a Report

```plaintext
jana = mnth = 1;
feba = mnth = 2;
mara = mnth = 3;
run;
```

The report is shown in Output 9.2.2.

**Output 9.2.2** Report That Uses Column Selection Techniques

**Column Selection by Month**

<table>
<thead>
<tr>
<th></th>
<th>JANA</th>
<th>Feba</th>
<th>MarA</th>
</tr>
</thead>
<tbody>
<tr>
<td>sales</td>
<td>4900.00</td>
<td>5100.00</td>
<td>0.00</td>
</tr>
<tr>
<td>retdis</td>
<td>505.00</td>
<td>480.00</td>
<td>0.00</td>
</tr>
<tr>
<td>tcos</td>
<td>2100.00</td>
<td>2400.00</td>
<td>0.00</td>
</tr>
<tr>
<td>selling</td>
<td>430.00</td>
<td>510.00</td>
<td>0.00</td>
</tr>
<tr>
<td>randd</td>
<td>130.00</td>
<td>110.00</td>
<td>0.00</td>
</tr>
<tr>
<td>general</td>
<td>410.00</td>
<td>390.00</td>
<td>0.00</td>
</tr>
<tr>
<td>admin</td>
<td>200.00</td>
<td>230.00</td>
<td>0.00</td>
</tr>
<tr>
<td>deprec</td>
<td>14.00</td>
<td>15.00</td>
<td>0.00</td>
</tr>
<tr>
<td>other</td>
<td>-8.00</td>
<td>2.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

To complete the report, compute new rows from existing rows. This is done in a row block (although it can also be done in the input block). Add a new column (QTR1) that accumulates all the actual data. The NOZERO option suppresses the zero column for March. The output produced by these statements is shown in Output 9.2.3.

```plaintext
proc computab data=incomrep;
/* add a new column to be selected */
/* qtr1 column will be selected several times */
columns actual1-actual3 qtr1 / nozero;
array collist[3] actual1-actual3;
rows sales retdis netsales tcos grosspft selling randd general
    admin deprec operexp operinc other taxblinc taxes netincom;
if type='ACTUAL';
i = month(date);
if i <= 3 then qtr1 = 1;
collist[i]=1;
rowcalc:
    if sales = . then return;
netsales = sales - retdis;
grosspft = netsales - tcos;
operexp = selling + randd + general + admin + deprec;
operinc = grosspft - operexp;
taxblinc = operinc + other;
netincom = taxblinc - taxes;
run;
```

run;
Output 9.2.3  Report That Uses Techniques to Compute New Rows

**Column Selection by Month**

<table>
<thead>
<tr>
<th></th>
<th>ACTUAL1</th>
<th>ACTUAL2</th>
<th>QTR1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALES</td>
<td>4900.00</td>
<td>5100.00</td>
<td>10000.00</td>
</tr>
<tr>
<td>RETDIS</td>
<td>505.00</td>
<td>480.00</td>
<td>985.00</td>
</tr>
<tr>
<td>NETSALES</td>
<td>4395.00</td>
<td>4620.00</td>
<td>9015.00</td>
</tr>
<tr>
<td>TCOS</td>
<td>2100.00</td>
<td>2400.00</td>
<td>4500.00</td>
</tr>
<tr>
<td>GROSSPFT</td>
<td>2295.00</td>
<td>2220.00</td>
<td>4515.00</td>
</tr>
<tr>
<td>SELLING</td>
<td>430.00</td>
<td>510.00</td>
<td>940.00</td>
</tr>
<tr>
<td>RANDD</td>
<td>130.00</td>
<td>110.00</td>
<td>240.00</td>
</tr>
<tr>
<td>GENERAL</td>
<td>410.00</td>
<td>390.00</td>
<td>800.00</td>
</tr>
<tr>
<td>ADMIN</td>
<td>200.00</td>
<td>230.00</td>
<td>430.00</td>
</tr>
<tr>
<td>DEPREC</td>
<td>14.00</td>
<td>15.00</td>
<td>29.00</td>
</tr>
<tr>
<td>OPEREXP</td>
<td>1184.00</td>
<td>1255.00</td>
<td>2439.00</td>
</tr>
<tr>
<td>OPERINC</td>
<td>1111.00</td>
<td>965.00</td>
<td>2076.00</td>
</tr>
<tr>
<td>OTHER</td>
<td>-8.00</td>
<td>2.00</td>
<td>-6.00</td>
</tr>
<tr>
<td>TAXBLINC</td>
<td>1103.00</td>
<td>967.00</td>
<td>2070.00</td>
</tr>
<tr>
<td>TAXES</td>
<td>500.00</td>
<td>490.00</td>
<td>990.00</td>
</tr>
<tr>
<td>NETINCOM</td>
<td>603.00</td>
<td>477.00</td>
<td>1080.00</td>
</tr>
</tbody>
</table>

Now that you have all the numbers calculated, add specifications to improve the report’s appearance. Specify titles, row and column labels, and formats. The report produced by these statements is shown in Output 9.2.4.

```plaintext
/* now get the report to look the way you want it */
title 'Pro Forma Income Statement';
title2 'XYZ Computer Services, Inc.';
title3 'Period to Date Actual';
title4 'Amounts in Thousands';
proc computab data=incomrep;
  columns actual1-actual3 qtr1 /
    nozero f=comma7. +3 ' ';
  array collist[3] actual1-actual3;
  columns actual1 / 'Jan';
  columns actual2 / 'Feb';
  columns actual3 / 'Mar';
  columns qtr1 / 'Total' 'Qtr 1';
  rows sales / ' ' 'Gross Sales';
  rows retdis / 'Less Returns & Discounts';
  rows netsales / 'Net Sales' +3 ol;
  rows tcos / ' ' 'Total Cost of Sales';
  rows grosspft / ' ' 'Gross Profit';
  rows selling / ' ' 'Operating Expenses:'
    ' Selling';
  rows randd / ' R & D';
  rows general / +3;
```
Example 9.2: Enhancing a Report

```plaintext
rows admin / ' Administrative';
rows deprec / ' Depreciation' ul;
rows operexp / ' ' skip;
rows operinc / 'Operating Income';
rows other / 'Other Income/-Expense' ul;
rows taxblinc / 'Taxable Income';
rows taxes / 'Income Taxes' ul;
rows netincom / ' Net Income' dul;

if type = 'ACTUAL';
i = month( date );
collist[i] = 1;

colcalc:
    qtr1 = actual1 + actual2 + actual3;

rowcalc:
    if sales = . then return;
    netsales = sales - retdis;
    grosspft = netsales - tcos;
    operexp = selling + randd + general + admin + deprec;
    operinc = grosspft - operexp;
    taxblinc = operinc + other;
    netincom = taxblinc - taxes;

run;
```
### Output 9.2.4 Specifying Titles, Row and Column Labels, and Formats

#### Pro Forma Income Statement
**XYZ Computer Services, Inc.**  
**Period to Date Actual**  
**Amounts in Thousands**

<table>
<thead>
<tr>
<th></th>
<th>Jan</th>
<th>Feb</th>
<th>Otr 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Gross Sales</strong></td>
<td>4,900</td>
<td>5,100</td>
<td>10,000</td>
</tr>
<tr>
<td><strong>Less Returns &amp; Discounts</strong></td>
<td>505</td>
<td>480</td>
<td>985</td>
</tr>
<tr>
<td><strong>Net Sales</strong></td>
<td>4,395</td>
<td>4,620</td>
<td>9,015</td>
</tr>
<tr>
<td><strong>Total Cost of Sales</strong></td>
<td>2,100</td>
<td>2,400</td>
<td>4,500</td>
</tr>
<tr>
<td><strong>Gross Profit</strong></td>
<td>2,295</td>
<td>2,220</td>
<td>4,515</td>
</tr>
</tbody>
</table>

**Operating Expenses:**

<table>
<thead>
<tr>
<th></th>
<th>Jan</th>
<th>Feb</th>
<th>Otr 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Selling</strong></td>
<td>430</td>
<td>510</td>
<td>940</td>
</tr>
<tr>
<td><strong>R &amp; D</strong></td>
<td>130</td>
<td>110</td>
<td>240</td>
</tr>
<tr>
<td><strong>GENERAL</strong></td>
<td>410</td>
<td>390</td>
<td>800</td>
</tr>
<tr>
<td><strong>Administrative</strong></td>
<td>200</td>
<td>230</td>
<td>430</td>
</tr>
<tr>
<td><strong>Depreciation</strong></td>
<td>14</td>
<td>15</td>
<td>29</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>1,184</td>
<td>1,255</td>
<td>2,439</td>
</tr>
</tbody>
</table>

**Operating Income**

<table>
<thead>
<tr>
<th></th>
<th>Jan</th>
<th>Feb</th>
<th>Otr 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operating Income</strong></td>
<td>1,111</td>
<td>965</td>
<td>2,076</td>
</tr>
<tr>
<td><strong>Other Income/-Expense</strong></td>
<td>-8</td>
<td>2</td>
<td>-6</td>
</tr>
<tr>
<td><strong>Taxable Income</strong></td>
<td>1,103</td>
<td>967</td>
<td>2,070</td>
</tr>
<tr>
<td><strong>Income Taxes</strong></td>
<td>500</td>
<td>490</td>
<td>990</td>
</tr>
<tr>
<td><strong>Net Income</strong></td>
<td>603</td>
<td>477</td>
<td>1,080</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---
Example 9.3: Comparison of Actual and Budget

This example shows a more complex report that compares the actual data with the budgeted values. The same input data as in the previous example is used.

The report produced by these statements is shown in Output 9.3.1. The report shows the values for the current month and the year-to-date totals for budgeted amounts, actual amounts, and the actuals as a percentage of the budgeted amounts. The data have the values for January and February. Therefore, the CURMO variable (current month) in the RETAIN statement is set to 2. The values for the observations where the month of the year is 2 (February) are accumulated for the current month values. The year-to-date values are accumulated from those observations where the month of the year is less than or equal to 2 (January and February).

```
data incomrep;
  length type $ 8;
  input type :$8. date :monyy7. sales retdis tcos selling randd
          general admin deprec other taxes;
  format date monyy7.;
datalines;
BUDGET JAN1989 4600 300 2200 480 110 500 210 14 -8 510
BUDGET FEB1989 4700 330 2300 500 110 500 200 14 0 480
BUDGET MAR1989 4800 360 2600 500 120 600 250 15 2 520
ACTUAL JAN1989 4900 505 2100 430 130 410 200 14 -8 500
ACTUAL FEB1989 5100 480 2400 510 110 390 230 15 2 490;

title 'Pro Forma Income Statement';
title2 'XYZ Computer Services, Inc.';
title3 'Budget Analysis';
title4 'Amounts in Thousands';
options linesize=96;
proc computab data=incomrep;
  columns cmbud cmact cmpct ytdbud ytdact ytdpct /
    zero=' ';
  columns cmbud--cmpct / mtitle='- Current Month: February -';
  columns ytdbud--ytdpct / mtitle='- Year To Date -';
  columns cmbud ytdbud / 'Budget' f=comma6.;
  columns cmact ytdact / 'Actual' f=comma6.;
  columns cmpct ytdpct / '% ' f=7.2;
  columns.cmbud--ytdpct / _titles_;
  retain curmo 2; /* current month: February */
  rows sales    / ''
    'Gross Sales';
  rows retdis   / 'Less Returns & Discounts';
  rows netsales / 'Net Sales' +3 ol;
  rows tcos     / ''
    'Total Cost of Sales';
  rows grosspft / ''
    'Gross Profit' +3;
```

---

Example 9.3: Comparison of Actual and Budget

This example shows a more complex report that compares the actual data with the budgeted values. The same input data as in the previous example is used.

The report produced by these statements is shown in Output 9.3.1. The report shows the values for the current month and the year-to-date totals for budgeted amounts, actual amounts, and the actuals as a percentage of the budgeted amounts. The data have the values for January and February. Therefore, the CURMO variable (current month) in the RETAIN statement is set to 2. The values for the observations where the month of the year is 2 (February) are accumulated for the current month values. The year-to-date values are accumulated from those observations where the month of the year is less than or equal to 2 (January and February).

```
data incomrep;
  length type $ 8;
  input type :$8. date :monyy7.
    sales retdis tcos selling randd
          general admin deprec other taxes;
  format date monyy7.;
datalines;
BUDGET JAN1989 4600 300 2200 480 110 500 210 14 -8 510
BUDGET FEB1989 4700 330 2300 500 110 500 200 14 0 480
BUDGET MAR1989 4800 360 2600 500 120 600 250 15 2 520
ACTUAL JAN1989 4900 505 2100 430 130 410 200 14 -8 500
ACTUAL FEB1989 5100 480 2400 510 110 390 230 15 2 490;

title 'Pro Forma Income Statement';
title2 'XYZ Computer Services, Inc.';
title3 'Budget Analysis';
title4 'Amounts in Thousands';
options linesize=96;
proc computab data=incomrep;
  columns cmbud cmact cmpct ytdbud ytdact ytdpct /
    zero=' ';
  columns cmbud--cmpct / mtitle='- Current Month: February -';
  columns ytdbud--ytdpct / mtitle='- Year To Date -';
  columns cmbud ytdbud / 'Budget' f=comma6.;
  columns cmact ytdact / 'Actual' f=comma6.;
  columns cmpct ytdpct / '% ' f=7.2;
  columns cmbud--ytdpct / _titles_;
  retain curmo 2; /* current month: February */
  rows sales    / ''
    'Gross Sales';
  rows retdis   / 'Less Returns & Discounts';
  rows netsales / 'Net Sales' +3 ol;
  rows tcos     / ''
    'Total Cost of Sales';
  rows grosspft / ''
    'Gross Profit' +3;
```
rows selling /' '
    'Operating Expenses:
    ' Selling';
rows randd /' R & D';
rows general /'+3';
rows admin /' Administrative';
rows deprec /' Depreciation' ul;
rows operexp /' ';
rows operinc /'Operating Income' ol;
rows other /'Other Income/-Expense' ul;
rows taxblinc /'Taxable Income';
rows taxes /'Income Taxes' ul;
rows netincom /' Net Income' dul;

cmbud = type = 'BUDGET' & month(date) = curmo;
cmact = type = 'ACTUAL' & month(date) = curmo;
ytdbud = type = 'BUDGET' & month(date) <= curmo;
ytdact = type = 'ACTUAL' & month(date) <= curmo;

calc:
    if cmpct | ytdpct then return;
    netsales = sales - retdis;
    grosspft = netsales - tcos;
    operexp = selling + randd + general + admin + deprec;
    operinc = grosspft - operexp;
    taxblinc = operinc + other;
    netincom = taxblinc - taxes;

colpct:
    if cmbud & cmact then cmpct = 100 * cmact / cmbud;
    if ytdbud & ytdact then ytdpct = 100 * ytdact / ytdbud;
run;
### Example 9.3: Comparison of Actual and Budget

#### Output 9.3.1 Report That Uses Specifications to Tailor Output

**Pro Forma Income Statement**

*XYZ Computer Services, Inc.*

**Budget Analysis**

**Amounts in Thousands**

<table>
<thead>
<tr>
<th></th>
<th>Current Month: February</th>
<th>Year To Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Budget</td>
<td>Actual</td>
<td>%</td>
</tr>
<tr>
<td>4,700</td>
<td>5,100</td>
<td>108.51</td>
</tr>
<tr>
<td>330</td>
<td>480</td>
<td>145.45</td>
</tr>
<tr>
<td></td>
<td>4,370</td>
<td>105.72</td>
</tr>
<tr>
<td>2,300</td>
<td>2,400</td>
<td>104.35</td>
</tr>
<tr>
<td>2,070</td>
<td>2,220</td>
<td>107.25</td>
</tr>
</tbody>
</table>

**Operating Expenses:**

<table>
<thead>
<tr>
<th></th>
<th>Selling</th>
<th>R &amp; D</th>
<th>GENERAL</th>
<th>Administrative</th>
<th>Depreciation</th>
<th>Operating Income</th>
<th>Other Income/Expense</th>
<th>Taxable Income</th>
<th>Income Taxes</th>
<th>Net Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>510</td>
<td>110</td>
<td>390</td>
<td>230</td>
<td>15</td>
<td>746</td>
<td>1,532</td>
<td>1,524</td>
<td>990</td>
<td>534</td>
</tr>
<tr>
<td></td>
<td>980</td>
<td>220</td>
<td>1,000</td>
<td>410</td>
<td>28</td>
<td>2,638</td>
<td>-8</td>
<td>2,076</td>
<td>900</td>
<td>1,080</td>
</tr>
<tr>
<td>2,638</td>
<td>2,439</td>
<td>92.46</td>
<td>75.00</td>
<td>135.51</td>
<td>135.83</td>
<td>135.00</td>
<td></td>
<td></td>
<td>202.25</td>
<td></td>
</tr>
</tbody>
</table>

---
Example 9.4: Consolidations

This example consolidates product tables by region and region tables by corporate division. Output 9.4.1 shows the North Central and Northeast regional summaries for the Equipment division for the first quarter. Output 9.4.2 shows the profit summary for the Equipment division. Similar tables for the Publishing division are produced but not shown here.

```sas
data product;
  input pcode div region month sold revenue recd cost;
datalines;
  1 1 1 1 56 5600 29 2465
  1 1 2 13 1300 30 2550
  1 1 3 17 1700 65 5525
  2 1 1 2 240 50 4900
  2 1 2 82 9840 17 1666
  ... more lines ...
  1 1 1 1 37 3700 75 6375
  ...

proc format;
  value divfmt 1='Equipment'
                 2='Publishing';
  value regfmt 1='North Central'
                 2='Northeast'
                 3='South'
                 4='West';
run;

proc sort data=product;
  by div region pcode;
run;

proc computab data=product sumonly;
  by div region pcode;
  sumby _total_ div region;
  format div divfmt.;
  format region regfmt.;
  label div = 'DIVISION';
  columns jan feb mar qtr1 /
    mtitle='- first quarter -' ' ' nozero;
  columns apr may jun qtr2 /
    mtitle='- second quarter -' ' ' nozero;
```

Example 9.4: Consolidations

```plaintext
/* specify order of rows and row titles */
row sold / ' ' 'Number Sold' f=8.;
row revenue / ' ' 'Sales Revenue';
row recd / ' ' 'Number Received' f=8.;
row cost / ' ' 'Cost of Items Received';
row profit / ' ' 'Profit' 'Within Period' ol;
row pctmarg / ' ' 'Profit Margin' dul;

/* select column for appropriate month */
_col_ = month + ceil( month / 3 ) - 1;

/* calculate quarterly summary columns */
colcalc:
  qtr1 = jan + feb + mar;
  qtr2 = apr + may + jun;
  qtr3 = jul + aug + sep;
  qtr4 = oct + nov + dec;

/* calculate profit rows */
rowcalc:
  profit = revenue - cost;
  if cost > 0 then pctmarg = profit / cost * 100;
run;
```
**Output 9.4.1** Summary by Regions for the Equipment Division

**XYZ Development Corporation**  
**Corporate Headquarters: New York, NY**  
**Profit Summary**

```
DIVISION=Equipment region=Northeast pcode=1
```

Summary Table: DIVISION=Equipment region=North Central

```
<table>
<thead>
<tr>
<th>January</th>
<th>February</th>
<th>March</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Sold</td>
<td>198</td>
<td>223</td>
<td>119</td>
</tr>
<tr>
<td>Sales Revenue</td>
<td>22090.00</td>
<td>26830.00</td>
<td>14020.00</td>
</tr>
<tr>
<td>Number Received</td>
<td>255</td>
<td>217</td>
<td>210</td>
</tr>
<tr>
<td>Cost of Items Received</td>
<td>24368.00</td>
<td>20104.00</td>
<td>19405.00</td>
</tr>
<tr>
<td>Profit Within Period</td>
<td>-2278.00</td>
<td>6726.00</td>
<td>-5385.00</td>
</tr>
<tr>
<td>Profit Margin</td>
<td>-9.35</td>
<td>33.46</td>
<td>-27.75</td>
</tr>
</tbody>
</table>
```
## Output 9.4.1 continued

**XYZ Development Corporation**  
**Corporate Headquarters: New York, NY**  
**Profit Summary**

DIVISION=PUBLISHING region=North Central pcode=4

<table>
<thead>
<tr>
<th></th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number Sold</strong></td>
<td>82</td>
<td>180</td>
<td>183</td>
<td>445</td>
</tr>
<tr>
<td><strong>Sales Revenue</strong></td>
<td>9860.00</td>
<td>21330.00</td>
<td>21060.00</td>
<td>52250.00</td>
</tr>
<tr>
<td><strong>Number Received</strong></td>
<td>162</td>
<td>67</td>
<td>124</td>
<td>353</td>
</tr>
<tr>
<td><strong>Cost of Items Received</strong></td>
<td>16374.00</td>
<td>6325.00</td>
<td>12333.00</td>
<td>35032.00</td>
</tr>
<tr>
<td><strong>Profit Within Period</strong></td>
<td>-6514.00</td>
<td>15005.00</td>
<td>8727.00</td>
<td>17218.00</td>
</tr>
<tr>
<td><strong>Profit Margin</strong></td>
<td>-39.78</td>
<td>237.23</td>
<td>70.76</td>
<td>49.15</td>
</tr>
</tbody>
</table>

---

---
Output 9.4.2  Profit Summary for the Equipment Division

XYZ Development Corporation
Corporate Headquarters: New York, NY
Profit Summary

DIVISION=Publishing region=North Central pcode=4

---------------------------------- SUMMARY TABLE: DIVISION=Equipment ----------------------------------

<table>
<thead>
<tr>
<th>January</th>
<th>February</th>
<th>March</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Sold</td>
<td>280</td>
<td>403</td>
<td>302</td>
</tr>
<tr>
<td>Sales Revenue</td>
<td>31950.00</td>
<td>48160.00</td>
<td>35080.00</td>
</tr>
<tr>
<td>Number Received</td>
<td>417</td>
<td>284</td>
<td>334</td>
</tr>
<tr>
<td>Cost of Items Received</td>
<td>40742.00</td>
<td>26429.00</td>
<td>31738.00</td>
</tr>
<tr>
<td>Profit Within Period</td>
<td>-8792.00</td>
<td>21731.00</td>
<td>3342.00</td>
</tr>
<tr>
<td>Profit Margin</td>
<td>-21.58</td>
<td>82.22</td>
<td>10.53</td>
</tr>
</tbody>
</table>
Output 9.4.3 shows the consolidation report of profit summary over both divisions and regions.

**Output 9.4.3  Profit Summary**

**XYZ Development Corporation**
**Corporate Headquarters: New York, NY**
**Profit Summary**

--- SUMMARY TABLE: TOTALS ---

---- first quarter ----

<table>
<thead>
<tr>
<th></th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Sold</td>
<td>590</td>
<td>683</td>
<td>627</td>
<td>1900</td>
</tr>
<tr>
<td>Sales Revenue</td>
<td>41790.00</td>
<td>55910.00</td>
<td>44800.00</td>
<td>142500.00</td>
</tr>
<tr>
<td>Number Received</td>
<td>656</td>
<td>673</td>
<td>734</td>
<td>2063</td>
</tr>
<tr>
<td>Cost of Items</td>
<td>46360.00</td>
<td>35359.00</td>
<td>40124.00</td>
<td>121843.00</td>
</tr>
<tr>
<td>Profit Within Period</td>
<td>-4570.00</td>
<td>20551.00</td>
<td>4676.00</td>
<td>20657.00</td>
</tr>
<tr>
<td>Profit Margin</td>
<td>-9.86</td>
<td>58.12</td>
<td>11.65</td>
<td>16.95</td>
</tr>
</tbody>
</table>

---

**Example 9.5: Creating an Output Data Set**

This example uses data and reports similar to those in Example 9.3 to illustrate the creation of an output data set.

```
data product;
  input pcode div region month sold revenue recd cost;
datalines;
  1 1 1 1 56 5600 29 2465
  1 1 1 2 13 1300 30 2550
  1 1 1 3 17 1700 65 5525
  2 1 1 1 2 240 50 4900
  2 1 1 2 82 9840 17 1666
  1 1 1 1 37 3700 75 6375
  ... more lines ...
```
proc sort data=product out=sorted;
   by div region;
run;

/* create data set, profit */
proc computab data=sorted notrans out=profit noprint;
   by div region;
   sumby div;

/* specify order of rows and row titles */
row   jan  feb  mar  qtr1;
row   apr  may  jun  qtr2;
row   jul  aug  sep  qtr3;
row   oct  nov  dec  qtr4;

/* specify order of columns and column titles */
columns sold revenue recd cost profit pctmarg;

/* select row for appropriate month */
_row_ = month + ceil( month / 3 ) - 1;

/* calculate quarterly summary rows */
rowcalc:
   qtr1 = jan + feb + mar;
   qtr2 = apr + may + jun;
   qtr3 = jul + aug + sep;
   qtr4 = oct + nov + dec;

/* calculate profit columns */
colcalc:
   profit = revenue - cost;
   if cost > 0 then pctmarg = profit / cost * 100;
run;

/* make a partial listing of the output data set */
options linesize=96;
proc print data=profit(obs=10) noobs;
run;

Because the NOTRANS option is specified, column names become variables in the data set. REGION has missing values in the output data set for observations associated with consolidation tables. The output data set PROFIT, in conjunction with the option NOPRINT, illustrates how you can use the computational features of PROC COMPUTAB for creating additional rows and columns as in a spreadsheet without producing a report. Output 9.5.1 shows a partial listing of the output data set PROFIT.
Example 9.6: A What-If Market Analysis

PROC COMPUTAB can be used with other SAS/ETS procedures and with macros to implement commonly needed decision support tools for financial and marketing analysis.

The following input data set reads quarterly sales figures:

```sas
data market;
  input date :yyq6. units @@;
datalines;
1980Q1 3608.9 1980Q2 5638.4 1980Q3 6017.9 1980Q4 4929.6
... more lines ...
```

The following statements illustrate how PROC FORECAST makes a total market forecast for the next four quarters:

```sas
/* forecast the total number of units to be */
/* sold in the next four quarters */
proc forecast out=outcome trend=2
  id date;
  var units;
  interval=qtr lead=4;
run;
```

The macros WHATIF and SHOW build a report table and provide the flexibility of examining alternate what-if situations. The row and column calculations of PROC COMPUTAB compute the income statement. With macros stored in a macro library, the only statements required with PROC COMPUTAB are macro invocations and TITLE statements.
/* set up rows and columns of report and initialize */
/* market share and program constants */
%macro whatif(mktshr=,price=,ucost=,taxrate=,numshar=,overhead=);

columns mar / ' ' 'March';
columns jun / ' ' 'June';
columns sep / ' ' 'September';
columns dec / ' ' 'December';
columns total / 'Calculated' 'Total';
rows mktshr / 'Market Share' f=5.2;
rows tunits / 'Market Forecast';
rows units / 'Items Sold';
rows sales / 'Sales';
rows cost / 'Cost of Goods';
rows ovhd / 'Overhead';
rows gprof / 'Gross Profit';
rows tax / 'Tax';
rows pat / 'Profit After Tax';
rows earn / 'Earnings per Share';

rows mktshr--earn / skip;
rows sales--earn / f=dollar12.2;
rows tunits units / f=comma12.2;

/* initialize market share values */
init mktshr &mktshr;

/* define constants */
retain price &price ucost &ucost taxrate &taxrate numshar &numshar;

/* retain overhead and sales from previous quarter */
retain prevovhd &overhead prevsale;
%mend whatif;

/* perform calculations and print the specified rows */
%macro show(rows);

/* initialize list of row names */
%let row1 = mktshr;
%let row2 = tunits;
%let row3 = units;
%let row4 = sales;
%let row5 = cost;
%let row6 = ovhd;
%let row7 = gprof;
%let row8 = tax;
%let row9 = pat;
%let row10 = earn;

/* find parameter row names in list and eliminate */
/* them from the list of noprint rows */
%let n = 1;
Example 9.6: A What-If Market Analysis

%let word = %scan(&rows,&n);
%do %while(&word NE );
   %let i = 1;
   %let row1 = &word;
%do %while(&row&i NE &word);
   %let i = %eval(&i+1);
%end;
%if &i<11 %then %let row&i = ;
%let n = %eval(&n+1);
%let word = %scan(&rows,&n);
%end;

rows &row1 &row2 &row3 &row4 &row5 &row6 &row7 &row8 &row9 &row10 dummy / noprint;

/* select column using lead values from proc forecast */
mar = _lead_ = 1;
jun = _lead_ = 2;
sep = _lead_ = 3;
dec = _lead_ = 4;

rowreln:;
    /* inter-relationships */
    share = round( mktshr, 0.01 );
    tunits = units;
    units = share * tunits;
    sales = units * price;
    cost = units * ucost;

    /* calculate overhead */
    if mar then prevsale = sales;
    if sales > prevsale
        then ovhd = prevovhd + .05 * ( sales - prevsale );
        else ovhd = prevovhd;
    prevovhd = ovhd;
    prevsale = sales;
    gprof = sales - cost - ovhd;
    tax = gprof * taxrate;
    pat = gprof - tax;
    earn = pat / numshar;

coltot:;
    if mktshr
        then total = ( mar + jun + sep + dec ) / 4;
        else total = mar + jun + sep + dec;
%mend show;
run;

The following PROC COMPUTAB statements use the PROC FORECAST output data set with invocations of the macros defined previously to perform a what-if analysis of the predicted income statement. The report is shown in Output 9.6.1.
Chapter 9: The COMPUTAB Procedure

* Fleet Footwear, Inc. *
* Marketing Analysis Income Statement *
* Based on Forecasted Unit Sales *
* All Values Shown *

**title1** 'Fleet Footwear, Inc.';
**title2** 'Marketing Analysis Income Statement';
**title3** 'Based on Forecasted Unit Sales';
**title4** 'All Values Shown';

**options** linesize=96;

**proc computab data=outcome cwidth=12;**

%**whatif**(mktshr=.02 .07 .15 .25,price=38.00,
ucost=20.00,taxrate=.48,numshar=15000,overhead=5000);

%**show**(mktshr tunits units sales cost ovhd gprof tax pat earn);
**run;**

**Output 9.6.1** PROC COMPUTAB Report That Uses Macro Invocations

**Fleet Footwear, Inc.**

**Marketing Analysis Income Statement**
**Based on Forecasted Unit Sales**
**All Values Shown**

<table>
<thead>
<tr>
<th>Market Share</th>
<th>March</th>
<th>June</th>
<th>September</th>
<th>December</th>
<th>Calculated Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.02</td>
<td>0.07</td>
<td>0.15</td>
<td>0.25</td>
<td>0.12</td>
</tr>
<tr>
<td>Market Forecast</td>
<td>23,663.94</td>
<td>24,169.61</td>
<td>24,675.27</td>
<td>25,180.93</td>
<td>97,689.75</td>
</tr>
<tr>
<td>Items Sold</td>
<td>473.28</td>
<td>1,691.87</td>
<td>3,701.29</td>
<td>6,295.23</td>
<td>12,161.67</td>
</tr>
<tr>
<td>Sales</td>
<td>$17,984.60</td>
<td>$64,291.15</td>
<td>$140,649.03</td>
<td>$239,218.83</td>
<td>$462,143.61</td>
</tr>
<tr>
<td>Cost of Goods</td>
<td>$9,465.58</td>
<td>$33,837.45</td>
<td>$74,025.80</td>
<td>$125,904.65</td>
<td>$243,233.48</td>
</tr>
<tr>
<td>Overhead</td>
<td>$5,000.00</td>
<td>$7,315.33</td>
<td>$11,133.22</td>
<td>$16,061.71</td>
<td>$39,510.26</td>
</tr>
<tr>
<td>Gross Profit</td>
<td>$3,519.02</td>
<td>$23,138.38</td>
<td>$55,490.00</td>
<td>$97,252.47</td>
<td>$179,399.87</td>
</tr>
<tr>
<td>Tax</td>
<td>$1,689.13</td>
<td>$11,106.42</td>
<td>$26,635.20</td>
<td>$46,681.19</td>
<td>$86,111.94</td>
</tr>
<tr>
<td>Profit After Tax</td>
<td>$1,829.89</td>
<td>$12,031.96</td>
<td>$28,854.80</td>
<td>$50,571.28</td>
<td>$93,287.93</td>
</tr>
<tr>
<td>Earnings per Share</td>
<td>$0.12</td>
<td>$0.80</td>
<td>$1.92</td>
<td>$3.37</td>
<td>$6.22</td>
</tr>
</tbody>
</table>
The following statements produce a similar report for different values of market share and unit costs. The report in Output 9.6.2 displays the values for the market share, market forecast, sales, after-tax profit, and earnings per share.

```
title3 'Revised';
title4 'Selected Values Shown';

options linesize=96;

proc computab data=outcome cwidth=12;
   %whatif(mktshr=.01 .06 .12 .20,price=38.00,
          ucost=23.00,taxrate=.48,numshar=15000,overhead=5000);
   %show(mktshr tunits sales pat earn);
run;
```

**Output 9.6.2** Report That Uses Macro Invocations for Selected Values

| Fleet Footwear, Inc. | Marketing Analysis Income Statement | Revised | Selected Values Shown |
|----------------------|------------------------------------|---------|
|                      | Calculated                         |         |

<table>
<thead>
<tr>
<th></th>
<th>March</th>
<th>June</th>
<th>September</th>
<th>December</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Market Share</td>
<td>0.01</td>
<td>0.06</td>
<td>0.12</td>
<td>0.20</td>
<td>0.10</td>
</tr>
<tr>
<td>Market Forecast</td>
<td>23,663.94</td>
<td>24,169.61</td>
<td>24,675.27</td>
<td>25,180.93</td>
<td>97,689.75</td>
</tr>
<tr>
<td>Sales</td>
<td>$8,992.30</td>
<td>$55,106.70</td>
<td>$112,519.22</td>
<td>$191,375.06</td>
<td>$367,993.28</td>
</tr>
<tr>
<td>Profit After Tax</td>
<td>-$754.21</td>
<td>$7,512.40</td>
<td>$17,804.35</td>
<td>$31,940.30</td>
<td>$56,502.84</td>
</tr>
<tr>
<td>Earnings per Share</td>
<td>-$0.05</td>
<td>$0.50</td>
<td>$1.19</td>
<td>$2.13</td>
<td>$3.77</td>
</tr>
</tbody>
</table>
Example 9.7: Cash Flows

The COMPUTAB procedure can be used to model cash flows from one time period to the next. The RETAIN statement is useful for enabling a row or column to contribute one of its values to its successor. Financial functions such as IRR (internal rate of return) and NPV (net present value) can be used on PROC COMPUTAB table values to provide a more comprehensive report. The following statements produce Output 9.7.1:

data cashflow;
  input date date9. netinc depr borrow invest tax div adv;
  datalines;
  30MAR1982 65 42 32 126 43 51 41
  30JUN1982 68 47 32 144 45 54 46
  30SEP1982 70 49 30 148 46 55 47
  30DEC1982 73 49 30 148 48 55 47;
  title1 'Blue Sky Endeavors';
  title2 'Financial Summary';
  title4 '(Dollar Figures in Thousands)';
  proc computab data=cashflow;
    cols qtr1 qtr2 qtr3 qtr4 / 'Quarter' f=7.1;
    col qtr1 / 'One';
    col qtr2 / 'Two';
    col qtr3 / 'Three';
    col qtr4 / 'Four';
    row begcash / 'Beginning Cash';
    row netinc / 'Income' ' Net income';
    row depr / 'Depreciation';
    row borrow;
    row subtot1 / 'Subtotal';
    row invest / 'Expenditures' ' Investment';
    row tax / 'Taxes';
    row div / 'Dividend';
    row adv / 'Advertising';
    row subtot2 / 'Subtotal';
    row cashflow/ skip;
    row irret / 'Internal Rate' ' of Return' zero='';
    rows depr borrow subtot1 tax div adv subtot2 / +3;
    retain cashin -5;
    _col_ = qtr( date );
    rowblock:
      subtot1 = netinc + depr + borrow;
      subtot2 = tax + div + adv;
      begcash = cashin;
      cashflow = begcash + subtot1 - subtot2;
      irret = cashflow;
      cashin = cashflow;
"
Example 9.7: Cash Flows

```
colblock:
   if begcash then cashin = qtr1;
   if irret then do;
      temp = irr( 4, cashin, qtr1, qtr2, qtr3, qtr4 );
      qtr1 = temp;
      qtr2 = 0; qtr3 = 0; qtr4 = 0;
   end;
run;
```

**Output 9.7.1** Report That Uses a RETAIN Statement and the IRR Financial Function

**Blue Sky Endeavors**

**Financial Summary**

(Dollar Figures in Thousands)

<table>
<thead>
<tr>
<th></th>
<th>Quarter One</th>
<th>Quarter Two</th>
<th>Quarter Three</th>
<th>Quarter Four</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beginning Cash</td>
<td>-5.0</td>
<td>-1.0</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Income</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net income</td>
<td>65.0</td>
<td>68.0</td>
<td>70.0</td>
<td>73.0</td>
</tr>
<tr>
<td>Depreciation</td>
<td>42.0</td>
<td>47.0</td>
<td>49.0</td>
<td>49.0</td>
</tr>
<tr>
<td>BORROW</td>
<td>32.0</td>
<td>32.0</td>
<td>30.0</td>
<td>30.0</td>
</tr>
<tr>
<td>Subtotal</td>
<td>139.0</td>
<td>147.0</td>
<td>149.0</td>
<td>152.0</td>
</tr>
<tr>
<td>Expenditures</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Investment</td>
<td>126.0</td>
<td>144.0</td>
<td>148.0</td>
<td>148.0</td>
</tr>
<tr>
<td>Taxes</td>
<td>43.0</td>
<td>45.0</td>
<td>46.0</td>
<td>48.0</td>
</tr>
<tr>
<td>Dividend</td>
<td>51.0</td>
<td>54.0</td>
<td>55.0</td>
<td>55.0</td>
</tr>
<tr>
<td>Advertising</td>
<td>41.0</td>
<td>46.0</td>
<td>47.0</td>
<td>47.0</td>
</tr>
<tr>
<td>Subtotal</td>
<td>135.0</td>
<td>145.0</td>
<td>148.0</td>
<td>150.0</td>
</tr>
<tr>
<td>CASHFLOW</td>
<td>-1.0</td>
<td>1.0</td>
<td>2.0</td>
<td>4.0</td>
</tr>
</tbody>
</table>

Internal Rate of Return 20.9
Subject Index

CDT (COMPUTAB data table)
  COMPUTAB procedure, 486

column blocks
  COMPUTAB procedure, 487

column selection
  COMPUTAB procedure, 484, 485

COLxxxxx: label
  COMPUTAB procedure, 479

COMPUTAB procedure
  CDT (COMPUTAB data table), 486
  column blocks, 487
  column selection, 484, 485
  COLxxxxx: label, 479
  consolidation tables, 478
  controlling row and column block execution, 485
  input block, 486
  missing values, 488
  order of calculations, 482
  output data sets, 488
  program flow, 480
  programming statements, 479
  reserved words, 488
  row blocks, 487
  ROWxxxxx: label, 479
  table cells, direct access to, 487
  consolidation tables
    COMPUTAB procedure, 478
  controlling row and column block execution
    COMPUTAB procedure, 485

input block
  COMPUTAB procedure, 486

missing values
  COMPUTAB procedure, 488

order of calculations
  COMPUTAB procedure, 482

output data sets
  COMPUTAB procedure, 488

program flow
  COMPUTAB procedure, 480

programming statements
  COMPUTAB procedure, 479

reserved words
  COMPUTAB procedure, 488

row blocks
Syntax Index

BY statement
  COMPUTAB procedure, 473
CDEC= option
  PROC COMPUTAB statement, 472
CELL statement
  COMPUTAB procedure, 473
CHAR option
  COLUMNS statement (COMPUTAB), 474
  ROWS statement (COMPUTAB), 476
‘column headings’ option
  COLUMNS statement (COMPUTAB), 474
COLUMNS statement
  COMPUTAB procedure, 473
  syntax, 469
CSPACE= option
  PROC COMPUTAB statement, 472
CWIDTH= option
  PROC COMPUTAB statement, 472
DATA= option
  PROC COMPUTAB statement, 472
DOL option
  ROWS statement (COMPUTAB), 477
DUL option
  ROWS statement (COMPUTAB), 477
FORMAT= option
  COLUMNS statement (COMPUTAB), 475
  ROWS statement (COMPUTAB), 478
FUZZ= option
  PROC COMPUTAB statement, 472
INIT statement
  COMPUTAB procedure, 475
INITMISS option
  PROC COMPUTAB statement, 472
_LABEL_ option
  COLUMNS statement (COMPUTAB), 474
  ROWS statement (COMPUTAB), 476
LJC option
  COLUMNS statement (COMPUTAB), 475
  ROWS statement (COMPUTAB), 478
MTITLE= option
  COLUMNS statement (COMPUTAB), 474
+n option
  COLUMNS statement (COMPUTAB), 474
  ROWS statement (COMPUTAB), 477
_NAME_ option
  COLUMNS statement (COMPUTAB), 474
  ROWS statement (COMPUTAB), 477
NOPRINT option
  COLUMNS statement (COMPUTAB), 474
  PROC COMPUTAB statement, 472
  ROWS statement (COMPUTAB), 477
NORTR option
  PROC COMPUTAB statement, 472
NOTRANS option
  PROC COMPUTAB statement, 472
NOTRANSPOSE option
  PROC COMPUTAB statement, 472
NOZERO option
  COLUMNS statement (COMPUTAB), 474
  ROWS statement (COMPUTAB), 477
OL option
  ROWS statement (COMPUTAB), 477
OPTIONS option
  PROC COMPUTAB statement, 473
OUT= option
  PROC COMPUTAB statement, 473
OVERPRINT option
  ROWS statement (COMPUTAB), 477
_PAGE_ option
  COLUMNS statement (COMPUTAB), 474
  ROWS statement (COMPUTAB), 478
PROC COMPUTAB
  NOTRANS, 489
  OUT=, 488
PROC COMPUTAB statement, 471
‘row titles’ option
  ROWS statement (COMPUTAB), 477
ROWS statement
  COMPUTAB procedure, 476
RTS= option
  PROC COMPUTAB statement, 472
SKIP option
  ROWS statement (COMPUTAB), 477
SUMBY statement
  COMPUTAB procedure, 478
SUMONLY option
  PROC COMPUTAB statement, 473
_TITLES_ option
   COLUMNS statement (COMPUTAB), 475

UL option
   ROWS statement (COMPUTAB), 477

ZERO= option
   COLUMNS statement (COMPUTAB), 475
   ROWS statement (COMPUTAB), 478