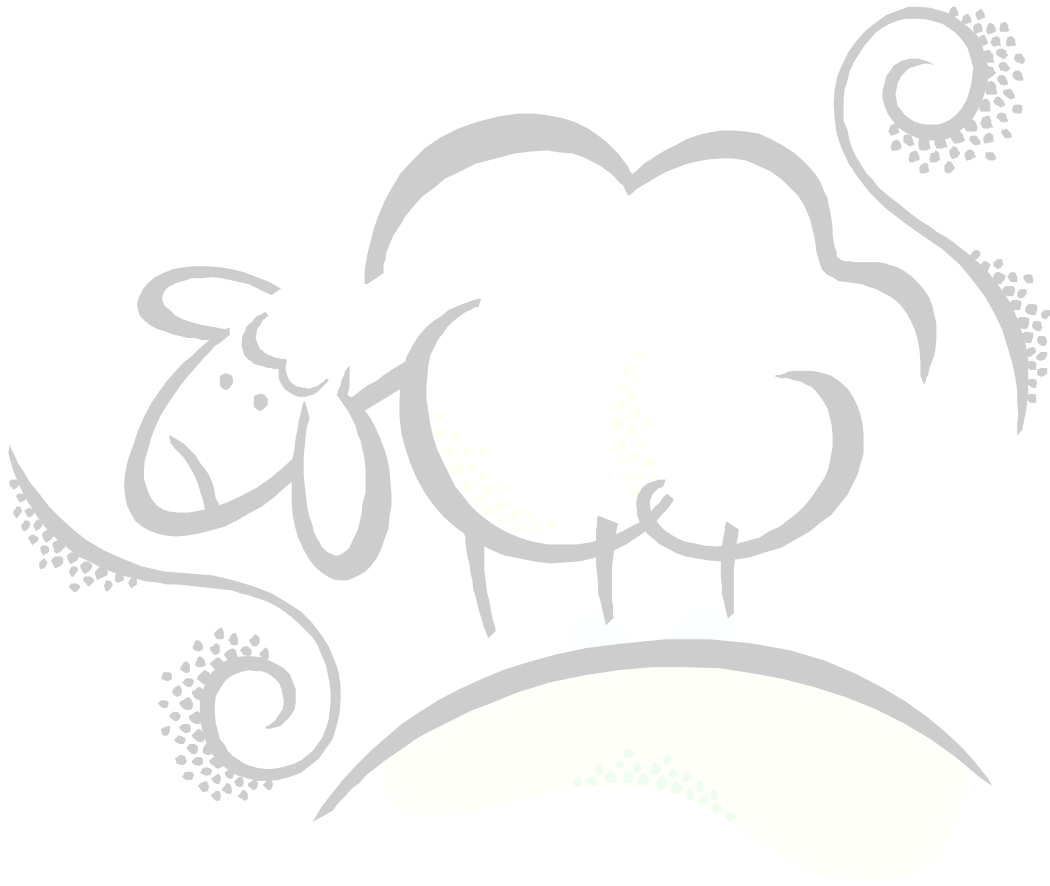


Introduction  
to the  
SAS  
Output  
Delivery  
System  
(ODS)

This work may be reproduced and redistributed, in whole or in part, without alteration and without prior written permission, provided all copies contain the following statement:

Copyright ©2011 sheepsqueezers.com. This work is reproduced and distributed with the permission of the copyright holder.

This presentation as well as other presentations and documents found on the sheepsqueezers.com website may contain quoted material from outside sources such as books, articles and websites. It is our intention to diligently reference all outside sources. Occasionally, though, a reference may be missed. No copyright infringement whatsoever is intended, and all outside source materials are copyright of their respective author(s).



## INTRODUCTION

This document introduces you to the basics of the SAS Output Delivery System (ODS). ODS provides a multitude of choices for reporting and displaying analytical results from the SAS System. For example, you can create Adobe Acrobat Portable Document Format (PDF) files, HTML files for use on the internet, and Rich Text Format (RTF) files for use with Microsoft Word or WordPad that contain your SAS procedure output, SAS graphs, etc. But, this is just the tip of the iceberg with SAS ODS, as you will see in the remaining part of this document.

As many of you are aware, several SAS procedures like `FREQ`, `MEANS`, etc. give the user the ability to create an output SAS dataset containing the results of the procedure. Both `FREQ` and `MEANS` have the `OUT=` option which allows you to specify the name of your output SAS dataset. Some of you may have noticed that the newer SAS procedures, like `SURVEYMEANS`, `SURVEYREG`, etc., do not come with this functionality. The ability to create an output SAS dataset from these procedures is via the SAS Output Delivery System (ODS) and its `ODS OUTPUT` statement.

And, as they say on those late-night television commercials: But, wait! There's more!

The SAS Output Delivery System (ODS) allows you to modify the look-and-feel of the reports based on the fonts, colors, etc. – the *style*, if you please – that you want to use rather than settling for SAS-supplied defaults. You have control over styles for the column headers, table headers, table footers, cell data, etc. No longer do you have to settle for SAS's line printer-like output...blech!

And, as they say on those early-morning television commercials: But, wait! There's even more!

The SAS Output Delivery System (ODS) allows you to output in formats like Extensible Markup Language (XML), Comma-Separated Values (CSV) and Web Markup Language (WML) formats – known as *tagsets*. And, you can even define your own tagsets, if you want!

And, as they say on those mid-afternoon television commercials: But, wait! There's still more!

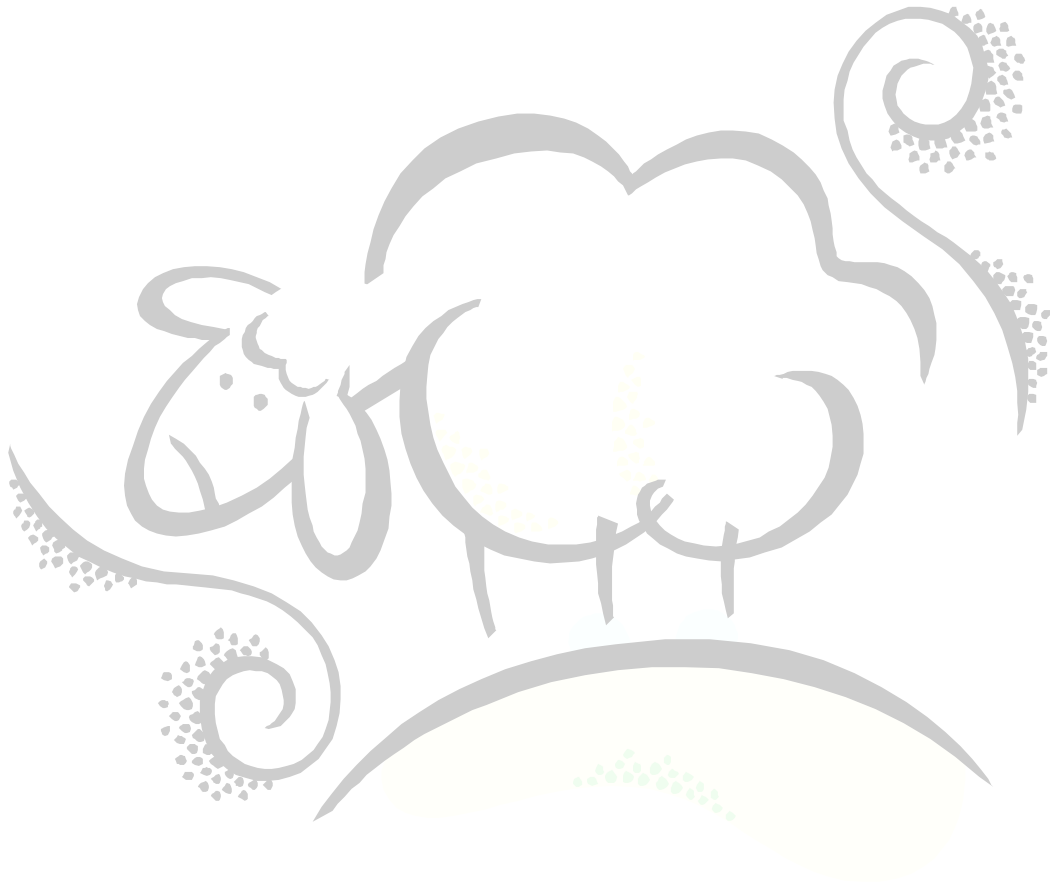
You can save your ODS results in a special ODS file called a *document* so that you can "replay" one or more of them at a later time without having to re-run you analysis! No more waiting for analyses to re-run if you just want to see or use the results again!

Finally, as they say on those early-evening television commercials: What are you waiting for?

# PART I

## ODS Kick Start

This section briefly introduces the basic features of the SAS Output Delivery System (ODS) which you are most likely to use on a day-to-day basis as you work with the SAS System. This section is arranged in a Question and Answer format so that the reader does not have to search through pages and pages of material to find the desired topic.



## Q.1: How do I create a SAS dataset from a SAS procedure when there's no OUT= statement?

A.1:

In order to create a SAS dataset from a SAS procedure that does not have an OUT= statement – or even with a SAS procedure that does have one – you must first determine the special names SAS gives to each part of the desired SAS procedure output. This special name is known as a path in ODS terminology. For example, given the SAS dataset below,

```
data FatKids;
infile cards;
input @1 KidName $char8.
      @10 HeightInInches 2.
      @15 WeightInPounds 3.;
cards;
ALBERT    45    150
ROSEMARY  35    123
TOMMY     78    167
BUDDY     12    189
FARQUAR   76    198
SIMON     87    256
LAUREN    54    876
;
run;
```

Let's assume we want to run a UNIVARIATE procedure on the variable WeightInPounds from the FatKids dataset. The first thing we have to do is run the UNIVARIATE procedure with the ODS TRACE statement turned on, followed by the UNIVARIATE and the the ODS TRACE statement turned back off:

```
ods trace on;
proc univariate data=FatKids;
  var WeightInPounds;
run;
ods trace off;
```

The output from these commands appears in the SAS Log and looks like this:

```
Output Added:
-----
Name:          Moments
Label:         Moments
Template:      base.univariate.Moments
Path:          Univariate.WeightInPounds.Moments
-----
```

Output Added:

```
-----  
Name:      BasicMeasures  
Label:     Basic Measures of Location and Variability  
Template:  base.univariate.Measures  
Path:     Univariate.WeightInPounds.BasicMeasures  
-----
```

Output Added:

```
-----  
Name:      TestsForLocation  
Label:     Tests For Location  
Template:  base.univariate.Location  
Path:     Univariate.WeightInPounds.TestsForLocation  
-----
```

Output Added:

```
-----  
Name:      Quantiles  
Label:     Quantiles  
Template:  base.univariate.Quantiles  
Path:     Univariate.WeightInPounds.Quantiles  
-----
```

Output Added:

```
-----  
Name:      ExtremeObs  
Label:     Extreme Observations  
Template:  base.univariate.ExtObs  
Path:     Univariate.WeightInPounds.ExtremeObs  
-----
```

You will notice that there are five sections above, each one corresponds to the five sections produced by the SAS UNIVARIATE procedure. The Name and Label in each section above nearly corresponds to the title of each section produced from the UNIVARIATE procedure itself. We discuss the Template later on in this document. The most important thing to know is the Path name. In the above, the Path name for the moments is Univariate.WeightInPounds.Moments. Clearly, this period-delimited text string is the name of the SAS procedure, followed by the variable of interest, followed by the name SAS ODS gives to the moments section from the UNIVARIATE procedure. Now, let's say we want to create a SAS dataset containing the moments for WeightInPounds. You would use these commands to accomplish this:

```
ods output  
Univariate.WeightInPounds.Moments=FatMomentsDataSet;  
proc univariate data=FatKids;  
  var WeightInPounds;  
run;  
ods output close;
```



Note that the command ODS OUTPUT is followed by name of the desired Path associated with the Moments section of the UNIVARIATE procedure followed by an equal sign followed by the name of our SAS dataset FatMomentsDataset. Here is what that dataset looks like:

| VarName        | Label1          | cValue1    | nValue1    | Label2           | cValue2    | nValue2     |
|----------------|-----------------|------------|------------|------------------|------------|-------------|
| WeightInPounds | N               | 7          | 7.000000   | Sum Weights      | 7          | 7.000000    |
| WeightInPounds | Mean            | 279.857143 | 279.857143 | Sum Observations | 1959       | 1959.000000 |
| WeightInPounds | Std Deviation   | 266.181284 | 266.181284 | Variance         | 70852.4762 | 70852       |
| WeightInPounds | Skewness        | 2.51231272 | 2.512313   | Kurtosis         | 6.44913858 | 6.449139    |
| WeightInPounds | Uncorrected SS  | 973355     | 973355     | Corrected SS     | 425114.857 | 425115      |
| WeightInPounds | Coeff Variation | 95.1132716 | 95.113272  | Std Error Mean   | 100.607069 | 100.607069  |

Take note that the dataset FatMomentsDataSet is laid out in a similar format as the actual UNIVARIATE Moments output. That is, instead of there being one row per statistic, there two sets of three columns with six rows of data. One set of three columns contains the N, Mean, Std Deviation, Skewness, Uncorrected SS, and Coeff Variation statistics while the other set of three columns contains the Sum Weights, Sum Observations, Variance, Kurtosis, Corrected SS and Std Error Mean statistics. Be aware of this and you'll be able to sleep better at night. Note also that cValue1 and cValue2 are the character representations of nValue1 and nValue2.

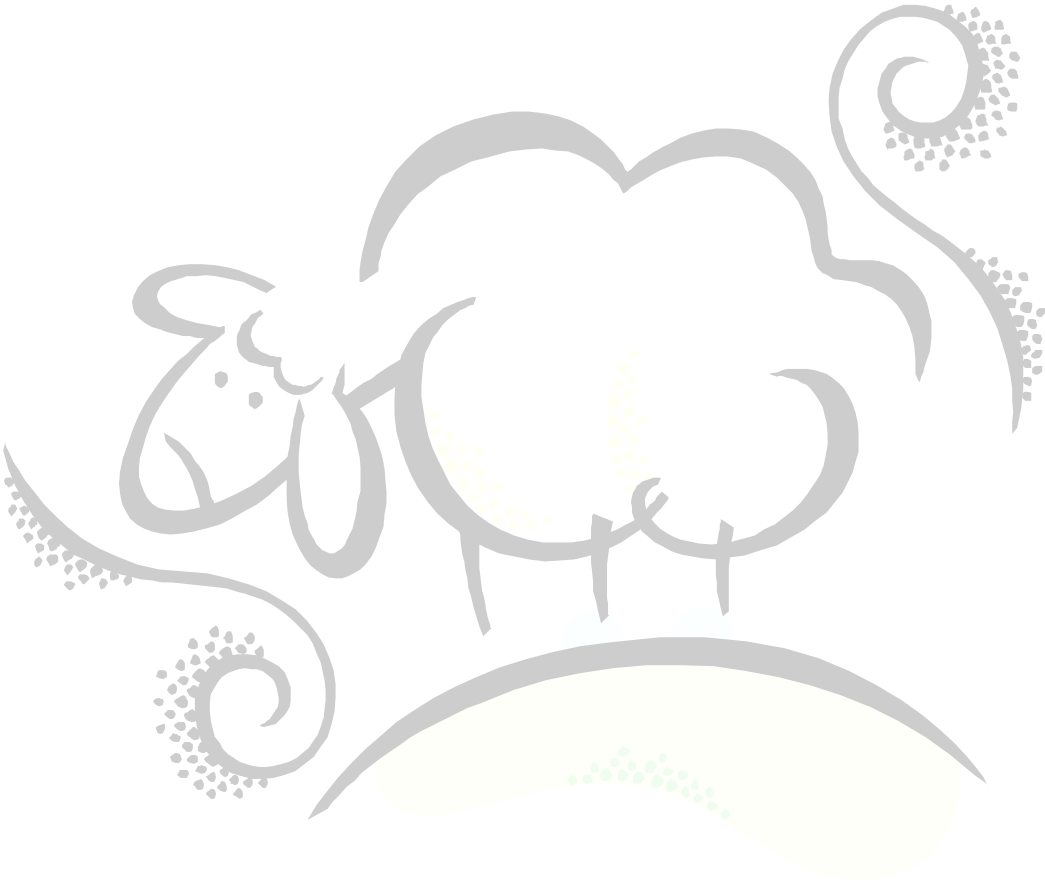
You might be wondering what would happen if you asked for a UNIVARIATE on both variables WeightInPounds and HeightInInches. Based on the SAS code above, you would still ONLY get the moments for WeightInPounds because that is the Path you specified above. In order to get all of the variables to appear in your output SAS dataset, you would need to use the Name rather than the Path in the code above:

```
ods output Moments=FatMoments;
proc univariate data=FatKids;
  var WeightInPounds HeightInInches;
run;
ods output close;
```

Here is what the dataset FatMoments looks like:

| VarName        | Label1          | cValue1    | nValue1    | Label2           | cValue2    | nValue2     |
|----------------|-----------------|------------|------------|------------------|------------|-------------|
| WeightInPounds | N               | 7          | 7.000000   | Sum Weights      | 7          | 7.000000    |
| WeightInPounds | Mean            | 279.857143 | 279.857143 | Sum Observations | 1959       | 1959.000000 |
| WeightInPounds | Std Deviation   | 266.181284 | 266.181284 | Variance         | 70852.4762 | 70852       |
| WeightInPounds | Skewness        | 2.51231272 | 2.512313   | Kurtosis         | 6.44913858 | 6.449139    |
| WeightInPounds | Uncorrected SS  | 973355     | 973355     | Corrected SS     | 425114.857 | 425115      |
| WeightInPounds | Coeff Variation | 95.1132716 | 95.113272  | Std Error Mean   | 100.607069 | 100.607069  |
| HeightInInches | N               | 7          | 7.000000   | Sum Weights      | 7          | 7.000000    |
| HeightInInches | Mean            | 55.2857143 | 55.285714  | Sum Observations | 387        | 387.000000  |
| HeightInInches | Std Deviation   | 26.9054783 | 26.905478  | Variance         | 723.904762 | 723.904762  |
| HeightInInches | Skewness        | -0.4556582 | -0.455658  | Kurtosis         | -0.8111112 | -0.811111   |
| HeightInInches | Uncorrected SS  | 25739      | 25739      | Corrected SS     | 4343.42857 | 4343.428571 |
| HeightInInches | Coeff Variation | 48.6662398 | 48.666240  | Std Error Mean   | 10.1693149 | 10.169315   |

Notice that both WeightInPounds and HeightInInches appear in the SAS dataset FatMoments.



## Q.2: How do I create an Adobe Acrobat PDF File from all of my SAS procedures?

### A.2:

In order to create a PDF file – or an RTF file, or HTML file, etc. – you need to specify the ODS statement followed by your desired output option like PDF, RTF, HTML, etc. These options are known as an output *destination* in ODS lingo. You then follow this with as many SAS procedures as you want and the output from each one is placed in your desired destination. Next, you must “close” the destination using the ODS CLOSE statement. Here is an example using our FatKids dataset to create an Adobe Acrobat PDF file:

```
ods listing close;
ods pdf file="C:\FatKids_Analysis1.pdf";

proc univariate data=FatKids;
  var WeightInPounds HeightInInches;
  title1 'Univariate on the FatKids';
run;

proc print data=FatKids;
  var WeightInPounds HeightInInches;
  title1 'FatKids Data';
run;

proc corr data=FatKids;
  var WeightInPounds HeightInInches;
  title1 'Correlation on the FatKids';
run;

ods pdf close;
ods listing;
```

Note that we first closed the LISTING destination in order to prevent the procedure output from being also created in the SAS Listing. Next, we opened up the PDF output destination and specified where we wanted our PDF to be stored, here in C:\FatKids\_Analysis1.pdf. We then ran three SAS procedures and then closed the PDF output destination and re-opened the LISTING output destination. The closing and opening of the LISTING output destination is a best practice you should adhere to.

Below is a screenshot when we open up the PDF file in Adobe Acrobat reader and moved to the correlations output page:

**Correlation on the FatKids** 17:15 Saturday, November 5, 2005 6

**The CORR Procedure**

2 Variables: WeightInPounds HeightInches

| Simple Statistics |   |           |           |           |           |           |
|-------------------|---|-----------|-----------|-----------|-----------|-----------|
| Variable          | N | Mean      | Std Dev   | Sum       | Minimum   | Maximum   |
| WeightInPounds    | 7 | 279.85714 | 266.18128 | 1959      | 123.00000 | 876.00000 |
| HeightInches      | 7 | 55.28571  | 26.90548  | 387.00000 | 12.00000  | 87.00000  |

| Pearson Correlation Coefficients, N = 7<br>Prob >  r  under H0: Rho=0 |                   |                   |
|---|-------------------|-------------------|
|   | WeightInPounds    | HeightInches      |
| WeightInPounds  | 1.00000           | 0.06210<br>0.8948 |
| HeightInches  | 0.06210<br>0.8948 | 1.00000           |

As you can see, the correlation between height and weight in the FatKids dataset leads us to believe that there is no relationship between the variables HeightInches and WeightInPounds, and clearly leads us to conclude that fat kids in general have no concept of basic statistics.

If you are not very impressed by the black-and-white output you see above, SAS provides several different built-in styles which allow you to put a little bling-bling into your output. If we included the option `STYLE=BarrettsBlue` on the code

```
ODS PDF FILE="C:\FatKids_Analysis1.pdf";
```

like this

```
ods pdf style=BarrettsBlue file="C:\FatKids_Analysis2.pdf";
```

we would see output like this:

**Correlation on the FatKids**

6  
17:15 Saturday, November 5, 2005

**The CORR Procedure**

2 Variables: WeightInPounds HeightInInches

| Simple Statistics |   |           |           |           |           |           |
|-------------------|---|-----------|-----------|-----------|-----------|-----------|
| Variable          | N | Mean      | Std Dev   | Sum       | Minimum   | Maximum   |
| WeightInPounds    | 7 | 279.85714 | 266.18128 | 1959      | 123.00000 | 876.00000 |
| HeightInInches    | 7 | 55.28571  | 26.90548  | 387.00000 | 12.00000  | 87.00000  |

| Pearson Correlation Coefficients, N = 7<br>Prob >  r  under H0: Rho=0 |                   |                   |
|---|-------------------|-------------------|
|   | WeightInPounds    | HeightInInches    |
| WeightInPounds  | 1.00000           | 0.06210<br>0.8948 |
| HeightInInches  | 0.06210<br>0.8948 | 1.00000           |

Take note of the pretty colors. SAS provides the following built-in styles for you to use with the STYLE= option:

```
Listing of: SASHELP.TMPLMST
Path Filter is: Styles
Sort by: PATH/ASCENDING
```

| Obs      | Path                       | Type         |
|----------|----------------------------|--------------|
| 1        | Styles                     | Dir          |
| 2        | Styles.Analysis            | Style        |
| 3        | Styles.Astronomy           | Style        |
| 4        | Styles.Banker              | Style        |
| <b>5</b> | <b>Styles.BarrettsBlue</b> | <b>Style</b> |
| 6        | Styles.Beige               | Style        |
| 7        | Styles.Brick               | Style        |
| 8        | Styles.Brown               | Style        |
| 9        | Styles.Curve               | Style        |

|    |                        |       |
|----|------------------------|-------|
| 10 | Styles.D3d             | Style |
| 11 | Styles.Default         | Style |
| 12 | Styles.EGDefault       | Style |
| 13 | Styles.Education       | Style |
| 14 | Styles.Electronics     | Style |
| 15 | Styles.Festival        | Style |
| 16 | Styles.FestivalPrinter | Style |
| 17 | Styles.Gears           | Style |
| 18 | Styles.Journal         | Style |
| 19 | Styles.Magnify         | Style |
| 20 | Styles.Meadow          | Style |
| 21 | Styles.MeadowPrinter   | Style |
| 22 | Styles.Minimal         | Style |
| 23 | Styles.Money           | Style |
| 24 | Styles.NoFontDefault   | Style |
| 25 | Styles.Normal          | Style |
| 26 | Styles.NormalPrinter   | Style |
| 27 | Styles.Printer         | Style |
| 28 | Styles.Rsvp            | Style |
| 29 | Styles.Rtf             | Style |
| 30 | Styles.Sasweb          | Style |
| 31 | Styles.Sasweb2         | Style |
| 32 | Styles.Science         | Style |
| 33 | Styles.Seaside         | Style |
| 34 | Styles.SeasidePrinter  | Link  |
| 35 | Styles.Sketch          | Style |
| 36 | Styles.Statdoc         | Style |
| 37 | Styles.Statistical     | Style |
| 38 | Styles.Theme           | Style |
| 39 | Styles.Torn            | Style |
| 40 | Styles.Watercolor      | Style |
| 41 | Styles.blockPrint      | Style |
| 42 | Styles.fancyPrinter    | Style |
| 43 | Styles.sansPrinter     | Style |
| 44 | Styles.sasdocPrinter   | Style |
| 45 | Styles.serifPrinter    | Style |

Try several different styles and impress your friends!

### Q.3: How do I create an HTML file from all of my SAS procedures for use on the web?

#### A.3:

Similar to creating a PDF file in Q.2 above, instead of providing the PDF output destination, we provide the HTML output destination:

```
ods listing close;
ods html style=BarrettsBlue file="C:\FatKids_Analysis2.html";
proc univariate data=FatKids;
  var WeightInPounds HeightInInches;
  title1 'Univariate on the FatKids';
run;

proc print data=FatKids;
  var WeightInPounds HeightInInches;
  title1 'FatKids Data';
run;

proc corr data=FatKids;
  var WeightInPounds HeightInInches;
  title1 'Correlation on the FatKids';
run;
ods html close;
ods listing;
```

Note that we still provided the FILE= option and we closed the HTML output destination after we finished adding all of our procedures. A screenshot from Internet Explorer appears below:

The screenshot shows a web browser window displaying SAS output. At the top, there is a table with three rows of data:

|   |     |    |
|---|-----|----|
| 5 | 198 | 76 |
| 6 | 256 | 87 |
| 7 | 876 | 54 |

Below this is a section titled "Correlation on the FatKids" with the subtitle "The CORR Procedure". It indicates "2 Variables: WeightInPounds HeightInInches".

Next is a "Simple Statistics" table:

| Variable       | N | Mean      | Std Dev   | Sum       | Minimum   | Maximum   |
|----------------|---|-----------|-----------|-----------|-----------|-----------|
| WeightInPounds | 7 | 279.85714 | 266.18128 | 1959      | 123.00000 | 876.00000 |
| HeightInInches | 7 | 55.28571  | 26.90548  | 387.00000 | 12.00000  | 87.00000  |

Finally, there is a "Pearson Correlation Coefficients, N = 7" table with the note "Prob > |r| under H0: Rho=0":

|                | WeightInPounds    | HeightInInches    |
|----------------|-------------------|-------------------|
| WeightInPounds | 1.00000           | 0.06210<br>0.8948 |
| HeightInInches | 0.06210<br>0.8948 | 1.00000           |

Now, if we wanted to get really fancy, we could have ODS create a web document with several frames appearing in Internet Explorer. Here is how to do that:

```
ods listing close;
ods html style=BarrettsBlue body="C:\FatKids_body.html"
contents="C:\FatKids_contents.html"
frame="C:\FatKids_frame.html"
page="C:\FatKids_page.html";
proc univariate data=FatKids;
  var WeightInPounds HeightInInches;
  title1 'Univariate on the FatKids';
run;

proc print data=FatKids;
  var WeightInPounds HeightInInches;
  title1 'FatKids Data';
run;

proc corr data=FatKids;
  var WeightInPounds HeightInInches;
  title1 'Correlation on the FatKids';
run;
ods html close;
ods listing;
```



If you open up the FatKids\_frame.html page in Internet Explorer and move to the correlations section, here is what you will see:

The screenshot shows a web browser window with the following content:

- Left Frame (Navigation):**
  - 1. The Univariate Procedure
    - WeightInPounds
    - Moments
    - Basic Measures of Location and Variability
    - Tests For Location
    - Quantiles
    - Extreme Observations
  - HeightInInches
    - Moments
    - Basic Measures of Location and Variability
    - Tests For Location
    - Quantiles
    - Extreme Observations
- 2. The Print Procedure
  - Data Set WORK.FATKIDS
- 3. The Cor Procedure
  - Variables Information
  - Simple Statistics
  - Pearson Correlations

- Top-Left Frame (Data Table):**

|   |     |    |
|---|-----|----|
| 5 | 198 | 78 |
| 6 | 256 | 87 |
| 7 | 876 | 54 |
- Main Content Area:**

**Correlation on the FatKids**

The CORR Procedure

2 Variables: WeightInPounds HeightInInches

| Simple Statistics |   |           |           |           |           |           |
|-------------------|---|-----------|-----------|-----------|-----------|-----------|
| Variable          | N | Mean      | Std Dev   | Sum       | Minimum   | Maximum   |
| WeightInPounds    | 7 | 279.85714 | 266.18128 | 1959      | 123.00000 | 876.00000 |
| HeightInInches    | 7 | 55.28571  | 26.90548  | 387.00000 | 12.00000  | 87.00000  |

| Pearson Correlation Coefficients, N = 7 |                   |                   |
|---|-------------------|-------------------|
| Prob >  r  under H0: Rho=0              |                   |                   |
|   | WeightInPounds    | HeightInInches    |
| WeightInPounds                          | 1.00000           | 0.06210<br>0.8948 |
| HeightInInches                          | 0.06210<br>0.8948 | 1.00000           |

Note that the frame on the right contains the web page FatKids\_body.html, the frame on the upper-left contains the web page FatKids\_contents.html, and the frame on the lower-left contains the web page FatKids\_page.html.

Notice that the style BarrettsBlue works equally well in a PDF file as it does in a web page. This is true for almost all of the output destinations and styles you provide.

#### Q.4: Can I create a comma-delimited text file from my SAS dataset using ODS?

#### A.4:

Similar to creating a PDF and HTML file above, we can use the CSV or CSVALL output destinations in association with the PRINT procedure. The CSV destination creates a comma-delimited file from your dataset with each character column surrounded by double-quotes. CSVALL is the same except titles and footnotes are placed in the file as well. Here is the code to run both output destinations and their output data:

```
ods listing close;
ods csv file="C:\FatKids_Data.csv";
proc print data=FatKids noobs;
  var KidName HeightInInches WeightInPounds;
  title "FatKid Data";
run;
ods csv close;
ods listing;
```

```
"KidName", "HeightInInches", "WeightInPounds"
"ALBERT", 45, 150
"ROSEMARY", 35, 123
"TOMMY", 78, 167
"BUDDY", 12, 189
"FARQUAR", 76, 198
"SIMON", 87, 256
"LAUREN", 54, 876
```

```
ods listing close;
ods csvall file="C:\FatKids_Data.csvall";
proc print data=FatKids noobs;
  var KidName HeightInInches WeightInPounds;
  title "FatKid Data";
run;
ods csvall close;
ods listing;
```

FatKid Data

```
"KidName", "HeightInInches", "WeightInPounds"
"ALBERT", 45, 150
"ROSEMARY", 35, 123
"TOMMY", 78, 167
"BUDDY", 12, 189
"FARQUAR", 76, 198
"SIMON", 87, 256
"LAUREN", 54, 876
```

Note that it seems like overkill to be doing a PROC PRINT just to output data to a text file. We can accomplish a similar task using a DATA \_NULL\_ step:

```
ods listing close;
ods csv file="C:\FatKids_DataStep.csv";
data _null_;
  set FatKids;
  file print ods=(
    variables=(
      KidName HeightInInches WeightInPounds
    )
  );
  put _ods_;
run;
ods csv close;
ods listing;
```

"KidName", "HeightInInches", "WeightInPounds"  
"ALBERT", 45, 150  
"ROSEMARY", 35, 123  
"TOMMY", 78, 167  
"BUDDY", 12, 189  
"FARQUAR", 76, 198  
"SIMON", 87, 256  
"LAUREN", 54, 876

Note that we specify FILE PRINT ODS= in the data step along with a list of the variables we want to keep. Next, we issue a PUT \_ODS\_ statement to tell SAS to output the data to the CSV file.

## Q.5: How can I limit the results output by my SAS procedures when I use ODS?

### A.5:

SAS procedures occasionally output a lot of information that you may not want to put in your PDF or HTML file when using ODS. You can limit the output by using the ODS SELECT command by specifying the path, name or label from the ODS TRACE command. For example, when using the UNIVARIATE procedure, say you only wanted the moments and the extreme observations to output. From the ODS TRACE in Q.1 above, you know that the output name for the moments results is Moments and the output for the extreme observations is ExtremeObs. Using the ODS SELECT command, you can limit your output like this:

```
ods select Moments
      ExtremeObs;
ods listing close;
ods pdf file="C:\FatKids_Analysis1.pdf";

proc univariate data=FatKids;
  var WeightInPounds HeightInInches;
  title1 'Univariate on the FatKids';
run;

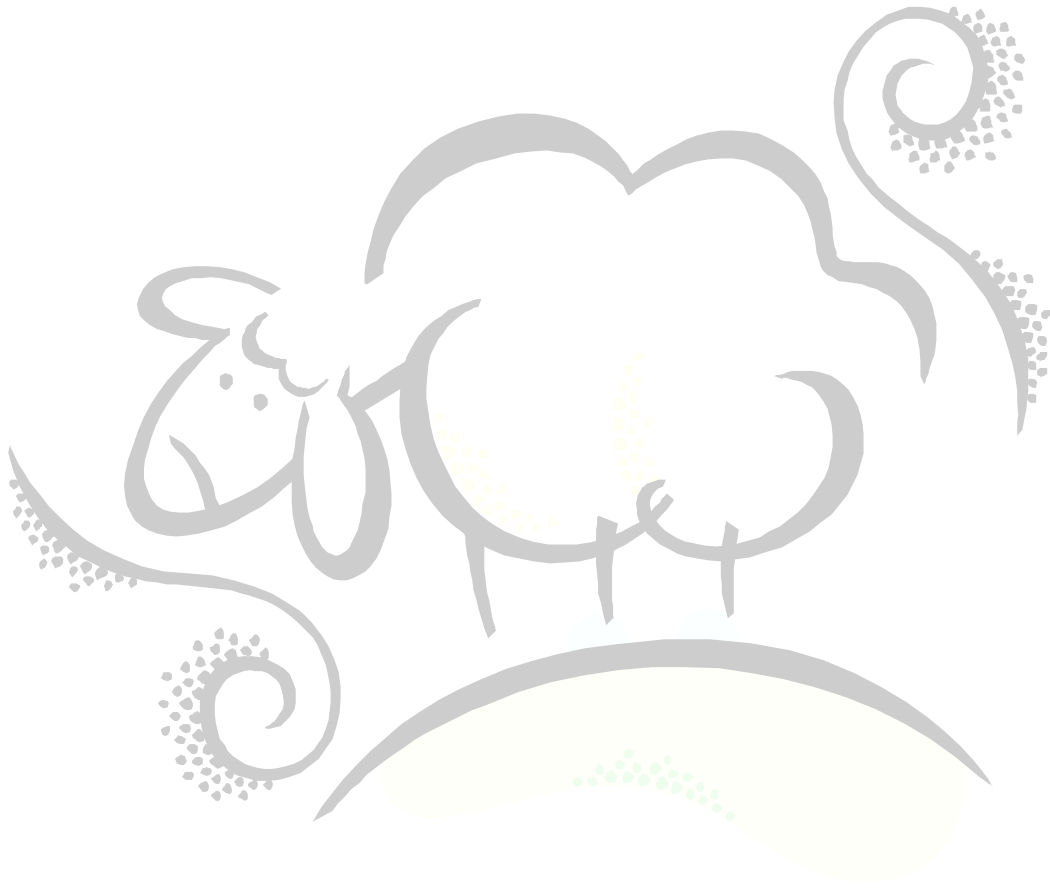
ods pdf close;
ods listing;
ods select ALL;
```

Make sure the issue ODS SELECT ALL so that you can clear out your previous selections.

## PART II

### Things Momma Never Taught You

This section briefly discusses `PROC REPORT` as well as introduces you to `HTML` (the *layout* language used in creating web pages) and `Cascading Style Sheets` (the *formatting* language used to provide web pages with font and color styles). Although they may appear to be unrelated to `SAS ODS`, the concepts presented in this section are similar to `ODS styles` and `PROC TEMPLATE` presented in Part III.



## Brief Introduction to PROC REPORT

Without going too deep into the details of how to write PROC REPORT code, we show two comprehensive examples and explain the parts. For this example, assume we are using the FatKids data:

```
proc report data=FatKids headline split="*" nowindows spacing=1 ls=256;
  column KidName HeightInInches WeightInPounds FattyIndex;
  define KidName/group 'First Name' width=10;
  define HeightInInches/analysis sum 'Height*(inches)' width=10 format=comma10.0;
  define WeightInPounds/analysis sum 'Weight*(pounds)' width=10 format=comma10.0;
  define FattyIndex/computed 'Fatty*Index' width=10 format=comma10.2;
  compute FattyIndex;
    FattyIndex=(10*_c2_ + 20*_c3_)/1000;
  endcomp;
  title1 "Fat Kid FattyIndex Computation";
  title2 "Fat Year: 2006";
run;
```

As usual, DATA= defines the name of the input dataset. The option SPLIT="\*" defines the asterisk as the split character in the column header names. The option NOWINDOWS tells SAS not to display the PROC REPORT window, but instead place the output in the SAS Listing window. The SPACING=1 option specifies the number of blank columns between the columns of data. The LS=256 option tells PROC REPORT that the maximum linesize for the report is 256 characters.

Next, we define all of the columns we are going to use in our report. Note that the column FATTYINDEX does not appear in the FatKids dataset, but is created in the COMPUTE section below and is required to be on the COLUMN line if it is going to be displayed. Next, we define all of our columns. We define the column KIDNAME to be a GROUP column with a header name of 'First Name' and a maximum width of 10 characters. We then define the HeightInInches and WeightInPounds columns to be ANALYSIS columns which will be summed. Note that although we provide GROUP and ANALYSIS variables, no actual summing by KidName will take place since the FatKid dataset is already summarized to the KIDNAME level.

Next, we define the column FATTYINDEX which is a computed column. We then move on to define the computation for the FATTYINDEX column by using the formula 10 times the HeightInInches plus 20 times the WeightInPounds all divided by 1000. Note how the code uses \_C2\_ to represent the 2<sup>nd</sup> column HeightInInches, and \_C3\_ to represent the 3<sup>rd</sup> column WeightInPounds.

Here is the corresponding output:

```
Fat Kid FattyIndex Computation
Fat Year: 2006
```

| First Name | Height<br>(inches) | Weight<br>(pounds) | Fatty<br>Index |
|------------|--------------------|--------------------|----------------|
| ALBERT     | 45                 | 150                | 3.45           |
| BUDDY      | 12                 | 189                | 3.90           |
| FARQUAR    | 76                 | 198                | 4.72           |

|          |    |     |       |
|----------|----|-----|-------|
| LAUREN   | 54 | 876 | 18.06 |
| ROSEMARY | 35 | 123 | 2.81  |
| SIMON    | 87 | 256 | 5.99  |
| TOMMY    | 78 | 167 | 4.12  |

For this next example, assume we have a SAS dataset similar to the FatKids data, but has an additional column called STAT\_DATE containing the day the height and weight were taken for each kid. Using PROC REPORT, we can define the column STAT\_DATE to be an ACROSS column which will show the date across the top of the page. Note how the COLUMN definition has changed. We have the column STAT\_DATE followed by a comma followed by a space-separated list of variables in parentheses. This indicated to PROC REPORT that STAT\_DATE goes across the top of the report and that the two columns HEIGHTININCHES and WEIGHTINPOUNDS appear as columns under the corresponding STAT\_DATE. The code and output follows:

```
proc report data=FatKidsOverTime headline split="*" nowd missing spacing=1
ls=256;
column KidName Stat_Date, (HeightInInches WeightInPounds) FattyIndex;
define KidName/group 'First Name' width=10;
define Stat_Date/across format=monyy6. order=data '-Quarter-';
define HeightInInches/analysis sum 'Height*(inches)' width=10 format=comma10.0;
define WeightInPounds/analysis sum 'Weight*(pounds)' width=10 format=comma10.0;
define FattyIndex/computed 'Fatty*Index' width=10 format=comma10.2;
compute FattyIndex;
FattyIndex=(10*( _c2_ + _c4_ + _c6_ + _c8_ ) + 20*( _c3_ + _c5_ + _c7_ + _c9_ ))/1000;
endcomp;
title1 "Fat Kid FattyIndex Computation Over Time";
title2 "Fat Year: 2006";
run;
```

Fat Kid FattyIndex Computation Over Time  
Fat Year: 2006

| First Name | Quarter         |                 |                 |                 |                 |                 |                 |                 | Fatty Index |
|------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-------------|
|            | JAN06           |                 | APR06           |                 | JUL06           |                 | OCT06           |                 |             |
|            | Height (inches) | Weight (pounds) | Height (inches) | Weight (pounds) | Height (inches) | Weight (pounds) | Height (inches) | Weight (pounds) |             |
| ALBERT     | 45              | 150             | 45              | 160             | 45              | 170             | 45              | 190             | 15.20       |
| BUDDY      | 12              | 189             | 12              | 199             | 12              | 219             | 12              | 249             | 17.60       |
| FARQUAR    | 76              | 198             | 76              | 198             | 76              | 218             | 76              | 218             | 19.68       |
| LAUREN     | 54              | 876             | 54              | 886             | 54              | 896             | 54              | 976             | 74.84       |
| ROSEMARY   | 35              | 123             | 35              | 133             | 35              | 143             | 35              | 163             | 12.64       |
| SIMON      | 87              | 256             | 87              | 266             | 87              | 276             | 87              | 356             | 26.56       |
| TOMMY      | 78              | 167             | 78              | 177             | 78              | 187             | 78              | 217             | 18.08       |

Note that we also had to update the FATTYINDEX since there are now more columns in the PROC REPORT due to the ACROSS option provided on STAT\_DATE.

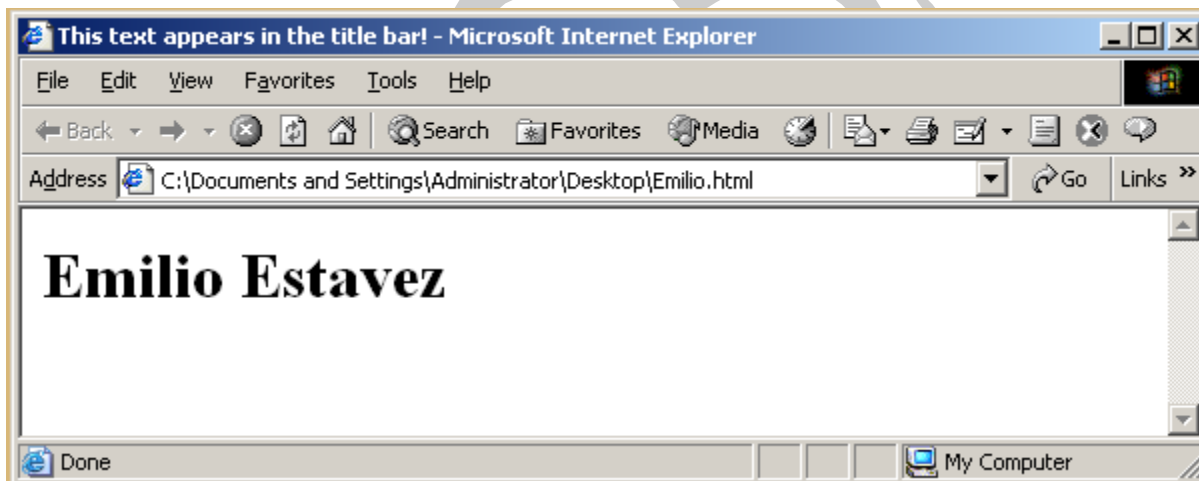
To relate this back to SAS ODS, the concept of defining columns and computing new columns in PROC REPORT appears in a similar way in PROC TEMPLATE.



## Brief Introduction to HTML and CSS

Many of you are familiar with the web...that's where you can find all sorts of information related to porn. In order to make those informational webpages, you use the language of the web called Hypertext Markup Language (HTML). For example, to display the following in a webpage in Internet Explorer, all you have to do is open up a text editor, type in the text and then open the file using Internet Explorer:

```
<HTML>
<HEAD>
  <TITLE>
    This text appears in the title bar!
  </TITLE>
</HEAD>
<BODY>
  <H1>Emilio Estavez</H1>
</BODY>
</HTML>
```



As you can see, HTML is made up of *HTML tags* such as <HTML>, <HEAD>, <BODY>, etc. These HTML tags tell Internet Explorer how to display information like the name Emilio Estavez. All HTML files start with the *starting HTML tag* <HTML> and end with *ending HTML tag* </HTML>. There are two major sections that follow: the HEAD section, which contains header-type information like what to display in the title bar; and the BODY section, which is what is actually displayed on the page. The code in the HEAD section is never displayed in the webpage. As you can see above, in the HEAD section, we define the TITLE to be displayed at the top of Internet Explorer, while in the BODY section we display the name Emilio Estavez in the largest font available by using the H1 tag. Note that what appears in Internet Explorer is not very stylistic because the HTML tags do not define style but only define layout on the page. We'll get to style later on.

Let's create a more interesting web page using the data for the FatKids. In this case, we use the TABLE tag to create an Excel-like spreadsheet. Within the TABLE we have rows and columns. Each row is defined by using a TR tag. Each column – the data appearing in the rows, really – are defined by using the TD tag. Think of TR as meaning TABLE ROW and TD as meaning TABLE DATA. Here is the HTML code to display the FatKids data:

```

<HTML>
  <HEAD>
    <TITLE>
      FatKids 2006 Data
    </TITLE>
  </HEAD>

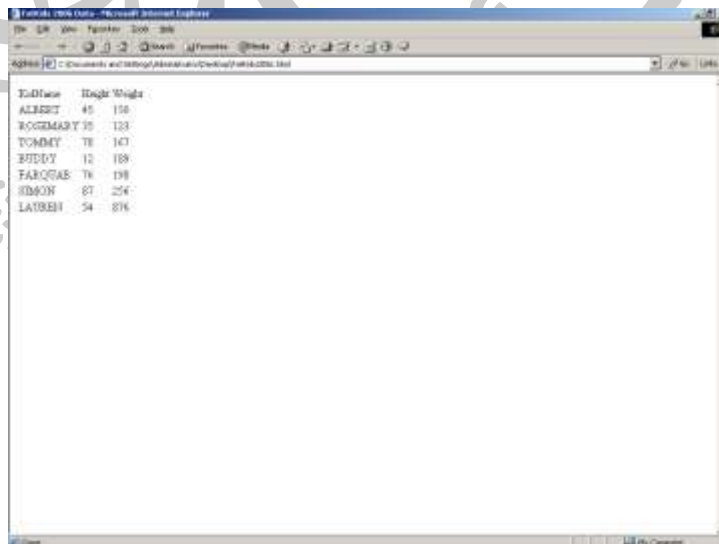
  <BODY>

    <TABLE>
      <TR> <TD>KidName </TD> <TD>Height</TD> <TD>Weight</TD> </TR>
      <TR> <TD>ALBERT </TD> <TD>45 </TD> <TD>150 </TD> </TR>
      <TR> <TD>ROSEMARY</TD> <TD>35 </TD> <TD>123 </TD> </TR>
      <TR> <TD>TOMMY </TD> <TD>78 </TD> <TD>167 </TD> </TR>
      <TR> <TD>BUDDY </TD> <TD>12 </TD> <TD>189 </TD> </TR>
      <TR> <TD>FARQUAR </TD> <TD>76 </TD> <TD>198 </TD> </TR>
      <TR> <TD>SIMON </TD> <TD>87 </TD> <TD>256 </TD> </TR>
      <TR> <TD>LAUREN </TD> <TD>54 </TD> <TD>876 </TD> </TR>
    </TABLE>

  </BODY>
</HTML>

```

Note that we define a table using the starting tag `<TABLE>` and the ending tag `</TABLE>`. We define each row with the starting tag `<TR>` and the ending tag `</TR>`, and each data element (or cell) using the starting tag `<TD>` and ending tag `</TD>`. When you open this HTML file up in Internet Explorer, here is what you see:



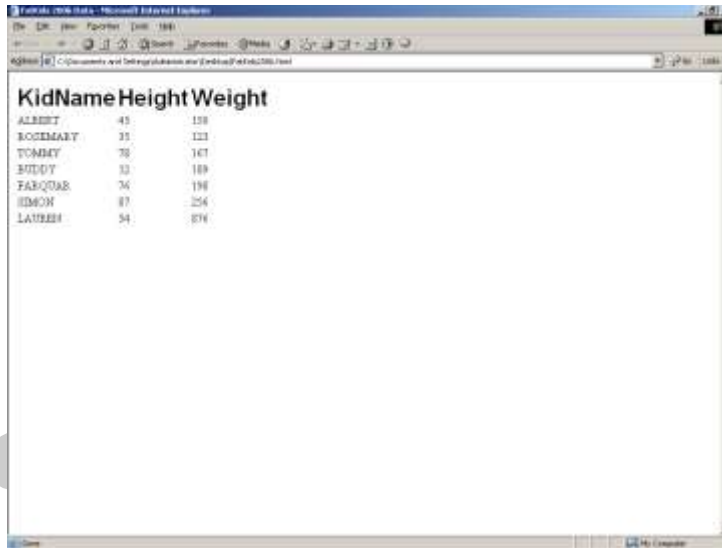
Again, this is not very stylistic: the headers are not bold, the font is not Courier, the numbers are not right-justified. Here is where **Cascading Style Sheets (CSS)** comes in. CSS allows you to define the font, font size, and other more stylistic things for the individual items in your webpage. For example, suppose I wanted to use the Arial 24-point bold font for the headers. Here is what the code looks like to accomplish this:

```

<TR STYLE="font-family:Arial;font-size:24pt;font-weight:bold"><TD>KidName</TD><TD>Height</TD><TD>Weight</TD></TR>

```

As you can see, we are using the STYLE= element inside the TR tag that contains the headers. To define the font we want, we use the font-family attribute followed by a colon followed by the name of the font we want to use, Arial in this case. To define the font size, we use the font-size attribute followed by a colon followed by the number of points we want the font to be, 24pt in this case. To make the font bold, we use the font-weight attribute followed by a colon followed by the word bold. And here is what that looks like in Internet Explorer:

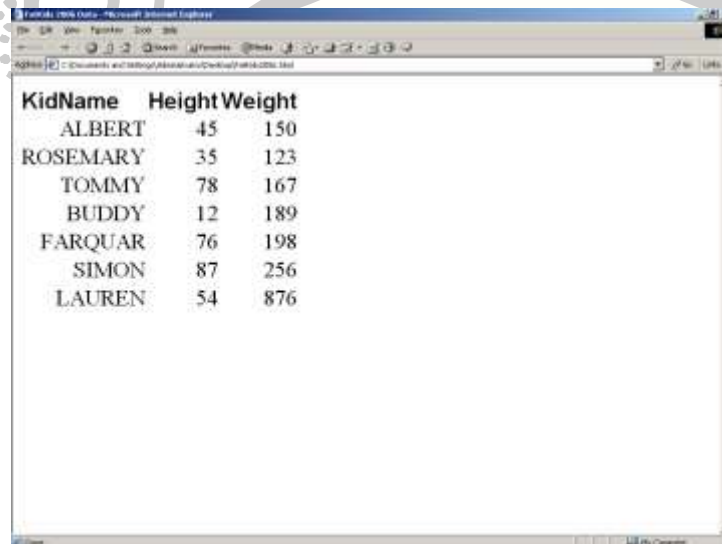


| KidName  | Height | Weight |
|----------|--------|--------|
| ALBERT   | 45     | 150    |
| ROSEMARY | 35     | 123    |
| TOMMY    | 78     | 167    |
| BUDDY    | 12     | 189    |
| FARQUAR  | 76     | 198    |
| SIMON    | 87     | 254    |
| LAUREN   | 54     | 876    |

Next, let's make each row of data appear as 24 point Courier font with the data right-justified. Here is the code to do that (I only show one data row):

```
<TR STYLE="font-family:CourierNew;font-size:24pt;text-align:right"><TD>ALBERT</TD><TD>45</TD><TD>150</TD></TR>
```

You'll notice that I removed the font-weight attribute since I don't want the text to be bold. Here is what the webpage looks like now:

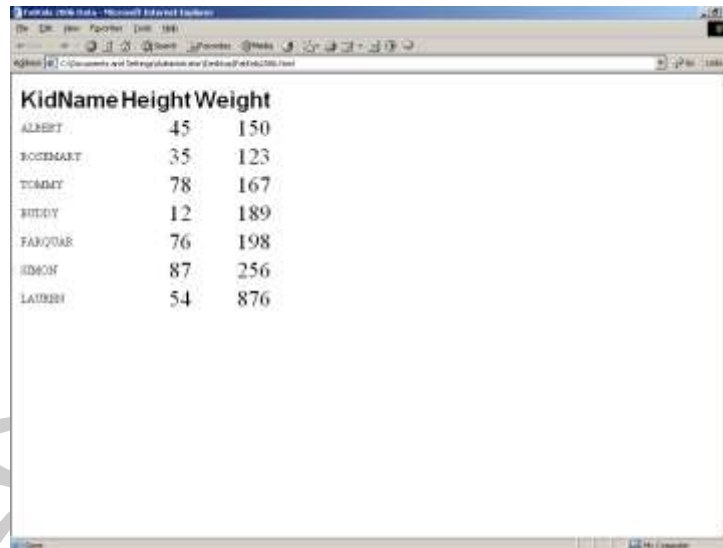


| KidName  | Height | Weight |
|----------|--------|--------|
| ALBERT   | 45     | 150    |
| ROSEMARY | 35     | 123    |
| TOMMY    | 78     | 167    |
| BUDDY    | 12     | 189    |
| FARQUAR  | 76     | 198    |
| SIMON    | 87     | 254    |
| LAUREN   | 54     | 876    |

As you can see, the name also appear right-justified, which is not exactly what we want. The reason that this occurred was that we put the STYLE= element inside each TR tag

which causes the entire row to be affected. If we had placed the STYLE= element inside the TD tags containing the height and weight information, we would have this code (again only one row is shown):

```
<TR>
  <TD>ALBERT</TD>
  <TD STYLE="font-family:CourierNew;font-size:24pt;text-align:right">45</TD>
  <TD STYLE="font-family:CourierNew;font-size:24pt;text-align:right">150</TD>
</TR>
```

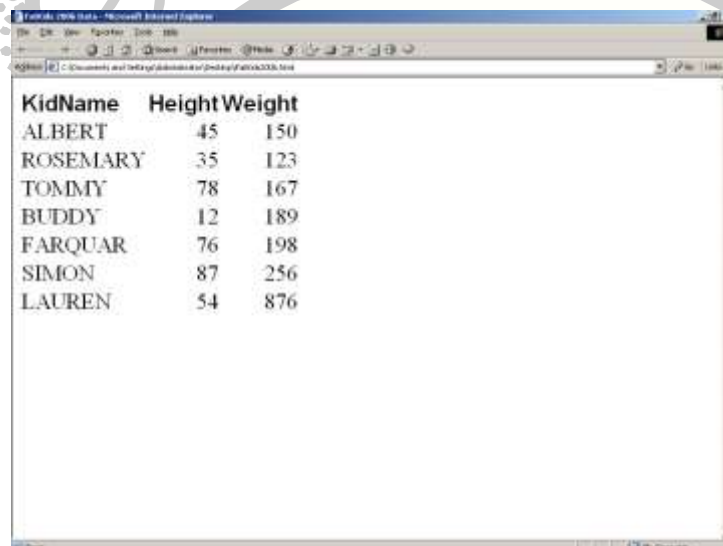


| KidName  | Height | Weight |
|----------|--------|--------|
| ALBERT   | 45     | 150    |
| ROSEMARY | 35     | 123    |
| TOMMY    | 78     | 167    |
| BUDDY    | 12     | 189    |
| FARQUAR  | 76     | 198    |
| SIMON    | 87     | 256    |
| LAUREN   | 54     | 876    |

Next, let's change the kid name to be Courier 24-point as well, but ensure that it'll be left-justified (here is part of one row):

```
<TD STYLE="font-family:CourierNew;font-size:24pt;text-align:left">ALBERT</TD>
```

and here is the result:



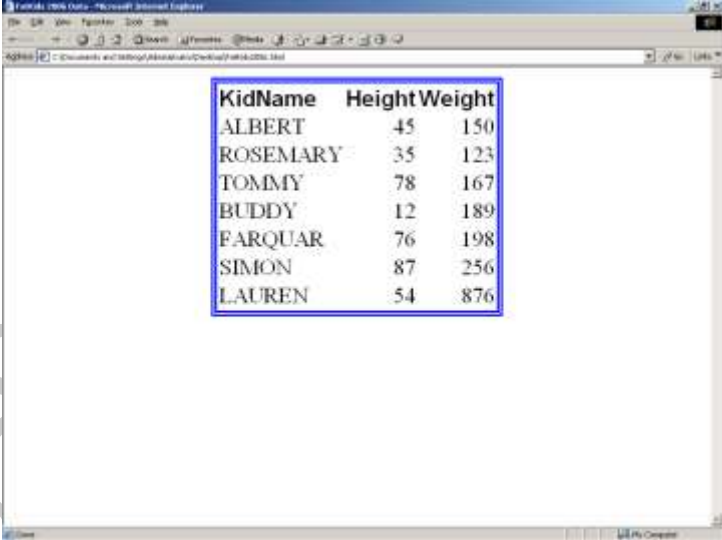
| KidName  | Height | Weight |
|----------|--------|--------|
| ALBERT   | 45     | 150    |
| ROSEMARY | 35     | 123    |
| TOMMY    | 78     | 167    |
| BUDDY    | 12     | 189    |
| FARQUAR  | 76     | 198    |
| SIMON    | 87     | 256    |
| LAUREN   | 54     | 876    |

Next, let's center the table in the Internet Explorer window, and put a double border around the entire table. Here is the code to do that (only the TABLE row is shown):

```
<TABLE ALIGN="CENTER" STYLE="border:6pt double blue">
```

Note that we aligned the table using the ALIGN= element. In the STYLE= element we used the border attribute. The border attribute is followed by three space-delimited options: the size of the border (6pt in this example), the type of line to draw around the border (double in this example) and the color of the border (blue in this example).

And here is the result:

A screenshot of a Microsoft Internet Explorer browser window. The browser's address bar shows a local file path. The main content area displays a table with a blue double border. The table has three columns: 'KidName', 'Height', and 'Weight'. The data rows are: ALBERT (45, 150), ROSEMARY (35, 123), TOMMY (78, 167), BUDDY (12, 189), FARQUAR (76, 198), SIMON (87, 256), and LAUREN (54, 876).

| KidName  | Height | Weight |
|----------|--------|--------|
| ALBERT   | 45     | 150    |
| ROSEMARY | 35     | 123    |
| TOMMY    | 78     | 167    |
| BUDDY    | 12     | 189    |
| FARQUAR  | 76     | 198    |
| SIMON    | 87     | 256    |
| LAUREN   | 54     | 876    |

You can go crazy with stylistic changes, but I think you get the idea. The ability to change font family, font size, colors, alignments, etc. we've presented here is very similar to how SAS ODS approaches displaying SAS procedure output in a PDF, RTF, HTML, or whatever output destination you choose.

# PART III

## Easing Our Way to Style

Three Base SAS procedures – PRINT, REPORT and TABULATE – allow you to change *some* of the ODS style elements, such as font size and background colors, directly from the SAS procedure code itself. This differs from the remaining procedures which require you to use PROC TEMPLATE to achieve the same style changes. In this section we discuss how to change the style elements for the PRINT procedure and leave it up to the interested reader to look into REPORT and TABULATE. We discuss PROC TEMPLATE in the next section.

Note that the procedures PRINT, REPORT and TABULATE produce output in a *tabular* format similar to using the <TABLE> HTML tag described in the previous section...keep this in mind when reading through this section.



## USING SAS ODS TO CHANGE PROC PRINT OUTPUT

If we were to run a PROC PRINT on the FatKids data, we would see the following output in the SAS Listing window:

```
proc print data=FatKids split="*" n;
  id KidName;
  var HeightInInches WeightInPounds;
  label KidName="Name of*Child"
        HeightInInches="Height*(inches)"
        WeightInPounds="Weight*(pounds)";
  title1 "2006 Fat Kid Data";
run;
```

2006 Fat Kid Data

| Name of<br>Child | Height<br>(inches) | Weight<br>(pounds) |
|------------------|--------------------|--------------------|
| ALBERT           | 45                 | 150                |
| ROSEMARY         | 35                 | 123                |
| TOMMY            | 78                 | 167                |
| BUDDY            | 12                 | 189                |
| FARQUAR          | 76                 | 198                |
| SIMON            | 87                 | 256                |
| LAUREN           | 54                 | 876                |

N = 7

No surprise, but this output is rather dull and is lacking in style. The PRINT procedure allows you to modify fonts, font sizes, colors and more for specific areas of the procedure output.

Note that these style changes do **NOT** work when using the LISTING output destination since it is simply text output. So, we will use the HTML output destination instead to produce a single webpage from our PRINT output. Here is the modified code and the corresponding output:

```
ods listing close;
ods html file="C:\FatKids_Data1.html";
proc print data=FatKids split="*" n;
  id KidName;
  var HeightInInches WeightInPounds;
  label KidName="Name of*Child"
        HeightInInches="Height*(inches)"
        WeightInPounds="Weight*(pounds)";
  title1 "2006 Fat Kid Data";
run;
ods html close;
ods listing;
```



| Name of Child | Height (inches) | Weight (pounds) |
|---------------|-----------------|-----------------|
| ALBERT        | 45              | 150             |
| ROSEMARY      | 35              | 123             |
| TOMMY         | 75              | 167             |
| BUDDY         | 12              | 99              |
| FARGUAR       | 70              | 148             |
| SIMON         | 87              | 254             |
| LAUREN        | 54              | 87              |

As you can see, the headings have a grey background with blue colored text. Let's change this PRINT output so that the headings have a white background and black colored italicized text. Here is the same PRINT code as above with a STYLE= option on the PROC PRINT line added in:

```
ods listing close;
ods html file="C:\FatKids_Data1.html";
proc print data=FatKids split="*" n
    style(HEADER)={font_style=italic foreground=black
background=white};
    id KidName;
    var HeightInInches WeightInPounds;
    label KidName="Name of*Child"
        HeightInInches="Height*(inches)"
        WeightInPounds="Weight*(pounds)";
    title1 "2006 Fat Kid Data";
run;
ods html close;
ods listing;
```

| Name of Child | Height (inches) | Weight (pounds) |
|---------------|-----------------|-----------------|
| ALBERT        | 45              | 150             |
| ROSEMARY      | 35              | 123             |
| TOMMY         | 75              | 167             |
| BUDDY         | 12              | 99              |
| FARGUAR       | 70              | 148             |
| SIMON         | 87              | 254             |
| LAUREN        | 54              | 87              |

Take note that we specified the option STYLE followed by the word HEADER in parentheses, followed by an equal sign followed by three style attributes similar to the STYLE attributes we introduced in the previous section on Cascading Style Sheets. Note that in the PROC PRINT STYLE= option, we use blanks to delimit the attributes rather than using semi-colons.

Note that the Name of Child column was not affected by our HEADER style change. This is because the variable KidName is actually specified on the ID line and not the VAR line. The ID line has its own STYLE command. Let's try our PRINT again as above, but let's fix the ID column to have a black background and white italicized text:

```
ods listing close;
ods html file="C:\FatKids_Data1.html";
proc print data=FatKids split="*" n
      style(HEADER)={font_style=italic foreground=black background=white};
  id KidName/style(HEADER)={font_style=italic foreground=white background=black};
  var HeightInInches WeightInPounds;
  label KidName="Name of*Child"
        HeightInInches="Height*(inches)"
        WeightInPounds="Weight*(pounds)";
  title1 "2006 Fat Kid Data";
run;
ods html close;
ods listing;
```

As you can see, we placed a STYLE option on the ID line after the slash. Here is the result:

| Name of Child | Height (inches) | Weight (pounds) |
|---------------|-----------------|-----------------|
| ALBERT        | 45              | 100             |
| ROSEMARY      | 35              | 123             |
| TOMMY         | 29              | 167             |
| BUDDY         | 12              | 199             |
| FARQUAR       | 76              | 199             |
| SIMON         | 37              | 256             |
| LAUREN        | 54              | 070             |
| N = 7         |                 |                 |

Next, let's change the height and weight values to be the Courier font:

```
ods listing close;
ods html file="C:\FatKids_Data1.html";
proc print data=FatKids split="*" n
      style(HEADER)={font_style=italic foreground=black background=white};
  id KidName/style(HEADER)={font_style=italic foreground=white background=black};
  var HeightInInches WeightInPounds/style(DATA)={font_face=Courier};
  label KidName="Name of*Child"
```

```

        HeightInInches="Height*(inches) "
        WeightInPounds="Weight*(pounds) ";
title1 "2006 Fat Kid Data";
run;
ods html close;
ods listing;

```

Here is the result:

| Name of Child | Height (inches) | Weight (pounds) |
|---------------|-----------------|-----------------|
| ALBERT        | 45              | 139             |
| ROSEMARY      | 35              | 123             |
| TOMMY         | 75              | 167             |
| BUDDY         | 12              | 599             |
| FAROUAR       | 76              | 138             |
| SIMON         | 87              | 254             |
| LAUREN        | 34              | 873             |

N=7

As you can see, the font in the cells has changed to Courier. Next, let's make the text N=7 at the bottom of the SAS PRINT output bold as well. Here is the code and the resulting webpage:

```

ods listing close;
ods html file="C:\FatKids_Data1.html";
proc print data=FatKids split="*" n
        style(HEADER)={font_style=italic foreground=black
background=white}
        style(N)={font_weight=bold};
id KidName/style(HEADER)={font_style=italic foreground=white
background=black};
var HeightInInches WeightInPounds/style(DATA)={font_face=Courier};
label KidName="Name of*Child"
        HeightInInches="Height*(inches) "
        WeightInPounds="Weight*(pounds) ";
title1 "2006 Fat Kid Data";
run;
ods html close;
ods listing;

```

| Name of Child | Height (inches) | Weight (pounds) |
|---------------|-----------------|-----------------|
| ALBERT        | 45              | 155             |
| ROSEMARY      | 35              | 123             |
| TOMMY         | 78              | 167             |
| BUDDY         | 12              | 59              |
| FARQUAR       | 76              | 138             |
| SIMON         | 87              | 254             |
| LAUREN        | 34              | 87              |
| N=7           |                 |                 |

By now, you get the idea, so here is a list of all of the PROC PRINT sections you can change with the STYLE option: BYLABEL, DATA, GRANDTOTAL, HEADER, N, OBS, OBSHEADER, TABLE, and TOTAL. You can look up the style attributes you can change in the PROC PRINT section of the Base SAS Procedures Guide.

The remaining two procedures, REPORT and TABULATE, also have the ability to change style attributes just like PRINT. Please refer to the Base SAS Procedures Guide to learn more about these two procedures.

Note that we have been using curly braces ({} ) to delimit the style attributes in this section. SAS also allows you to use brackets ([]) to delimit the style attributes. I decided to use curly braces in this section and the remaining sections as a reminder to the reader of the style attributes we encountered in the section on HTML and Cascading Style Sheets above.

## PART IV

### PROC TEMPLATE is Life!

You can smoke a cigarette if you want to...or not. It's your decision because you have ultimate control over your body.

And PROC TEMPLATE gives you ultimate control over how to create personalized styles for your procedure output.

Sure, the last two sentences were trite and completely unrelated, but you didn't have to read them...it was your decision, smarty-pants!



## Baby's Got BarrettsBlue Eyes

In Part I, when we tried the style named BarrettsBlue, the more curious of you may have wondered just where that style came from...and what style is used when BarrettsBlue is not used. As all of us SAS heads know, when we want to save a SAS dataset for later use, we can make a permanent SAS dataset; when we want to save a user-defined format for later use, we create a permanent format catalog; when we want save our graphs for later use in PROC REPLAY, say, we create a permanent graphics catalog. SAS comes with two pre-created permanent catalogs containing the SAS-defined templates. These catalogs are known as *item stores* in SAS ODS lingo. You can show the location of these two item stores -- and any item stores you define yourself -- by using the ODS PATH statement as shown below:

```
ods path show;
```

```
Current ODS PATH list is:
1. SASUSER.TEMPLAT (UPDATE)
2. SASHELP.TMPLMST (READ)
```

You'll note the very familiar two-level SAS names, SASUSER and SASHELP. The first specifies that the item store TEMPLAT is located in SASUSER while the second, TMPLMST, is located in SASHELP. Note that the first item store, SASUSER.TEMPLAT, can be updated by you. While the second, SASHELP.TMPLMST, is read-only.

So, what's in these item stores? To list the items in each item store, use the PROC TEMPLATE LIST statement like this:

```
proc template;
  list/store=sashelp.tmplmst;
run;
```

```
Listing of: SASHELP.TMPLMST
Path Filter is: *
Sort by: PATH/ASCENDING
```

| Obs                  | Path                           | Type        |
|----------------------|--------------------------------|-------------|
| <skipped some lines> |                                |             |
| 206                  | Base.Univariate                | Dir         |
| 207                  | Base.Univariate.BinPercents    | Table       |
| 208                  | Base.Univariate.Bins           | Table       |
| 209                  | Base.Univariate.ConfLimits     | Table       |
| 210                  | Base.Univariate.ExtObs         | Table       |
| 211                  | Base.Univariate.ExtVal         | Table       |
| 212                  | Base.Univariate.FitGood        | Table       |
| 213                  | Base.Univariate.FitParms       | Table       |
| 214                  | Base.Univariate.FitQuant       | Table       |
| 215                  | Base.Univariate.Frequency      | Table       |
| 216                  | Base.Univariate.LocCount       | Table       |
| 217                  | Base.Univariate.Location       | Table       |
| 218                  | Base.Univariate.Measures       | Table       |
| 219                  | Base.Univariate.Missings       | Table       |
| 220                  | Base.Univariate.Modes          | Table       |
| 221                  | <b>Base.Univariate.Moments</b> | <b>Link</b> |
| 222                  | Base.Univariate.Normal         | Table       |
| 223                  | Base.Univariate.PValue         | Link        |
| 224                  | Base.Univariate.Quantiles      | Table       |
| 225                  | Base.Univariate.Robustscale    | Table       |
| 226                  | Base.Univariate.Trim           | Table       |
| 227                  | Base.Univariate.Wins           | Table       |
| <skipped some lines> |                                |             |

What you see in the table above is all of the templates associated with the UNIVARIATE procedure. Recall from Part I that when we surrounded our PROC UNIVARIATE code with the ODS TRACE commands, we saw trace output like this:

```
Output Added:
-----
Name:          Moments
Label:         Moments
Template:   base.univariate.Moments
Path:         Univariate.WeightInPounds.Moments
-----
```

You'll note that this trace tells you that the template associated with the UNIVARIATE procedure's *moments* output is `base.univariate.Moments`. You'll also notice that this template is located in the item store `SASHELP.TMPLMST` as you can see on the previous page. You could probably guess that the word "base" refers to the Base SAS procedures, the "univariate" refers to the procedure itself, and the "Moments" refers to the output. Each SAS procedure has one or more templates associated with it, one for each output.

Let's try this again, but this time let's search for all of the templates associated with the SURVEYMEANS procedure in the SAS STAT module. Here is the PROC TEMPLATE LIST code you can use to see all of the templates associated with all output generated by the SURVEYMEANS procedure:

```
proc template;
  list stat.surveymeans/store=sashelp.tmplmst;
run;
```

```
Listing of: SASHELP.TMPLMST
Path Filter is: Stat.Surveymeans
Sort by: PATH/ASCENDING
```

| Obs | Path                                | Type   |
|-----|-------------------------------------|--------|
| 1   | Stat.SurveyMeans                    | Dir    |
| 2   | Stat.SurveyMeans.CLHeadMean         | Header |
| 3   | Stat.SurveyMeans.CLHeadRatio        | Header |
| 4   | Stat.SurveyMeans.CLHeadSum          | Header |
| 5   | Stat.SurveyMeans.ClassVarInfo       | Table  |
| 6   | Stat.SurveyMeans.Column             | Column |
| 7   | Stat.SurveyMeans.Df                 | Link   |
| 8   | Stat.SurveyMeans.Domain             | Table  |
| 9   | Stat.SurveyMeans.EqualSign          | Column |
| 10  | Stat.SurveyMeans.Factoid            | Link   |
| 11  | Stat.SurveyMeans.Header             | Header |
| 12  | Stat.SurveyMeans.LlSideCLHeadMean   | Header |
| 13  | Stat.SurveyMeans.LlSideCLHeadSum    | Header |
| 14  | Stat.SurveyMeans.LCLMean            | Header |
| 15  | Stat.SurveyMeans.LCLSum             | Header |
| 16  | Stat.SurveyMeans.Probt              | Link   |
| 17  | Stat.SurveyMeans.Ratio              | Table  |
| 18  | Stat.SurveyMeans.StackingDomain     | Table  |
| 19  | Stat.SurveyMeans.StackingStatistics | Table  |
| 20  | Stat.SurveyMeans.StackingStrataInfo | Table  |
| 21  | Stat.SurveyMeans.Statistics         | Table  |



|    |                                   |        |
|----|-----------------------------------|--------|
| 22 | Stat.SurveyMeans.StdErr           | Link   |
| 23 | Stat.SurveyMeans.StrataInfo       | Table  |
| 24 | Stat.SurveyMeans.U1SideCLHeadMean | Header |
| 25 | Stat.SurveyMeans.U1SideCLHeadSum  | Header |
| 26 | Stat.SurveyMeans.UCLMean          | Header |
| 27 | Stat.SurveyMeans.UCLSum           | Header |
| 28 | Stat.SurveyMeans.tValue           | Link   |

As you can see in both the UNIVARIATE and SURVEYMEANS list of templates, there is a single line labeled Dir under the Types heading. Dir stands for Directory. Item stores are arranged in an hierarchical fashion meaning that there is top-level directory and items below it. (Ignore the remaining Types for now.)

That's all well and good, but where does BarrettsBlue come from? Well, if you scanned the entire PROC TEMPLATE LIST, you'll notice a directory labeled Styles. Under this directory, there is Styles.BarrettsBlue as well as the default style Styles.Default, which is used when you do not specify a style like BarrettsBlue. As a reminder, here is the code we used in Part I:

```
ods pdf style=BarrettsBlue file="C:\FatKids_Analysis2.pdf";
```

You can specify the STYLE= option for PDF, HTML and any other output destination except for LISTING and OUTPUT. Note that LISTING is plain-text and is not affected by colors and fonts and OUTPUT is used to create a SAS dataset which is equally not affected by colors and fonts.

Take note that there is no mention of either SASUSER.TEMPLAT or SASHELP.TMPLMST when referring to Styles.BarrettsBlue. That's because all item stores are searched until a match is found. The output of the ODS PATH SHOW command lists the search path in order.

So, how does knowing all of this help you out? Well, with PROC TEMPLATE you have the choice of either creating your own templates from scratch, or using a pre-defined template and just modifying the styles you are concerned with. The rest of this chapter focuses on modifying pre-defined templates.

Recall from Part III that we modified styles in the PRINT procedure of the FatKids data to change the header fonts, colors, etc. Let's try that example again, but we will use the default style Styles.Default and modify it. The first thing we have to know is: what does Styles.Default look like from a SAS code standpoint? To see the PROC TEMPLATE code behind Styles.Default, issue the following command:

```
proc template;
  source Styles.Default;
run;
```

We won't show the code here because there is a lot to it, but be aware that the HEADER and DATA that we modified in the PRINT procedure in Part III are the same names you use in your PROC TEMPLATE. Here is the PROC TEMPLATE code to mimic the PRINT procedure from Part III:

```
proc template;
```

```

define style MyNewStyle;
  parent=Styles.Default;
  style HEADER from HEADER/font_style=italic
                    foreground=black
                    background=white;
  style DATA from DATA/font_face=CourierNew;
  style NOTECONTENT from NOTECONTENT/font_weight=Bold;
end;
run;

```

As with other SAS procedures, you start off with PROC TEMPLATE. Next, you issue a DEFINE STYLE command followed by the name of your new style, here we called it MyNewStyle. Next, we tell PROC TEMPLATE that we are going to inherit the style information from Styles.Default so that we do not have to define all of the styles. Next, we change three styles. The first two are familiar: HEADER and DATA. These two have the same meaning as the styles in the PRINT procedure from Part III. Note that in order to change the style for the HEADERS, we let PROC TEMPLATE know that we are inheriting all of the default header information from Styles.Default by using the syntax: style header **from header**. Then, we go ahead and change the header font and colors after the forward slash in a similar way to the PRINT procedure.

Next, the NOTECONTENT is associated with the N=7 information from the PRINT procedure in PART III. I determined that I needed to modify the NOTECONTENT by looking at the CLASS= attribute in the HTML that is generated by the ODS HTML output file. Here is what the N=7 row looks like at the bottom of the HTML file FatKids\_Data1.html:

```
<td class="l NOTECONTENT" colspan="3">N = 7</td>
```

The lowercase letter “l” means to left-justify the text N=7 in the table data field. Take note of the NOTECONTENT class. This indicates to SAS what the style is called and how we knew what style to modify in the PROC TEMPLATE code.

Here is the ODS HTML code that uses MyNewStyle:

```

ods listing close;
ods html style=MyNewStyle file='C:\FatKids_Data1.html';
proc print data=FatKids split="*" n;
  id KidName;
  var HeightInInches WeightInPounds;
  label KidName="Name of*Child"
        HeightInInches="Height*(inches) "
        WeightInPounds="Weight*(pounds) ";
  title1 "2006 Fat Kid Data";
run;
ods html close;
ods listing;

```

| Name of Child | Height (inches) | Weight (pounds) |
|---------------|-----------------|-----------------|
| ALBERT        | 49              | 150             |
| ROSEMARY      | 39              | 123             |
| TOMMY         | 78              | 167             |
| BUDDY         | 52              | 180             |
| FARGUAR       | 70              | 188             |
| SEMON         | 87              | 256             |
| LAUREN        | 54              | 370             |
| N=7           |                 |                 |

As you can see, the headers all have a white background, but the results from our PRINT procedure in Part III had a black background for the text “Name of Child”. Based on the discussion thus far, there is no way to change the background color of just a single column of data. We will, of course, show you how to get around this later on.

Now, if you are happy with the style MyNewStyle you have created, you probably want to save it in your own item store. Note that by default any new style is stored in SASUSER.TEMPLAT in the Styles “directory”. If you want to create your own item store to contain just your own styles, you can define a SAS Libname to a location of your choice and store your style there. Here is an example of how to create your own item store and place your new style MyNewStyle in a directory called MyStyles:

```
libname MyItmStr "C:\";
run;

ods path (PREPEND) MyItmStr.MyStyles(UPDATE);
ods path show;

proc template;
  define style MyStyles.MyNewStyle;
    parent=Styles.Default;
    style HEADER from HEADER/font_style=Italic
                      foreground=black
                      background=white;
    style DATA from DATA/font_face=CourierNew;
    style NOTECONTENT from NOTECONTENT/font_weight=Bold;
  end;
run;

proc template;
  list /store=MyItmStr.MyStyles;
run;
```

Note that we set up a SAS Libname to the location of the item store. Item stores have an extension of **.sas7bitm**. Next, recall that we said that SASUSER.TEMPLAT and

SASHELP.TMPLMST are SAS-defined default item stores. As we explained before, these two item stores are searched in order based on the results of issuing an ODS PATH SHOW command. In order for our new style to be found in the search path when we are creating a brand new item store, we must add our item store to the search path. In this case, we are prepending our item store to the search path so that our new style is found quickly and SAS does not have to search through all of the item stores to find it. Next, we enter in the libname followed by a dot followed by the work MyStyles. MyStyles is the name of our directory and any styles we create will appear under that directory. Next, we ensure that we can update our new item store by entering the keyword UPDATE in parentheses. Next, we issue an ODS PATH SHOW command to see if the path has been updated. Here are the results of this command in the SAS Log file:

```
Current ODS PATH list is:
```

1. **MYITMSTR.MYSTYLES (UPDATE)**
2. SASUSER.TEMPLAT (UPDATE)
3. SASHELP.TMPLMST (READ)

Next, we rerun our PROC TEMPLATE only this time we tell the DEFINE STYLE statement where to place our new style: MyStyles.MyNewStyle. If you list your item store, this is what you will see:

```
proc template;
  list/store=MyItmStr.MyStyles;
run;
```

```
Listing of: MYITMSTR.MYSTYLES
Path Filter is: *
Sort by: PATH/ASCENDING
```

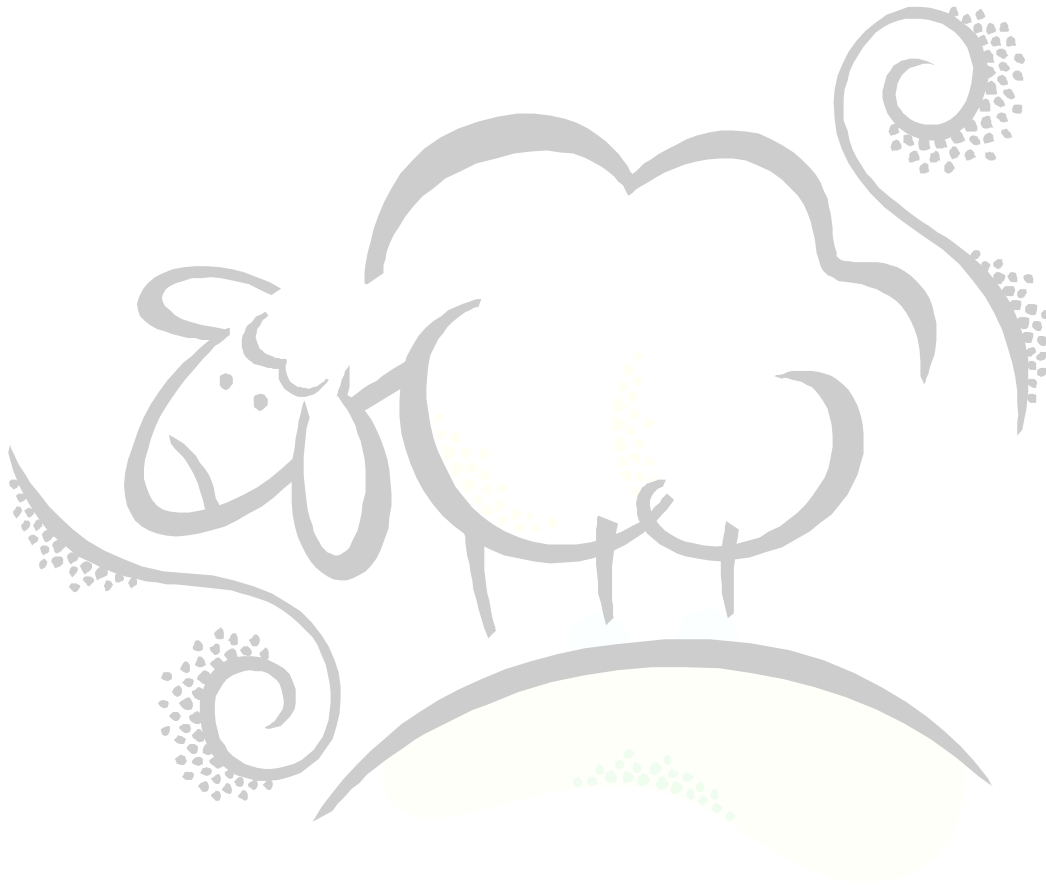
| Obs | Path                | Type  |
|-----|---------------------|-------|
| 1   | MyStyles            | Dir   |
| 2   | MyStyles.MyNewStyle | Style |

If you want to see the PROC TEMPLATE source code for your style MyNewStyle, you issue this code and your PROC TEMPLATE source code will appear in the SAS Log file:

```
proc template;
  source MyStyles.MyNewStyle/store=MyItmStr.MyStyles;
run;
```

```
1520 proc template;
1521 source MyStyles.MyNewStyle/store=MyItmStr.MyStyles;
NOTE: Path 'MyStyles.MyNewStyle' is in: MYITMSTR.MYSTYLES.
define style MyStyles.MyNewStyle / store = MYITMSTR.MYSTYLES;
  parent = Styles.Default;
  style HEADER from HEADER /
    background = white
    foreground = black
    font_style = Italic;
  style DATA from DATA /
```

```
font_face = CourierNew;  
style NOTECONTENT from NOTECONTENT /  
font_weight = Bold;  
end;  
1522 run;  
NOTE: PROCEDURE TEMPLATE used (Total process time):  
real time      0.01 seconds  
cpu time       0.02 seconds
```



## The Author's a Big Fat Liar!

Recall in Part II we briefly introduced PROC REPORT and then said that PROC TEMPLATE and PROC REPORT had many things in common. The previous section focused on just styles – my little white lie to you – but PROC TEMPLATE gives you more than just the ability to change fonts and colors, it allows you to define columns just like PROC REPORT. In fact, PROC TEMPLATE goes even further in that you can define headers, footers, and can even change the color of a cell based on the value of the data. Recall I mentioned that you should consider SAS procedure output as just a table of data. In fact, in order to use PROC TEMPLATE to its fullest, you begin by defining a table and then within the table you define columns, headers, footers, styles, etc.

Let's design a template for the FatKid data:

```
proc template;

/* Here we start to define the table definition */
define table FatKidTable;

/* Define the columns...these are not the same as the */
/* column names in the dataset. */
column tFatKidName tFatKidHeight tFatKidWeight;

/* Let PROC TEMPLATE know what the TITLE1 is called */
header hdr1;

/* Define TITLE1 */
define header hdr1;
text "2006 Fat Kid Data";
style=header{font_size=4 just=left};
end;

/* Define the column representing the KidName */
define tFatKidName;

/* Define the header of this column */
define header hdrName;
text "Name of Child";
style=header{font_size=2 font_weight=bold};
just=center;
end;

id=on; /* Force this column to repeat when there are several pages */
just=left;
vjust=middle; /* vjust is the vertical justification */
style=data{font_size=2 background=grey font_weight=bold};
header=hdrName;

end;

/* Define the column representing the Height */
define tFatKidHeight;

/* Define the header of this column */
```

```

define header hdrHt;
  text "Height*(inches)";
  style=header{font_size=2 font_weight=bold};
  just=center;
  split="*";
end;

id=off;
just=right;
vjust=middle;
style=data{font_size=2};
header=hdrHt;

end;

/* Define the column representing the Weight */
define tFatKidWeight;

  /* Define the header of this column */
  define header hdrWt;
    text "Weight*(pounds)";
    style=header{font_size=2 font_weight=bold};
    just=center;
    split="*";
  end;

  id=off;
  just=right;
  vjust=middle;
  header=hdrWt;
  style=data{font_size=2};

  /* Create a CellStyle that will turn the font bold and colored red */
  /* if WeightInPounds is more than 500. */
  cellstyle _val_>500 as data{font_size=2
                                font_weight=bold
                                foreground=red};

end;

end;
run;

```

Here is the code to use our newly defined template. Note that the COLUMNS associate a column in the table definition above with the columns we have in the dataset FatKids:

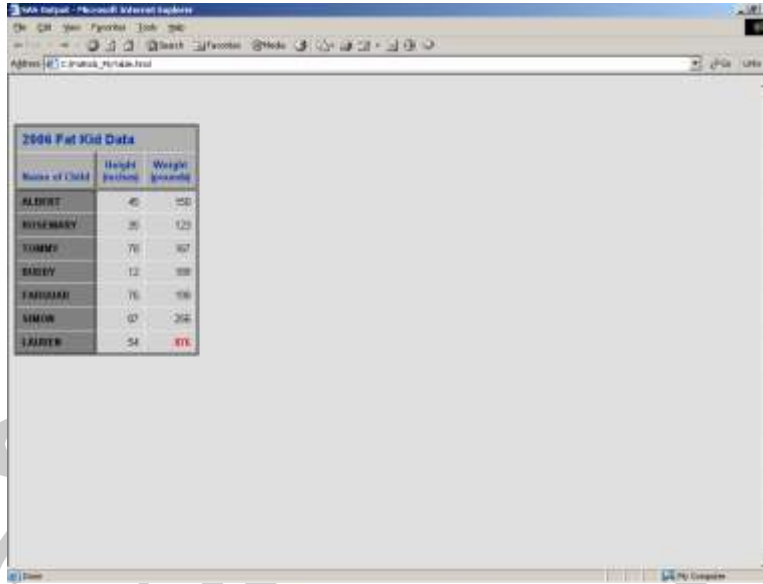
```

ods listing close;
ods html file="C:\FatKids_MyTable.html";
data _null_;
  set FatKids;
  file print ods=(
    template="FatKidTable"
    columns=(
      tFatKidName=KidName
      tFatKidHeight=HeightInInches

```

```
        tFatKidWeight=WeightInPounds
    )
);
put _ods_;
run;
ods html close;
ods listing;
```

Here are the results:



| Name of Child | Height (inches) | Weight (pounds) |
|---------------|-----------------|-----------------|
| ALBERT        | 45              | 50              |
| ROSEMARY      | 26              | 123             |
| TOMMY         | 76              | 67              |
| BARRY         | 12              | 88              |
| FRANKIE       | 76              | 106             |
| SIMON         | 07              | 264             |
| LAUREN        | 54              | 87              |

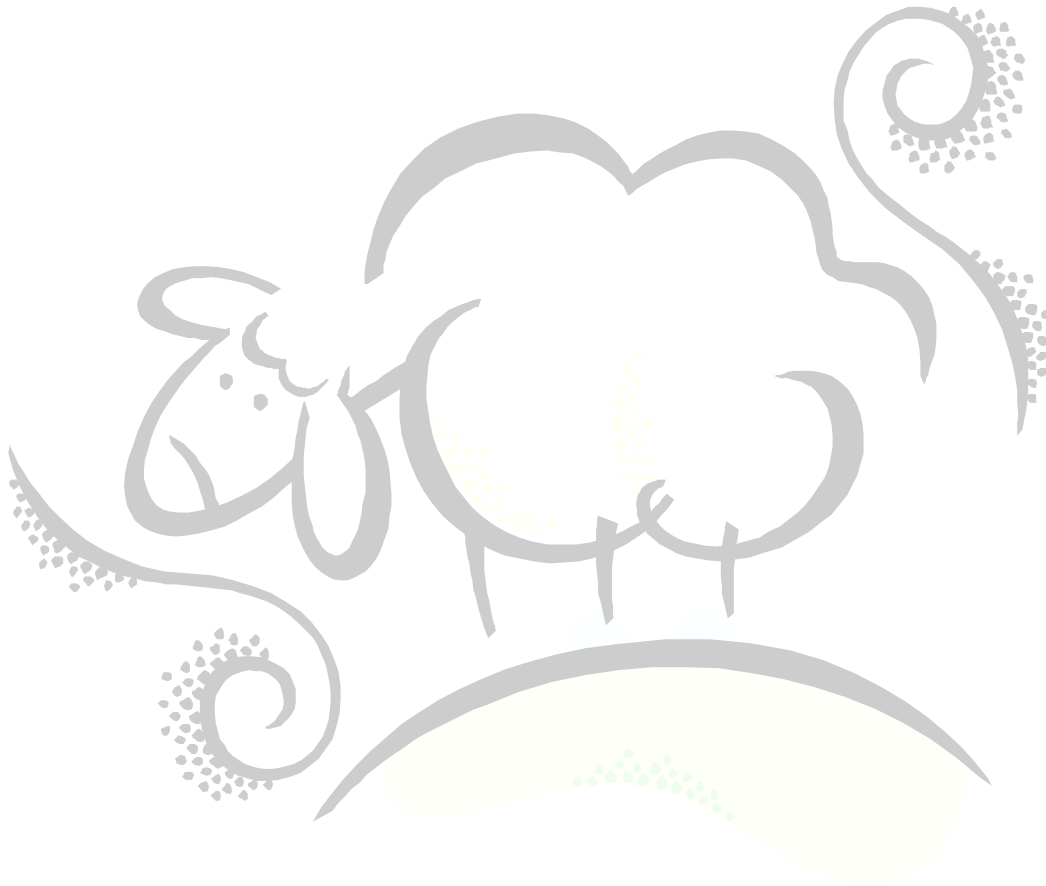
It may not surprise you that there are many more things you can do with PROC TEMPLATE than we discussed in this little chapter. Consult the SAS Output Delivery System manual for much more!!



# PART V

## Tag! You're It!

Besides creating style and table definitions, `PROC TEMPLATE` allows you to create your own *tagset*. A tagset enables you to create your own markup language like XML or HTML, or as you will see below, you can output anything you like.



## Quick Intro to Tagsets

The SAS Output Delivery System comes with several built-in tagsets, as you can see from the code below:

```
proc template;  
  list Tagsets;  
run;
```

```
Listing of: SASHELP.TMPLMST  
Path Filter is: Tagsets  
Sort by: PATH/ASCENDING
```

| Obs | Path                         | Type   |
|-----|------------------------------|--------|
| 1   | Tagsets                      | Dir    |
| 2   | Tagsets.Accessible           | Tagset |
| 3   | Tagsets.Cascading_stylesheet | Tagset |
| 4   | Tagsets.Chtml                | Tagset |
| 5   | Tagsets.Colorlatex           | Tagset |
| 6   | Tagsets.Config_debug         | Tagset |
| 7   | Tagsets.Csv                  | Tagset |
| 8   | Tagsets.Csvall               | Tagset |
| 9   | Tagsets.Csvbyline            | Tagset |
| 10  | Tagsets.Default              | Tagset |
| 11  | Tagsets.Docbook              | Tagset |
| 12  | Tagsets.Event_map            | Tagset |
| 13  | Tagsets.ExcelXP              | Tagset |
| 14  | Tagsets.Graph                | Tagset |
| 15  | Tagsets.Graph_rtf            | Tagset |
| 16  | Tagsets.Html4                | Tagset |
| 17  | Tagsets.Htmlcss              | Tagset |
| 18  | Tagsets.Htmlpanel            | Tagset |
| 19  | Tagsets.Imode                | Tagset |
| 20  | Tagsets.Latex                | Tagset |
| 21  | Tagsets.MSOffice2k           | Tagset |
| 22  | Tagsets.Mvshhtml             | Tagset |
| 23  | Tagsets.Namedhtml            | Tagset |
| 24  | Tagsets.Odsapp               | Tagset |
| 25  | Tagsets.Odsgraph             | Tagset |
| 26  | Tagsets.Odsstyle             | Tagset |
| 27  | Tagsets.Odsxrpcs             | Tagset |
| 28  | Tagsets.Phtml                | Tagset |
| 29  | Tagsets.Pmml                 | Tagset |
| 30  | Tagsets.Pyx                  | Tagset |
| 31  | Tagsets.Rtf                  | Tagset |
| 32  | Tagsets.SASReport10          | Tagset |
| 33  | Tagsets.SASReport11          | Tagset |
| 34  | Tagsets.SASReport12          | Link   |
| 35  | Tagsets.SASReport13          | Link   |
| 36  | Tagsets.SASReport14          | Link   |
| 37  | Tagsets.SASReport15          | Link   |
| 38  | Tagsets.SASReport_html       | Link   |
| 39  | Tagsets.SASReport_html1      | Link   |
| 40  | Tagsets.Sasreport_html10     | Tagset |
| 41  | Tagsets.Sasreport_html11     | Tagset |
| 42  | Tagsets.Sasxmacc             | Tagset |
| 43  | Tagsets.Sasxmacc2002         | Tagset |
| 44  | Tagsets.Sasxmacc2003         | Tagset |

|    |                         |        |
|----|-------------------------|--------|
| 45 | Tagsets.Sasxmdtd        | Tagset |
| 46 | Tagsets.Sasxmiss        | Tagset |
| 47 | Tagsets.Sasxnmis        | Tagset |
| 48 | Tagsets.Sasxmns         | Tagset |
| 49 | Tagsets.Sasxmodm        | Tagset |
| 50 | Tagsets.Sasxmog         | Tagset |
| 51 | Tagsets.Sasxmoh         | Tagset |
| 52 | Tagsets.Sasxmoim        | Tagset |
| 53 | Tagsets.Sasxmor         | Tagset |
| 54 | Tagsets.Sasxmphp        | Tagset |
| 55 | Tagsets.Sasxmxd         | Tagset |
| 56 | Tagsets.Short_map       | Tagset |
| 57 | Tagsets.Simplelatex     | Tagset |
| 58 | Tagsets.Statgraph       | Tagset |
| 59 | Tagsets.Style_display   | Tagset |
| 60 | Tagsets.Style_popup     | Tagset |
| 61 | Tagsets.Supermap        | Tagset |
| 62 | Tagsets.Tablesonlylatex | Tagset |
| 63 | Tagsets.Text_map        | Tagset |
| 64 | Tagsets.Tpl_style_list  | Tagset |
| 65 | Tagsets.Tpl_style_map   | Tagset |
| 66 | Tagsets.Troff           | Tagset |
| 67 | Tagsets.Wml             | Tagset |
| 68 | Tagsets.Wmlolist        | Tagset |
| 69 | Tagsets.XMLcdisc        | Tagset |
| 70 | Tagsets.Xhtml           | Tagset |
| 71 | Tagsets.sasFMT          | Tagset |
| 72 | Tagsets.sasXML          | Tagset |
| 73 | Tagsets.sasioXML        | Tagset |

We can create our own tagset by inheriting the tagset information from one of the previously defined tagsets above. In the code below, we inherit from the CSV tagset and change how the tagset code works so that given an incoming dataset, the output is a series of Oracle INSERT INTO statements.

```
proc template;
define tagset OracleInserts; /* OracleInserts is the name of our tagset */
  parent=Tagsets.Csv;

  /* Define TBLNAME as an incoming macro variable that defines the */
  /* name of the table. */
  mvar tblname;

  /* Initialize two variables, $flag which indicates whether we can */
  /* write the row and $totcols which is the total number of columns */
  /* in the dataset. */
  define event initialize;
    eval $flag 0;
    eval $totcols 0;
  end;

  /* We undefine the event HEADER so that we don't print the column */
  /* headers to our file...Oracle wouldn't like this!! */
  define event header;
  end;

  /* Use the COL_HEADER_LABEL event to count the total number of */
  /* columns in the dataset...this event occurs before the data rows */
  /* are processed by PROC TEMPLATE. */
end;
```

```

define event col_header_label;
  eval $totcols $totcols+1;
end;

/* For each row of data, we output the text INSERT INTO tblname VALUES ( */
/* only if our $flag is not zero. We finish the row event by outputting */
/* the final right parenthesis and the slash for Oracle. */
/* Note that we start counting of the current column at zero. */
define event row;

start:
  eval $curcol 0;
  put 'INSERT INTO ' tblname ' VALUES('/if $flag;

finish:
  put ")" nl "/" nl/if $flag;

end;

/* Write out the actual data values by surrounding text with quotes and */
/* leaving numbers along. Dates will also be surrounded by quotes. */
define event data;

start:
  eval $curcol $curcol+1;
  put 'INSERT INTO ' tblname ' VALUES('/if ^$flag;
  put "' VALUE '"/if cmp("string",type);
  put VALUE/if !cmp("string",type);

  eval $diff $curcol-$totcols;
  put ', '/if $diff;

finish:
  eval $flag 1;

end;

end;
run;

```

Here is the SAS code used to create our INSERT statements. Notice how we define the SAS macro variable TBLNAME to be the name of the table we are inserting data into:

```

%let tblname=FATTY_TABLE;
ods markup file="C:\Fatty_Table_Inserts.sql" tagset=OracleInserts;
proc print data=FatKids noobs;
  var KidName HeightInInches WeightInPounds;
run;
ods markup close;

```

And this is what the output looks like:

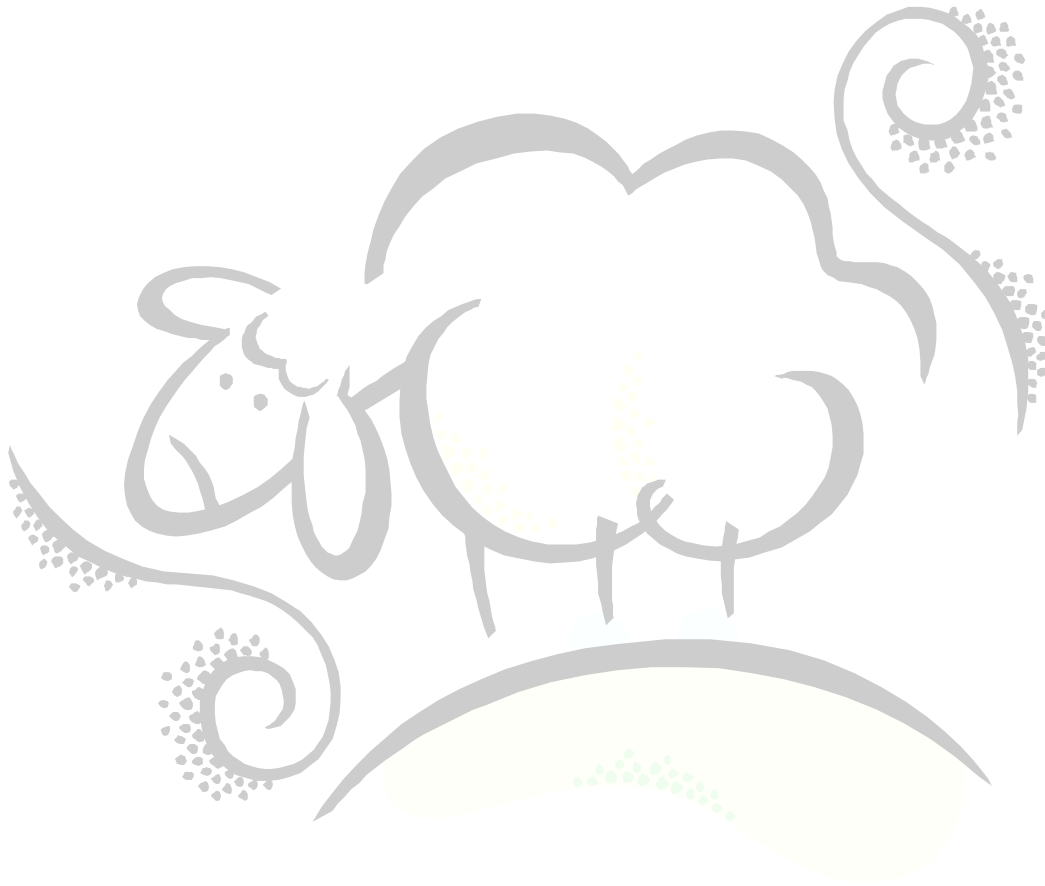
```

INSERT INTO FATTY_TABLE VALUES ("ALBERT",45,150)
/
INSERT INTO FATTY_TABLE VALUES ("ROSEMARY",35,123)
/
INSERT INTO FATTY_TABLE VALUES ("TOMMY",78,167)
/
INSERT INTO FATTY_TABLE VALUES ("BUDDY",12,189)

```

```
/  
INSERT INTO FATTY_TABLE VALUES ("FARQUAR",76,198)  
/  
INSERT INTO FATTY_TABLE VALUES ("SIMON",87,256)  
/  
INSERT INTO FATTY_TABLE VALUES ("LAUREN",54,876)  
/
```

Note that you can use the CSV tagset to create comma-separated values from your SAS dataset.



## PART VI

### Zen ODS!

Two new ODS features available in SAS Version 9 are ODS LAYOUT and ODS REGION. These two features allow you put more than one report or graph on a single page in a similar way to PROC GREPLAY except it's easier! We like easy!

## Quick Example of ODS LAYOUT and ODS REGION

Below is an example SAS program that places four graphical images on a single page. The graph appears at the end.

```
options center nodate nonumber;
run;

data FatKids;
infile cards;
input @1 KidName $char8.
      @10 HeightInInches 2.
      @15 WeightInPounds 3.;
      FattyIndex=WeightInPounds/HeightInInches;
cards;
ALBERT 45 150
ROSEMARY 35 123
TOMMY 78 167
BUDDY 12 189
FARQUAR 76 198
SIMON 87 256
LAUREN 54 876
;
run;

goptions reset=symbol;
run;

title1 "Fat Kid Data and Graphs";
title2;

ods listing close;
ods pdf style=BarrettsBlue file="c:\temp\FatKidData.pdf" notoc;

ods layout start width=8.5in height=11in;

ods region x=.25in y=.25in width=8in height=2.5in;

proc print data=FatKids label split="*" noobs;
var KidName HeightInInches WeightInPounds FattyIndex;
label KidName="Kid Name"
      HeightInInches="Height*(inches)"
      WeightInPounds="Weight*(pounds)"
      FattyIndex="Fatty*Index";
run;

symbol1 interpol=none value=dot height=.5in;

ods region x=.25in y=2.5in width=7in height=3.5in;
proc gplot data=FatKids;
plot HeightInInches*WeightInPounds/frame grid;
title1;
title2;
run;
quit;
```



```
ods region x=.25in y=6.25in width=3.5in height=3.8in;  
proc gchart data=FatKids;  
  vbar KidName/sumvar=HeightInInches frame;  
  title1;  
  title2;  
run;  
quit;
```

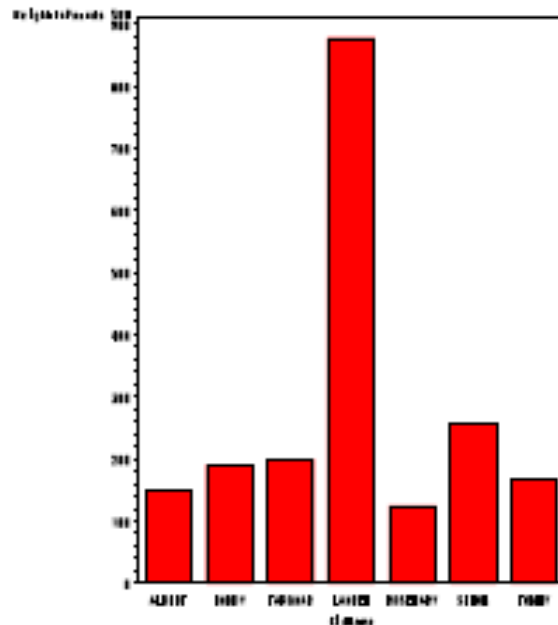
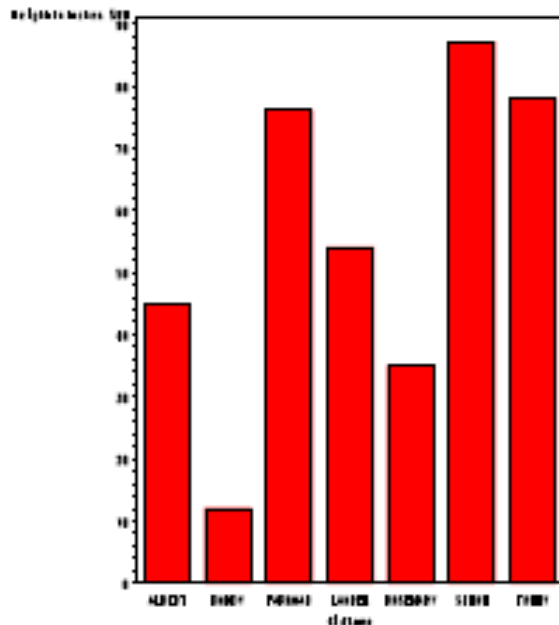
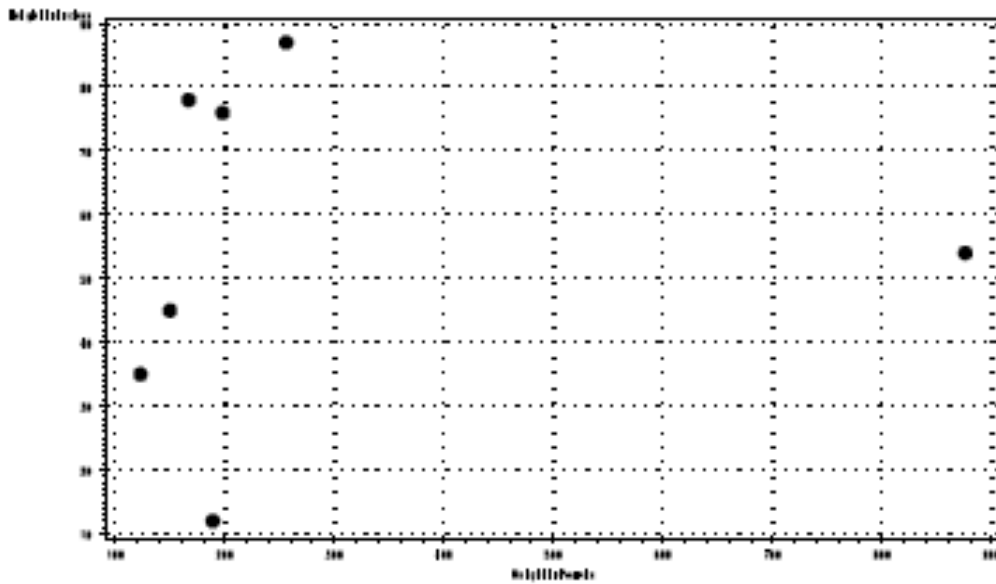
```
ods region x=4.25in y=6.25in width=3.5in height=3.8in;  
proc gchart data=FatKids;  
  vbar KidName/sumvar=WeightInPounds frame;  
  title1;  
  title2;  
run;  
quit;
```

```
ods layout end;  
ods pdf close;  
ods listing;
```



### Fat Kid Data and Graphs

| Kid Name | Height (inches) | Weight (pounds) | Fatty Index |
|----------|-----------------|-----------------|-------------|
| ALBERT   | 45              | 150             | 3.3333      |
| ROSEMARY | 35              | 123             | 3.5143      |
| TOMMY    | 78              | 167             | 2.1410      |
| BUDDY    | 12              | 189             | 15.7500     |
| FARQUAR  | 76              | 198             | 2.6053      |
| SIMON    | 87              | 256             | 2.9425      |
| LAUREN   | 54              | 876             | 16.2222     |





### Support sheepsqueezers.com

If you found this information helpful, please consider supporting [sheepsqueezers.com](http://sheepsqueezers.com). There are several ways to support our site:

- Buy me a cup of coffee by clicking on the following link and donate to my PayPal account: [Buy Me A Cup Of Coffee?](http://BuyMeACupOfCoffee.com).
- Visit my Amazon.com Wish list at the following link and purchase an item: <http://amzn.com/w/3OBK1K4EIWIR6>

Please let me know if this document was useful by e-mailing me at [comments@sheepsqueezers.com](mailto:comments@sheepsqueezers.com).