Paper 109 – 2010

Thirty Three Tricks with PROC REPORT:

A Quick Introduction to the Powerful REPORT Procedure

Ben Cochran, The Bedford Group, Raleigh, NC

Abstract

When the REPORT Procedure was first introduced by SAS with the advent of Version 6, most of the SAS world was mainframe users. This new procedure brought with it a great deal of power and flexibility that added much strength to SAS' arsenal of report generating procedures. It had powerful formatting, summarizing, and analysis capabilities that made it easier to create many different kinds of reports from a simple listing to a very complex report. However, some critics have stated that, while it has awesome features, it looks like a mainframe report. That was true until SAS released the Output Delivery System (ODS) experimentally in Version 7, and in production mode in Version 8. If the REPORT procedure was a cake, then ODS delivers the icing to generate truly beautiful reports. This paper offers a quick overview of the types of reports that can be generated with Proc REPORT, and how to add some of the ODS features to deliver stunning reports.

Introduction

This paper is intended for users who have not worked with the REPORT procedure, or at least not lately. First, a look is taken at the general syntax, and then a series of reports will be generated starting with a very simple report and then working up to more complex reports.

Typical Syntax

The REPORT procedure is made up of a PROC statement, a COLUMN statement, several DEFINE statements, and other optional statements that help with calculations and summarizations.

```
PROC REPORT  data= SAS-data-set  options ;
        COLUMNS variable_1 …. variable_n;
        DEFINE variable_1;
        DEFINE variable_2;
        . . .
        DEFINE variable_n;

        COMPUTE  blocks
        BREAK … ;
        RBREAK … ;
RUN;
```

COLUMNS statement defines which columns appear in the report, and their order.
DEFINE statements declare how variables are to be used in the report.
COMPUTE blocks allow calculations to be performed in the report.
BREAK / RBREAK statements allow summarization and some kinds of formatting at certain places in the report.

The REPORT procedure also has many options that can be used. Some of the most often used options are:

- DATA= specifies the dataset to be processed,
- PROMPT invokes the prompting mode, sort of like a wizard
- NOWINDOWS suppresses the REPORT window and directs the report to the output window
- REPORT = specifies a stored report to be used in generating a new report
- OUTREPT= names a location to store the report
- OUT= creates a SAS data set
- HEADLINE creates a horizontal line between the column headers and the body of the report
- HEADSKIP creates a blank line between the column headers and the body of the report

Now we are ready to take these statements and options and begin creating reports using the SASHELP.CLASS data set.

Trick 1: Generate a basic Report using the REPORT procedure.

```
proc report data=sashelp.class nowindows;
     columns name sex age height weight;
     define name   / display  'Name'   width=10;
     define sex    / display  'Gender' width=6;
     define age    / display  'Age'    width=4;
     define height / analysis 'Height' format=8.1;
     define weight / analysis 'Weight' format=8.1;
run;
```

Program 1.

Notice the DEFINE statements. The term following the '/' specifies the way the REPORT procedure uses the column. Columns can be defined as:

- GROUP - puts observations into categories
- DISPLAY - displays values for each observation
- ANALYSIS - contributes values to a calculation or statistic
- ORDER - defines the order of the report rows
- ACROSS - creates columns for each of its values
- COMPUTED - its values are created in a COMPUTE block.

Now, lets look at the output created from the above program.

The SAS System

Name	Gender	Age	Height	Weight
Alfred	M	14	69.0	112.5
Alice	F	13	56.5	84.0
Barbara	F	13	65.3	98.0
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83.0
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84.0
John	M	12	59.0	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90.0

Output 1 – Partial PROC REPORT output.

At first glance, this looks a little like PROC PRINT output without the OBS column. Aesthetically, the output could use some improvement, so let's enhance the report.

Trick 2: Add a Compute Block and a total row at the end of the report.

```
proc report data=sashelp.class nowindows headline headskip;
     columns name sex age height weight ratio;
     define name   / display  'Name'   width=10;
     define sex    / display  'Gender' width=6;
     define age    / display  'Age'    width=4;
     define height / analysis mean 'Height' format=8.1;
     define weight / analysis mean 'Weight' format=8.1;
     define ratio  / computed format=6.2;
     compute ratio;
        ratio = height.mean / weight.mean;
     endcompute;
     rbreak after / summarize dol dul;
run;
```

Program 2.

You can see the new column, **ratio**, as well as the averages for **Height**, **Weight** and **ratio** at the end of the report.

Name	Gender	Age	Height	Weight	ratio
Alfred	M	14	69.0	112.5	0.61
Alice	F	13	56.5	84.0	0.67
Barbara	F	13	65.3	98.0	0.67
Carol	F	14	62.8	102.5	0.61
Henry	M	14	63.5	102.5	0.62
James	M	12	57.3	83.0	0.69
Jane	F	12	59.8	84.5	0.71
Janet	F	15	62.5	112.5	0.56
Jeffrey	M	13	62.5	84.0	0.74
John	M	12	59.0	99.5	0.59
Joyce	F	11	51.3	50.5	1.02
Judy	F	14	64.3	90.0	0.71
Louise	F	12	56.3	77.0	0.73
Mary	F	15	66.5	112.0	0.59
Philip	M	16	72.0	150.0	0.48
Robert	M	12	64.8	128.0	0.51
Ronald	M	15	67.0	133.0	0.50
Thomas	M	11	57.5	85.0	0.68
William	M	15	66.5	112.0	0.59
			62.3	100.0	0.62

Output 2.

Notice the columns defined as ANALYSIS and COMPUTED have MEANS generated with the RBREAK statement. The DUL stands for double underline and DOL stands for double overline. Why is there no AVERAGE for **age** at the end of the report?

Trick 3: Generate subtotals and add a blank line after each gender.

```
proc report data=sashelp.class nowindows headline headskip;
     columns sex name age height weight ratio;
     define sex    / group    'Gender'   width=10;
     define name   / display  'Name' width=6;
     define age    / analysis mean 'Age'    width=4;
     define height / analysis mean 'Height' format=8.1;
     define weight / analysis mean 'Weight' format=8.1;
     define ratio  / computed format=6.2;
     compute ratio;
        ratio = height.mean / weight.mean;
     endcompute;
     break after sex / skip summarize dol dul;
run;
```

Program 3.

Notice the BREAK statement and its options. Also notice that the BREAK statement has been dropped.
Look at the definition of **age**; it has been changed from display to analysis. Also note the addition of the MEAN statistic.
Notice that the order of the variables have rearranged. **Sex** and **name** have been switched. **Sex** has been redefined as a GROUP variable.

The SAS System

Gender	Name	Age	Height	Weight	ratio
F	Alice	13	56.5	84.0	0.67
	Barbar	13	65.3	98.0	0.67
	Carol	14	62.8	102.5	0.61
	Jane	12	59.8	84.5	0.71
	Janet	15	62.5	112.5	0.56
	Joyce	11	51.3	50.5	1.02
	Judy	14	64.3	90.0	0.71
	Louise	12	56.3	77.0	0.73
	Mary	15	66.5	112.0	0.59
F		13.2	60.6	90.1	0.67
M	Alfred	14	69.0	112.5	0.61
	Henry	14	63.5	102.5	0.62
	James	12	57.3	83.0	0.69
	Jeffre	13	62.5	84.0	0.74
	John	12	59.0	99.5	0.59
	Philip	16	72.0	150.0	0.48
	Robert	12	64.8	128.0	0.51
	Ronald	15	67.0	133.0	0.50
	Thomas	11	57.5	85.0	0.68
	Willia	15	66.5	112.0	0.59
M		13.4	63.9	109.0	0.59

Output 3.

Trick 4: Calculate Percentages with PROC REPORT.

In the next task, calculate the percentages that each row represents for each group. In this case, the percentages are to add up to 100 for each group. One sub-task that needs to be done is to add up weight for each group, then divide each persons weight by the group total.

```
title 'Calculating Percentages with Proc Report';

proc report data=sashelp.class nowindows headline headskip;
     columns sex name height weight  weight_pct;
     define  sex     / group     'Gender' width=10;
     define  name    / display   'Name'   width=6;
     define  height / analysis mean  'Height' format=8.1;
     define  weight / analysis       'Weight' format=8.1;
     define  weight_pct /  '% of Weight' format=percent8.2;
     *-------- Calculations for each row -----------------*;
     compute weight_pct;
        weight_pct = weight.sum / weight_sum;
     endcompute;
     *----------------------------------------------------*;
     compute before sex;
        weight_sum = weight.sum;
     endcompute;
     break after sex / skip summarize dol dul;
run;
```

Program 4.

Notice the two compute blocks. The second one creates **weight_sum** which is the total weight for each group. The second compute block calculates each person's percent of the group. Also notice that the statistic has been dropped from the DEFINE statement for **weight.**

Calculating Percentages with Proc Report

Gender	Name	Height	Weight	% of Weight
F	Alice	56.5	84.0	10.36%
	Barbar	65.3	98.0	12.08%
	Carol	62.8	102.5	12.64%
	Jane	59.8	84.5	10.42%
	Janet	62.5	112.5	13.87%
	Joyce	51.3	50.5	6.23%
	Judy	64.3	90.0	11.10%
	Louise	56.3	77.0	9.49%
	Mary	66.5	112.0	13.81%
F		60.6	811.0	100.0%
M	Alfred	69.0	112.5	10.33%
	Henry	63.5	102.5	9.41%
	James	57.3	83.0	7.62%
	Jeffre	62.5	84.0	7.71%
	John	59.0	99.5	9.13%
	Philip	72.0	150.0	13.77%
	Robert	64.8	128.0	11.75%
	Ronald	67.0	133.0	12.21%
	Thomas	57.5	85.0	7.80%
	Willia	66.5	112.0	10.28%
M		63.9	1089.5	100.0%

Output 4.

Notice the **weight** column. The total weight is displayed for it, while the average is displayed for the **height** column. Is the subtotal for **weight** really realistic for this report? Maybe not, but it is needed here to calculate the percents. What we need is a way to have access to the sum of **weight**, but not display it in the report. Is there a way to do this in PROC REPORT?

Trick 5. Enhance the report by not displaying the WEIGHT column.

```
proc report data=sashelp.class nowindows headline headskip;
     columns sex name height weight  weight_pct;
     define  sex    / group    'Gender' width=10;
     define  name   / display  'Name'   width=6;
     define  height / analysis mean 'Height' format=8.1;
     define  weight / analysis noprint       format=8.1;
     define  weight_pct /  '% of Weight' format=percent8.2;
     *-------- Calculations for each row -----------------*;
     compute weight_pct;
        weight_pct = weight.sum / weight_sum;
     endcompute;
     *-----------------------------------------------------*;
```

Partial Program 5.

Notice the DEFINE statement for **weight**; there is a **NOPRINT** option that 'hides' the weight column from the output.

Gender	Name	Height	% of Weight
F	Alice	56.5	10.36%
	Barbar	65.3	12.08%
	Carol	62.8	12.64%
	Jane	59.8	10.42%
	Janet	62.5	13.87%
	Joyce	51.3	6.23%
	Judy	64.3	11.10%
	Louise	56.3	9.49%
	Mary	66.5	13.81%
F		60.6	100.0%

Partial Output 5.

Notice the absence of the **weight** column. But what if we wanted a column showing **weight**, and subtotals showing the average **weight** for each group in addition to the **weight %** column. Can this be done? What would be needed in the Proc REPORT step?

Trick 6. Add a column to display **weight** and its AVERAGE for each group.

```
proc report data=sashelp.class nowindows headline headskip;
     columns sex name height weight weight=weight2 weight_pct;
     define  sex      / group    'Gender' width=10;
     define  name     / display  'Name'   width=6;
     define  height   / analysis mean 'Height' format=8.1;
     define  weight   / analysis noprint       format=8.1;
     define  weight2  / analysis mean          format=8.1;
     define  weight_pct /  '% of Weight' format=percent8.2;
     *-------- Calculations for each row -----------------*;
     compute weight_pct;
        weight_pct = weight.sum / weight_sum;
     endcompute;
     *----------------------------------------------------*;
     compute before sex;
        weight_sum = weight.sum;
     endcompute;
     break after sex / skip summarize dol dul;
run;
```

Program 6.

Notice the **'weight = weight2'** in the COLUMNS statement. This creates an alias for **weight,** giving this report two weight columns, **weight** and **weight2**, each with its own definition. The statistic for **weight** defaults to **sum** and is used in calculating percentages, but is not displayed in the report because of the NOPRINT definition. **Weight2** is used to display the averages for each value of the **sex** variable.

Gender	Name	Height	Weight	% of Weight
F	Alice	56.5	84.0	10.36%
	Barbar	65.3	98.0	12.08%
	Carol	62.8	102.5	12.64%
	Jane	59.8	84.5	10.42%
	Janet	62.5	112.5	13.87%
	Joyce	51.3	50.5	6.23%
	Judy	64.3	90.0	11.10%
	Louise	56.3	77.0	9.49%
	Mary	66.5	112.0	13.81%
F		60.6	90.1	100.0%

Partial Output 6.

The above output display only the first half of the report.

Up to this point, the report has only shown a single statistic for each column. What if you wanted to see more than one statistic for each variable. Can this be done with PROC REPORT?

Trick 7. Calculate multiple statistics for each analysis variable.

In order to calculate multiple statistics for each column, we are going to have to manipulate the data. The trick that makes this work is to have a duplicate group column for each statistic. In other words, in this example, two statistics are displayed.

```
data prep;
    length name $16;
    set sashelp.class;
```

```
    gender = sex;
run;
```

Once the data is manipulated, it can now be processed by the REPORT procedure.

```
proc report data=prep nowindows headline headskip;
     columns sex gender name weight weight=weight_mn weight=weight_md;
     define  sex       / group     'Gender' width=6;
     define  gender    / group      noprint;
     define  name      / group     'Name'   width=16;
     define  weight    / analysis   format=8.2  ;
     define  weight_md / median noprint;
     define  weight_mn / mean   noprint;
     *-----------------------------------------------------*;
     compute after sex;
        name='Median Weight';
        weight.sum  = weight_md;
     endcompute;
     *-----------------------------------------------------*;
     compute after gender;
        name='Average Weight';
        weight.sum  = weight_mn;
     endcompute;
     *-----------------------------------------------------*;
     break after sex / skip summarize  dul ol;
     break after gender /   summarize  dol;
run;
```

Program 7.

Notice there is the **weight** column and two alias': **weight_md** and **weight_mn**. **Weight** is displayed, but **weight_mn** and **weight_md** have the NOPRINT definition.

You can only calculate one row (statistic) for each column in a compute block. You can only have one compute block for each variable (column). So, if you want two statistics for each column, then you need two compute blocks, but they have to be for different variables. Hence, in the new data set, we have the variable **sex**, and the variable **gender.** Both have the same values for each row. We have a compute block for each variable, one for **sex**, and one for **gender**. Notice the assignment statement for **name** in each. Notice the output below:

Gender	Name	Weight
F	Alice	84.00
	Barbara	98.00
	Carol	102.50
	Jane	84.50
	Janet	112.50
	Joyce	50.50
	Judy	90.00
	Louise	77.00
	Mary	112.00
F	Average Weight	90.11
F	Median Weight	90.00

Output 7. Partial REPORT Procedure output

Trick 8. Use conditional logic to create multiple summary rows at the end of the report.

In this example, we want to see multiple averages calculated for the same column. An overall average, a female average, and a male average for the **weight** variable will be shown. In order to accomplish this task, the data set will have to be pre-processed again. The following DATA step does this.

```
data prep2;
    length name  $15.
    set sashelp.class;
    f = 1;
    m = 1;
   goal = 99;
run;
```

You can use what some programmers call 'holding variables' in PROC REPORT. In the following example, these variables are put to work in the compute blocks.

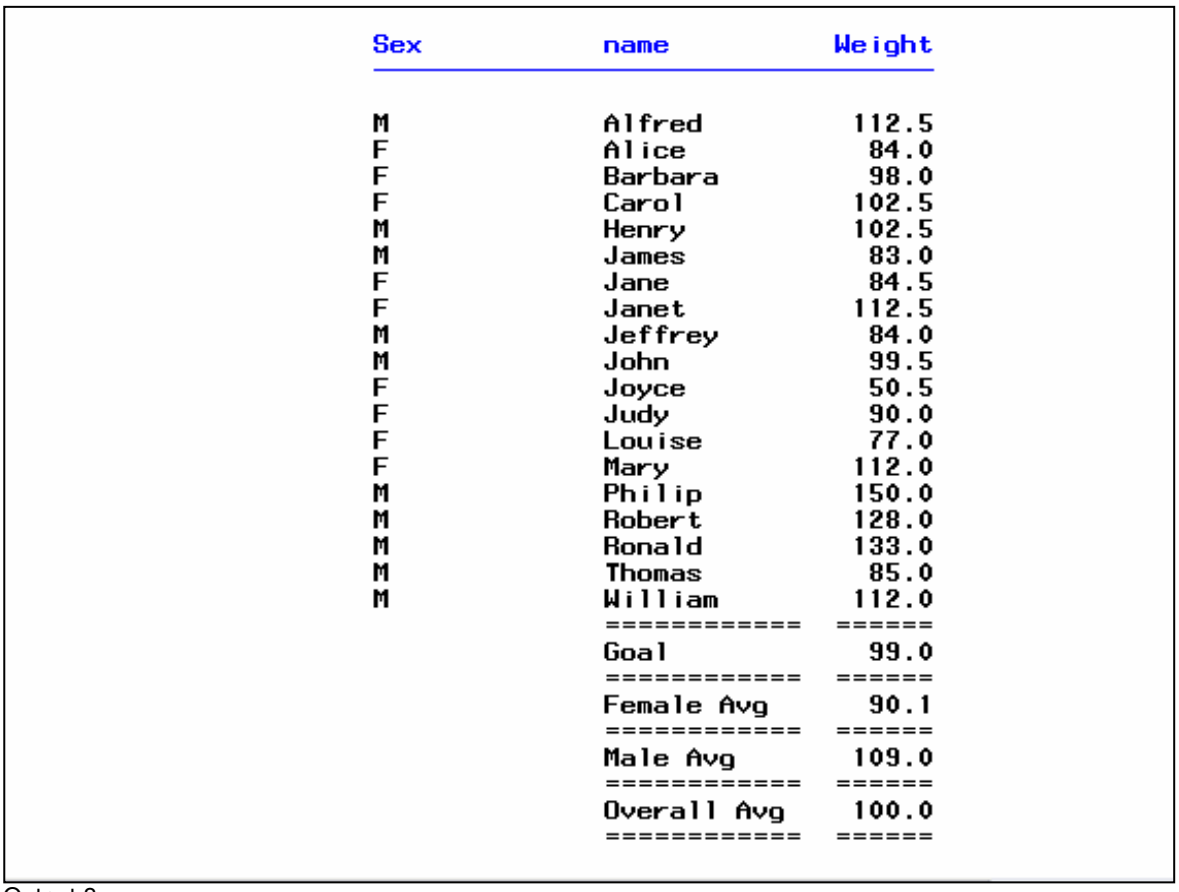
```
proc report data=prep2 nowindows;
     columns m f goal sex name weight weight=f_weight weight=m_weight ;
     define name    / display width=12;
     define sex     / display width=12 ;
     define m       / group noprint ;
     define f       / group noprint ;
     define goal    / group noprint ;
     define weight    / analysis mean format=6.1;
     define f_weight / sum noprint;
     define m_weight / sum noprint;
     *-------------------------------------------------------*;
```

Partial REPORT Procedure, Program 8.

```
     *-------------------------------------------------------*;
     compute weight;
          if sex="M" then do;  wholdm+weight.mean; mw+1; end;
          if sex="F" then do;  wholdf+weight.mean; wf+1; end;
     endcomp;
     *-------------------------------------------------------*;
     break after f / summarize  dul;
        compute after f;
            name='Female Avg';
            weight.mean = wholdf/wf;
        endcompute;
     break after m / summarize dul;
        compute after m;
            name='Male Avg';
            weight.mean=wholdm/mw;
        endcompute;
     break after goal / summarize dol dul ;
        compute after goal;
            name='Goal';
            weight.mean=goal;
        endcompute;
     rbreak after / summarize dul;
        compute after ;
            name='Overall Avg';
            weight=weight.mean;
        endcompute;
run;
```

Partial REPORT Procedure, Program 8.

The variable **wholdm** and **wholdf** are 'holding variables', you can use them in calculations, but they do not appear in the report. Since the variables **f, m**, and **goal** have the same values for all observations, the BREAK statements have the same effect as the RBREAK statement; appending rows at the end of the report. Notice the output.



Sex	name	Weight
M	Alfred	112.5
F	Alice	84.0
F	Barbara	98.0
F	Carol	102.5
M	Henry	102.5
M	James	83.0
F	Jane	84.5
F	Janet	112.5
M	Jeffrey	84.0
M	John	99.5
F	Joyce	50.5
F	Judy	90.0
F	Louise	77.0
F	Mary	112.0
M	Philip	150.0
M	Robert	128.0
M	Ronald	133.0
M	Thomas	85.0
M	William	112.0
	Goal	99.0
	Female Avg	90.1
	Male Avg	109.0
	Overall Avg	100.0

Output 8.

Trick 9. Use ODS to enhance the report.

With the Output Delivery System, you can send your output to any number of locations as well as create special formatting. In the next example, the last PROC REPORT step is 'sandwiched' between basic ODS statements.

```
ods  listing close;
ods rtf file = 'c:\sugi30.rtf';
     previous proc report step
ods pdf close;
```

The first ODS statement closes the output window, while the second ODS statement opens a file for the output. The third one closes the rtf file.

Sex	Name	Weight
M	Alfred	112.5
F	Alice	84.0
F	Barbara	98.0
F	Carol	102.5
M	Henry	102.5
M	James	83.0
F	Jane	84.5
M	Jeffrey	84.0
M	John	99.5
F	Joyce	50.5
F	Judy	90.0
F	Louise	77.0
M	Robert	128.0
M	Thomas	85.0
	Goal	*99.0*
	Female Avg	*83.8*
	Male Avg	*99.2*
	Overall Avg	*91.5*

Output 9. ODS.

Notice the fonts of the column headers as well as the summary rows. Also notice other aesthetic considerations Such as the color of the fonts, and background color of each of the cells. We can use ODS to control every single attribute of the report.

Trick 10. Enhance the report with a few ODS features.

With ODS comes some new features in the syntax, including the **STYLE(*area*)=** option, where area = some part of the report. The areas that will be effected in this task are the columns, summary rows, and headers. We are going to put the STYLE(*area*)= option to work on the PROC statement first. Notice the code below only contains the PROC statement. Also notice which attributes are going to be effected.

```
ods rtf file='c:\sugi30.rtf';
proc report data=prep2(where=(age lt 15)) nowindows
     style(column) ={font_face='Arial'}
     style(summary)={font=('Arial,Helvetica, Helv') font_size=12.25pt}
     style(header) ={font_face='Arial' font_size=13.70pt};
```

PROC statement for Task 10.

The rest of PROC REPORT is the same as it was for the previous task except for the BREAK and the RBREAK statements. Notice the style= options and the attributes that they are controlling in the rest of the PROC REPORT step. Also notice that because they are on a BREAK or RBREAK statement, they do not have the area in parenthesis.

```
      *-------------------------------------------------------*;
      break after f / summarize style=[font_weight=bold font_size=12.50pt
                                background=cyan font_face='Arial'];
         compute after f;
             name='Female Avg';
             weight.mean = wholdf/wf;
         endcompute;
      break after m / summarize style=[font_weight=bold font_size=12.50pt
                                background=light green font_face='Arial'];
         compute after m;
             name='Male Avg';
             weight.mean=wholdm/mw;
         endcompute;
      break after goal / summarize style=[font_weight=bold font_size=12.50pt
                                   background=pink font_face='Arial'];
         compute after goal;
             name='Goal';
             weight.mean=goal;
         endcompute;
      rbreak after / summarize style=[font_weight=bold font_size=13.50pt
                               background=yellow font_face='Arial'];
         compute after ;
             name='Overall Avg';
             weight=weight.mean;
         endcompute;
run;
ods rtf close;
```

Last part of PROC REPORT step.

The effects of these style options can be seen in the output below. Notice that each of the summary rows in the report have different colors and font sizes. The higher the level of detail, the smaller the font. This output is intended to show some of the things you can do with the style options of ODS and the REPORT procedure. Hopefully, this output will give you some ideas of the possibilities that are available with the ODS. The sky is truly the limit when it comes to generating attractive reports with the SAS system.

Sex	Name	Weight
M	Alfred	112.5
F	Alice	84.0
F	Barbara	98.0
F	Carol	102.5
M	Henry	102.5
M	James	83.0
F	Jane	84.5
M	Jeffrey	84.0
M	John	99.5
F	Joyce	50.5
F	Judy	90.0
F	Louise	77.0
M	Robert	128.0
M	Thomas	85.0
	Goal	**99.0**
	Female Avg	**83.8**
	Male Avg	**99.2**
	Overall Avg	**91.5**

Output 10.

ACKNOWLEDGMENTS

I would like to acknowledge and greatly thank the Technical Support Department at SAS Institute for their helpful knowledge and expertise that they so freely gave.

CONTACT INFORMATION

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

Ben Cochran
The Bedford Group
Raleigh, NC 27607
Phone: 919.741.0370
Email: bedfordgroup@nc.rr.com