

Version 9 ODS - Intermediate

Prepared by



International SAS® Training and Consulting

Destiny Corporation
100 Great Meadow Rd Suite 601
Wethersfield, CT 06109-2379
Phone: (860) 721-1684 1-800-7TRAINING
Fax: (860) 721-9784
Email: info@destinycorp.com
Web: www.destinycorp.com
Copyright 2003

Introduction to Templates

As stated above, ODS output is really the combination of two separate elements: the output of the Procedure combined with the template structure.

By definition, a template is “an abstract description of how output should appear when it is formatted”.

The idea of a template is further clarified by noting that “templates describe several characteristics of the output, including headers, column ordering, style information, justification, and formats”.

We can categorize templates into two types:

- Table templates
- Style templates

The table template defines the table structure to hold the output. Recall from the ODS Trace statement that a single procedural output can involve several table templates.

Style templates, in contrast, determine the style of the overall page: background color, font, face, etc.

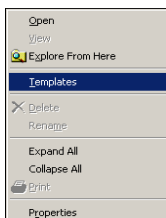
Inspecting Templates

The two templates work together, but have different functionalities.

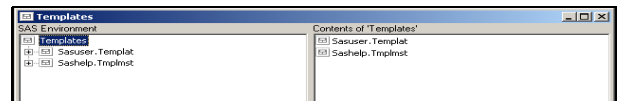
Both templates may be examined, edited, or created from scratch.

To see a list of ODS templates, right click on the Results folder of the Results window in the Windowing Environment.

A fly-out menu is displayed with ‘Templates’ enabled.



Click on Templates, and the Templates window is displayed.

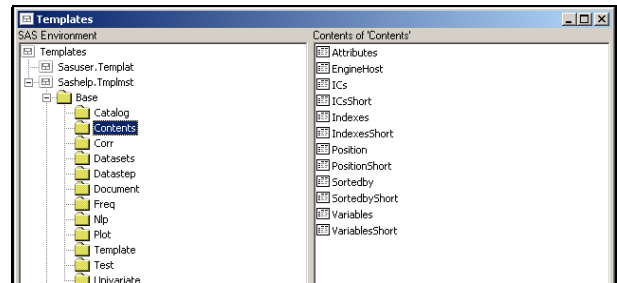


Notice that the Templates Folder has two subfolders:

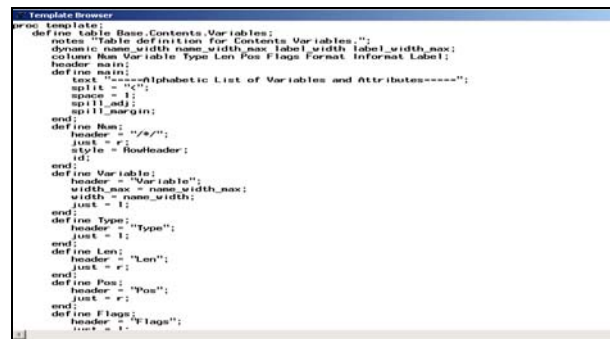
- Sasuser.Templat
- Sashelp.Tmplmst.

Both folders contain templates but their use is slightly different.

The Sashelp.Tmplmst folder holds the templates created by SAS Institute as part of Base SAS. Click on the folder to display its organization and identify templates by name and use.

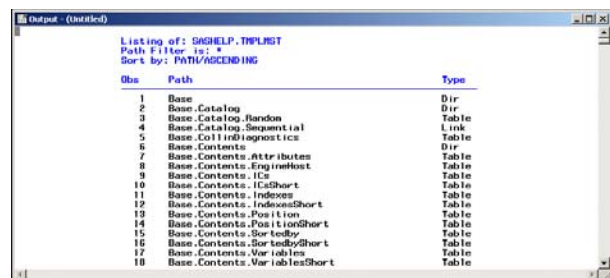
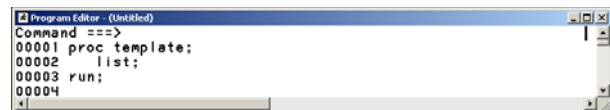


Double clicking on any template in the right pane will open a Template Browser window displaying the syntax of the table template.



In environments other than Windows, information about templates is available through program submission.

To inspect a list of all templates, submit the following syntax.



To be more specific, give additional information about the template by name.

Use any full-path or partial-path information such as:

- Base
- base.catalog
- base.catalog.random

Consider the following syntax, which shows the template syntax in the Log window.

The screenshot shows two windows. The top window is the 'Program Editor - (Untitled)' with the following code:

```
Command ==>
00001 proc template;
00002   path sashelp.tmplmst;
00003   source base.catalog.random;
00004 run;
```

The bottom window is the 'Log - (Untitled)' showing the execution of the template. It includes a note: 'To print the engine/host specific contents listing by entry for as well as internal information about each entry not published to dynamic column width obname width_max desc width desc_max lib column num obname type tombstone level crdate moddate desc pagesize bytemnt blocksizes pages modified;'. It also shows the definition of a table named 'Base.Catalog.Random' with various attributes and a header.

Here is another example that illustrates the capabilities of this syntax.

The screenshot shows the 'Program Editor - (Untitled)' with the following code:

```
Command ==>
00001 proc template;
00002   source base.datasets.members /
00003   store = sashelp.tmplmst;
00004 run;
00005
```

This program produces the following Log results.

The screenshot shows the 'Log - (Untitled)' window. It displays the execution of the template, including a note: 'NOTE: Template source is from SASHELP.TMPLMST (read-only file). The generated so that running them will write to the current ODS table definition for Directory Member list from both PROC DIR'. It also shows the definition of a table named 'Base.Datasets.Members' with various attributes and a header.

The template can be sent to a separate file by including the file= syntax.

The screenshot shows two windows. The top window is the 'Program Editor - (Untitled)' with the following code:

```
Command ==>
00001 proc template;
00002   path sashelp.tmplmst;
00003   source base.catalog.random /
00004   file = "a:\random_template.txt";
00005 run;
00006
```

The bottom window is a 'random_template - Notepad' showing the template code that was saved to the file:

```
define table Base.Catalog.Random;
notes "To print the engine/host specific contents listing by entry for a random acce";
dynamic obname_width obname_width_max desc_width desc_width_max libname memname;
column num obname type tombstone level crdate moddate desc pagesize blocksizes blockcnt b;
header main;
translate
_val_=-2 into " ";
_val_=-1 into " ";
define main;
text "Contents of Catalog " libname " " memname;
space = 1;
spill_adj;
spill_margin;
end;
define num;
header = " / ";
style = kowheader;
1;
end;
```

Template Stores

Templates are collected in a 'store'. A store denotes the library reference and catalog holding the templates.

Templates can be placed in as many stores as needed.

The following two stores are noteworthy:

- Sashelp.Tmplmst
- Sasuser.Templat

The Sashelp.Tmplmst store holds templates provided by the SAS Institute.

The Sasuser.Templat store holds the programmer-defined templates. When the programmer edits or creates an original template, it can be placed in the Sasuser.Templat store.

A template in Sasuser.Templat can have the same name as a template in Sashelp.Tmplmst.

By default, ODS first searches the Sasuser store, then the Sashelp store. The first template it finds with the correct name is used.

Templates can also be placed in locations other than Sashelp.Tmplmst and Sasuser.Templat.

The search order becomes more complex and the ODS Path statement specifies the search order for the named template.

The screenshot shows the 'Program Editor - (Untitled)' with the following code:

```
Command ==>
00001 ods path work.mystuff (read);
00002 ods path show;
00003
00004 ods path sasuser.templat (update)
00005       sashelp.tmplmst (read);
00006 ods path show;
00007
```

The ODS Path statement has two roles:

- It specifies the locations as well as the search sequence.
- When used in conjunction with 'show', it displays the locations and search sequence in the log window.

Notice that the ODS Path statement contains syntax about access.

Three access modes are recognized:

- Read – (default) read only; no modifications permitted.
- Update – permits adding to existing definition and read privileges.
- Write – permits overwriting existing definition and read privileges.

The screenshot shows the 'Log - (Untitled)' window. It displays the execution of the ODS Path statement, including the 'Current ODS PATH list is:' and the list of paths and their access modes:

```
1. WORK.MYSTUFF(READ)
1434
1435 ods path sasuser.templat (update)
1436       sashelp.tmplmst (read);
1437 ods path show;
Current ODS PATH list is:
1. SASUSER.TEMPLAT(UPDATE)
2. SASHELP.TMPLMST(READ)
```

Table vs. Style Templates

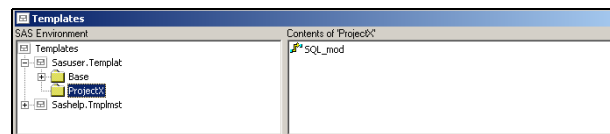
ODS uses a table template to structure the data, and a style template to specify the overall color, font, font size and so forth of the output.

Consider the same table template as using different style templates.

```

Program Editor - (Untitled)
Command ==>
00001 ods select attributes (persist);
00002
00003 ods html file="a:\style_one.html"
00004 style = default;
00005 proc contents data=saved.demog;
00006 run;
00007 ods html close;
00008
00009 ods html file = "a:\style_two.html"
00010 style = minimal;
00011 proc contents data=saved.demog;
00012 run;
00013 ods html close;
00014
00015 ods select all;
00016

```



```

Template Editor
proc template;
link ProjectX.SQL_mod to Base.SQL / notes = "Project X Output Template";
run;
*** END OF TEXT ***

```

Creating a Table Template

Proc Template can be used to edit and create a template.

Although the SAS System comes with numerous formats for ODS output, some projects and business needs require customized output.

The following example shows how to create a new template for specific output created through Data_Null_Step.

The CONTENTS Procedure

Data Set Name:	SAVED.DEMOG	Observations:	50
Member Type:	DATA	Variables:	10
Engine:	V8	Indexes:	0
Created:	8:38 Monday, October 4, 1999	Observation Length:	104
Last Modified:	8:38 Monday, October 4, 1999	Deleted Observations:	0
Protection:		Compressed:	NO
Data Set Type:		Sorted:	NO
Label:			

The CONTENTS Procedure

Data Set Name:	SAVED.DEMOG	Observations:	50
Member Type:	DATA	Variables:	10
Engine:	V8	Indexes:	0
Created:	8:38 Monday, October 4, 1999	Observation Length:	104
Last Modified:	8:38 Monday, October 4, 1999	Deleted Observations:	0
Protection:		Compressed:	NO
Data Set Type:		Sorted:	NO
Label:			

The distinction between template types is important when editing or creating templates.

The programmer must edit or create the one that handles the aspect of the output to be manipulated.

Editing a Table Template

Templates can be edited or created from scratch.

It is easier to edit an existing template.

In either case, templates not provided by SAS are not located in the SasHELP.Tmplmst store.

```

Program Editor - (Untitled)
Command ==>
00001 proc template;
00002 define table ProjectX.SQL_mod;
00003 parent = Base.SQL;
00004 notes "Project X Output Template";
00005 end;
00006 run;
00007

```

```

Log - (Untitled)
1 proc template;
NOTE: Writing HTML Body file: sashtm.htm
2 define table ProjectX.SQL_mod;
3 parent = Base.SQL;
4 notes "Project X Output Template";
5 end;
NOTE: LINK 'ProjectX.SQL_mod' has been saved to: SASUSER.TEMPLAT
6 run;
NOTE: PROCEDURE TEMPLATE used:
real time 0.07 seconds
cpu time 0.06 seconds

```

```

Program Editor - (Untitled)
Command ==>
00001 proc format;
00002 value $sexfmt
00003 "M" = "Male" "F" = "Female";
00004 value $statfmt
00005 "M" = "Married" "P" = "Separated"
00006 "D" = "Divorced" "W" = "Widowed"
00007 "S" = "Single";
00008 run;
00009

```

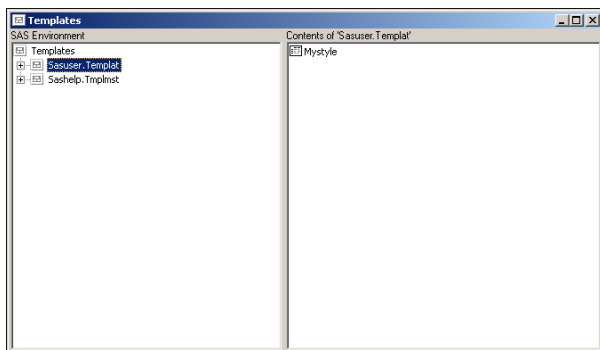
```

Program Editor - (Untitled)
Command ==>
00010 proc template;
00011 define table mystyle;
00012 column name status salary gender;
00013 define name;
00014 width = 20
00015 header = "Employee's Name";
00016 end;
00017 define status;
00018 width = 10
00019 format = $statfmt.
00020 header = "Marital Status";
00021 end;
00022 define salary;
00023 width = 10
00024 format = dollar9.
00025 header = "Annual Earnings";
00026 end;
00027 define gender;
00028 width = 9
00029 format = $sexfmt.
00030 header = "Gender";
00031 end;
00032 end;
00033 run;
00034
00035 data _null_;
00036 set saved.demog (keep=name gender status salary);
00037 file print ods=(template='mystyle');
00038 put _ods_;
00039 run;
00040

```

Output - (Untitled)

Employee's Name	Marital Status	Annual Earnings	Gender
Dave Derry	Single	\$13,592	Male
Julia Pendlebury	Married	\$8,870	Female
Norman Harvey	Divorced	\$12,672	Male
Harold Hicks	Married	\$23,760	Male
Mary Molesworth	Separated	\$10,512	Female
Bob Bobington	Single	\$7,520	Male
Freda Sanford	Married	\$28,512	Female
Julia Kidd	Married	\$14,840	Female
Helen Cinderford,	Married	\$47,520	Female
Mary Chapel	Married	\$23,760	Female
Jennifer Dawson	Married	\$20,592	Female
Julie Jennings	Married	\$13,760	Female
Shirley Walters	Widowed	\$47,520	Female
Diane Duley	Married	\$13,840	Female
Dawn Duvet	Single	\$12,840	Female
Steve Jones		\$11,008	Male
Brian Ellows	Divorced	\$53,070	Male
David Bolling	Married	\$11,360	Male
James Kinderly	Married	\$30,360	Male
Terence Laffer	Single	\$12,355	Male
Frederick Eldridon	Married	\$10,483	Male

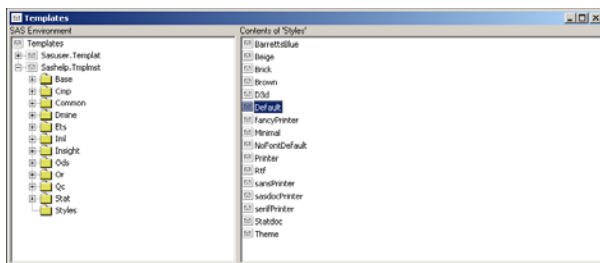


Proc Template gives us the ability to create specific structures.

For additional Proc Template programming syntax, refer to advanced courses on ODS.

ODS Style Templates

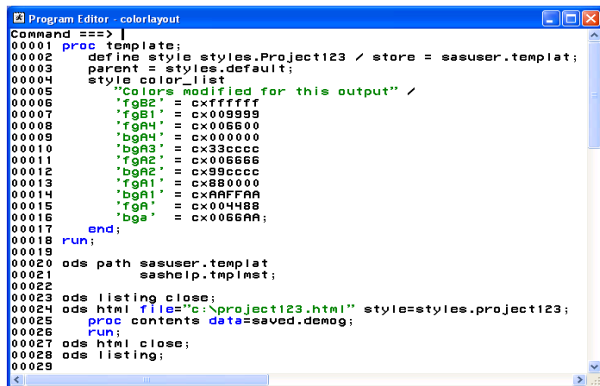
The basis for all ODS styles is the Styles.Default template.



The Style templates can be modified as needed by the user.

A few illustrations are displayed in the following sections.

Let's change the color layout.



The CONTENTS Procedure

Data Set Name	SAVED.DEMOG	Observations	50
Member Type	DATA	Variables	10
Engine	V9	Indexes	0
Created	9:36 Monday, September 23, 2002	Observation Length	104
Last Modified	9:36 Monday, September 23, 2002	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS		
Encoding	Default		

Engine/Host Dependent Information

Change the Proc Template syntax to point to D3D as the parent (by changing line 3 above).

The following outcome is displayed:

The CONTENTS Procedure

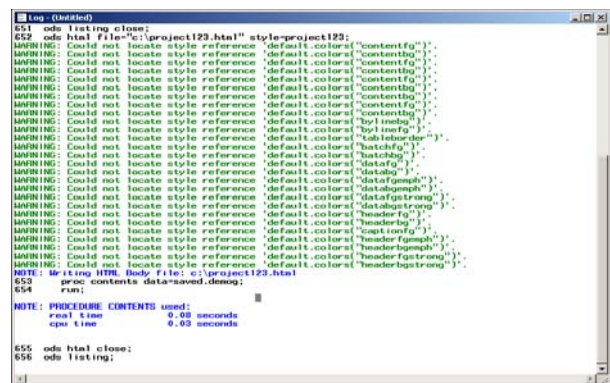
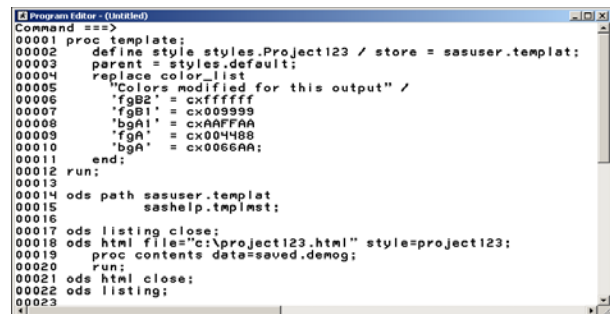
Data Set Name	SAVED.DEMOG	Observations	50
Member Type	DATA	Variables	10
Engine	V9	Indexes	0
Created	9:36 Monday, September 23, 2002	Observation Length	104
Last Modified	9:36 Monday, September 23, 2002	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS		
Encoding	Default		

Engine/Host Dependent Information

We can modify syntax to reference only those colors which need to be changed.

We can use syntax that does not re-reference the colors which stay the same.

However, note the warning messages in the log below.



When using the Replace statement, the entire list must be replaced, even if it means giving the same values as the default.

Simply changing the colors requiring modification does not work. SAS does not have a color reference for any color missing from the Replace statement.

The Style statement allows the programmer to modify only specific features of the template as shown earlier.

Any feature that is not changed in the Style statement is read from the parent template.

```

Program Editor - (Untitled)
Command ==>
00001 proc template;
00002   define style styles.ProjectABC / store = sasuser.templat;
00003   parent = styles.default;
00004   style color_list
00005     "Colors modified for this output" /
00006     'fgB2' = cxfffff;
00007     'fgB1' = cx009999;
00008     'bgA1' = cxAAFFAA;
00009     'fgA'  = cx004488;
00010     'bgA'  = cx0066AA;
00011   end;
00012 run;
00013
00014 ods path sasuser.templat
00015       sashelp.templst;
00016
00017 ods listing close;
00018 ods html file="c:\project123.html" style=projectABC;
00019   proc contents data=saved.demog;
00020   run;
00021 ods html close;
00022 ods listing;
00023

```

```

Log - (Untitled)
713
714   ods path sasuser.templat
715       sashelp.templst;
716
717   ods listing close;
718   ods html file="c:\project123.html" style=projectABC;
NOTE: Writing HTML Body file: c:\project123.html
719   proc contents data=saved.demog;
720   run;

NOTE: PROCEDURE CONTENTS used:
      real time    0.12 seconds
      cpu time     0.03 seconds

721   ods html close;
722   ods listing;

```

For those colors not specified in the Style statement, the template inherits the information from the parent template (in this case, default).

ODS VERIFY

The ODS VERIFY statement is used to activate and deactivate a warning in the Log window that the template used is not supplied by SAS Institute.

The default is 'off'.

```

Program Editor - (Untitled)
Command ==>
00001 ods verify on;
00002
00003 ods html file="beige.html" style=beige;
00004   proc contents data=saved.demog;
00005   run;
00006 ods html close;
00007
00008 ods html file="project123.html" style=project123;
00009   proc contents data=saved.demog;
00010   run;
00011 ods html close;
00012
00013 ods verify off;
00014

```

```

Log - (Untitled)
622   ods verify on;
623
624   ods html file="beige.html" style=beige;
NOTE: Writing HTML Body file: beige.html
625   proc contents data=saved.demog;
626   run;

NOTE: PROCEDURE CONTENTS used:
      real time    0.31 seconds
      cpu time     0.05 seconds

627   ods html close;
628
629   ods html file="project123.html" style=project123;
WARNING: Template STYLES.PROJECT123 was not supplied by SAS Institute!
NOTE: Writing HTML Body file: project123.html
630   proc contents data=saved.demog;
631   run;

NOTE: PROCEDURE CONTENTS used:
      real time    0.15 seconds
      cpu time     0.05 seconds

632   ods html close;
633
634   ods verify off;

```

Creating a SAS Data Set

We can use ODS to create a data set from an output object.

This means that the statistics are created using Proc Tabulate or Proc Means. Output is directed into a SAS data set as variables and observations.

To save output as a SAS data set, first use ODS trace to identify the path of the output.

```

Program Editor - tabulate
Command ==>
00001 /* determine name of table */
00002
00003 ods trace on;
00004
00005   proc tabulate data = saved.demog;
00006     class gender;
00007     table gender;
00008   run;
00009
00010   ods trace off;
00011
00012 /* direct table data to a dataset */
00013
00014   ods listing close;
00015
00016   ods output tabulate.report.table = work.genderstats;
00017   proc tabulate data = saved.demog;
00018     class gender;
00019     table gender;
00020   run;
00021   ods output close;
00022   ods listing;
00023
00024   proc print data = work.genderstats;
00025   run;
00026

```

For example, the path for Proc Tabulate is Tabulate.Report.Table.

```

Log - (Untitled)
Command ==>
301 /* determine name of table */
302
303 ods trace on;
304
305   proc tabulate data = saved.demog;
306     class gender;
307     table gender;
308   run;

Output Added:
-----
Name:      Table
Label:     Table 1
Data Name: Report
Path:     Tabulate.Report.Table

NOTE: There were 50 observations read from the data set SAVED.DEMOG.
NOTE: PROCEDURE TABULATE used (Total process time):
      real time    0.01 seconds
      cpu time     0.02 seconds

309
310   ods trace off;

```

The Listing window shows the results of the Proc print.

Obs	GENDER	_TYPE_	_PAGE_	_TABLE_	N
1	F	1	1	1	22
2	M	1	1	1	28

The syntax may be modified, given the parameters and the potential of the Procedure.

```

Program Editor - tabulate
Command ==>
00001 ods output tabulate.report.table = work.DemogStats;
00002   proc tabulate data = saved.demog;
00003     class gender status;
00004     table gender;
00005     table status;
00006   run;
00007   ods output close;
00008
00009   ods listing;
00010
00011   proc print data = work.DemogStats;
00012   run;
00013

```

Obs	GENDER	STATUS	_TYPE_	_PAGE_	_TABLE_	N
1			10	1	1	22
2	F		10	1	1	27
3		D	01	1	2	6
4		M	01	1	2	27
5		P	01	1	2	4
6		S	01	1	2	10
7		W	01	1	2	2

MATCH_ALL Option with Data Set Output

The ODS Output syntax can be taken further to achieve some useful results. It may be helpful to have all procedural output flow into the same SAS data set.

Consider the following syntax and output.

```

Program Editor - (Untitled)
Command ==>
00001 ods listing close;
00002
00003 ods output moments=work.moments;
00004 proc univariate data=saved.demog;
00005   var salary age children;
00006 run;
00007 ods output close;
00008
00009 ods listing;
00010
00011 proc print data=work.moments;
00012   var varname label1 cvalue1;
00013 run;
00014

```

Obs	VarName	Label1	cValue1
1	SALARY	N	49
2	SALARY	Mean	19982.3061
3	SALARY	Std Deviation	11386.7755
4	SALARY	Skewness	1.48758687
5	SALARY	Uncorrected SS	2.5789E10
6	SALARY	Coeff Variation	56.9842913
7	AGE	N	50
8	AGE	Mean	38.1
9	AGE	Std Deviation	12.5490873
10	AGE	Skewness	0.35801868
11	AGE	Uncorrected SS	80297
12	AGE	Coeff Variation	32.937237
13	CHILDREN	N	50
14	CHILDREN	Mean	1.82
15	CHILDREN	Std Deviation	1.46649659
16	CHILDREN	Skewness	0.48663962
17	CHILDREN	Uncorrected SS	271
18	CHILDREN	Coeff Variation	80.5767356

The Log window shows the results of the syntax.

```

Log - (Untitled)
Command ==>
175 ods listing close;
176
177 ods output moments=work.moments;
178 proc univariate data=saved.demog;
179   var salary age children;
180 run;

NOTE: The data set WORK.MOMENTS has 18 observations and 7 variables.
NOTE: There were 50 observations read from the data set SAVED.DEMOG.
NOTE: PROCEDURE UNIVARIATE used:
      real time    0.03 seconds
      cpu time     0.03 seconds

181 ods output close;
182
183 ods listing;
184
185 proc print data=work.moments;
186   var varname label1 cvalue1;
187 run;

NOTE: There were 18 observations read from the data set WORK.MOMENTS.
NOTE: PROCEDURE PRINT used:
      real time    0.01 seconds
      cpu time     0.01 seconds

```

On the other hand, the MATCH_ALL option will separate each variable's data to flow into a different SAS dataset.

```

Program Editor - (Untitled)
Command ==>
00001 ods listing close;
00002
00003 ods output moments(match_all)=work.moments;
00004 proc univariate data=saved.demog;
00005   var salary age children;
00006 run;
00007 ods output close;
00008
00009 ods listing;
00010
00011 proc print data=work.moments;
00012   var varname label1 cvalue1;
00013 run;
00014

```

Obs	Var	Label1	cValue1
1	SALARY	N	49
2	SALARY	Mean	19982.3061
3	SALARY	Std Deviation	11386.7755
4	SALARY	Skewness	1.48758687
5	SALARY	Uncorrected SS	2.5789E10
6	SALARY	Coeff Variation	56.9842913

```

Log - (Untitled)
Command ==>
188 ods listing close;
189
190 ods output moments(match_all)=work.moments;
191 proc univariate data=saved.demog;
192   var salary age children;
193 run;

NOTE: The data set WORK.MOMENTS has 6 observations and 7 variables.
NOTE: The data set WORK.MOMENTS1 has 6 observations and 7 variables.
NOTE: The data set WORK.MOMENTS2 has 6 observations and 7 variables.
NOTE: There were 50 observations read from the data set SAVED.DEMOG.
NOTE: PROCEDURE UNIVARIATE used:
      real time    0.05 seconds
      cpu time     0.05 seconds

194 ods output close;
195
196 ods listing;
197
198 proc print data=work.moments;
199   var varname label1 cvalue1;
200 run;

NOTE: There were 6 observations read from the data set WORK.MOMENTS.
NOTE: PROCEDURE PRINT used:
      real time    0.25 seconds
      cpu time     0.00 seconds

```

MATCH_ALL option with Macro Variable Output

The MATCH_ALL option can also be used to create a macro variable value located in the Global Symbol Table.

The macro variable holds the value of the SAS data sets created from the MATCH_ALL option.

```

Program Editor - (Untitled)
Command ==>
00001 ods listing close;
00002
00003 ods output moments(match_all=list)=work.moments7;
00004 proc univariate data=saved.demog;
00005   var salary age children;
00006 run;
00007 ods output close;
00008
00009 ods listing;
00010
00011 %put &list;
00012

```

```

Log - (Untitled)
Command ==>
247 ods listing close;
248
249 ods output moments(match_all=list)=work.moments7;
250 proc univariate data=saved.demog;
251   var salary age children;
252 run;

NOTE: The data set WORK.MOMENTS7 has 6 observations and 7 variables.
NOTE: The data set WORK.MOMENTS8 has 6 observations and 7 variables.
NOTE: The data set WORK.MOMENTS9 has 6 observations and 7 variables.
NOTE: There were 50 observations read from the data set SAVED.DEMOG.
NOTE: PROCEDURE UNIVARIATE used:
      real time    0.05 seconds
      cpu time     0.05 seconds

253 ods output close;
254
255 ods listing;
256
257 %put &list;
WORK.MOMENTS7 WORK.MOMENTS8 WORK.MOMENTS9
258

```

MATCH_ALL option with By-Group Processing

The MATCH_ALL option can be used to create interesting results with By-Group processing.

```

Program Editor - (Untitled)
Command ==>
00001 ods listing close;
00002
00003 proc sort data=saved.demog
00004   out=work.demog;
00005   by status;
00006 run;
00007
00008 ods output summary(match_all)=work.moments;
00009 proc means data=work.demog;
00010   by status;
00011 run;
00012 ods output close;
00013
00014 ods listing;
00015
00016 proc print data=work.moments4;
00017 run;
00018

```

The distinct values of Status are ' , D, M, P, S, W.

By printing work.moment4 (below), the S value is shown.

(Why is the S value given the suffix 4?)

Output: (Unlabeled)						
Obs	STATUS	AGE	AGE_N	AGE_Mean	AGE_StdDev	AGE_Min
1	S	AGE	10	30.4	11.992590305	18
Obs	AGE_Max	HEIGHT	HEIGHT_N	HEIGHT_Mean	HEIGHT_StdDev	HEIGHT_Min
1	54	HEIGHT	10	69.5	12.360330812	
Obs	HEIGHT_Max	WEIGHT	WEIGHT_N	WEIGHT_Mean	WEIGHT_StdDev	WEIGHT_Min
1	52	87	WEIGHT	10	72	
Obs	WEIGHT_Max	WEIGHT_Min	WEIGHT_Max	WEIGHT_Min	CHILDREN_N	CHILDREN_Mean
1	8.2865352631	60	82	CHILDREN	10	
Obs	CHILDREN_Mean	CHILDREN_StdDev	CHILDREN_Min	CHILDREN_Max		
1	0	0	0	0		
Obs	CARS	CARS_N	CARS_Mean	CARS_StdDev	CARS_Min	CARS_Max
1	CARS	10	0.6	0.5163977795	0	
Obs	CARS_Max	SALARY	SALARY_N	SALARY_Mean	SALARY_StdDev	SALARY_Min
1	1	SALARY	10	18658.5	13744.711017	
Obs	SALARY_Max	SALARY_Min	SALARY_Max	SALARY_Min		
1	7520	47520				

Results
Means
STATUS="S"
STATUS="D"
STATUS="M"
STATUS="P"
STATUS="S"
STATUS="W"
Print:

MATCH_ALL option with Persist=Proc

Consider the programming example below.

Two separate data sets are used in Proc univariate.

Three variables are referenced.

The resulting SAS data sets are work.quant, work.quant1, and work.quant2.

Program Editor - (Untitled)
Command ==>
00001 ods listing close;
00002
00003 ods output quantiles(match_all)=work.quant;
00004 proc univariate data=saved.demog;
00005 var age children;
00006 run;
00007 ods output close;
00008
00009 ods output quantiles=work.quant2;
00010 proc univariate data=saved.demogius;
00011 var salary;
00012 run;
00013 ods output close;
00014

The Log window shows the output created.

Log - (Untitled)
Command ==>
426 ods listing close;
427
428 ods output quantiles(match_all)=work.quant;
429 proc univariate data=saved.demog;
430 var age children;
431 run;
NOTE: The data set WORK.QUANTS has 11 observations and 3 variables.
NOTE: The data set WORK.QUANTS1 has 11 observations and 3 variables.
NOTE: There were 50 observations read from the data set SAVED.DEMOG.
NOTE: PROCEDURE UNIVARIATE used:
real time 0.36 seconds
cpu time 0.05 seconds
432 ods output close;
433
434 ods output quantiles=work.quant2;
435 proc univariate data=saved.demogius;
436 var salary;
437 run;
NOTE: The data set WORK.QUANTS2 has 11 observations and 3 variables.
NOTE: There were 104 observations read from the data set SAVED.DEMOGIUS.
NOTE: PROCEDURE UNIVARIATE used:
real time 0.03 seconds
cpu time 0.03 seconds
438 ods output close;

This syntax can be simplified by using the PERSIST=PROC syntax.

Program Editor - (Untitled)
Command ==>
00001 ods listing close;
00002
00003 ods output quantiles(match_all
00004 persist=proc)=work.quant;
00005 proc univariate data=saved.demog;
00006 var age children;
00007 run;
00008
00009 proc univariate data=saved.demogius;
00010 var salary;
00011 run;
00012 ods output close;
00013

The Log window shows the creation of the same three SAS data sets.

Log - (Untitled)
Command ==>
439 ods listing close;
440
441 ods output quantiles(match_all
442 persist=proc)=work.quant;
443 proc univariate data=saved.demog;
444 var age children;
445 run;
NOTE: The data set WORK.QUANTS has 11 observations and 5 variables.
NOTE: The data set WORK.QUANTS1 has 11 observations and 5 variables.
NOTE: There were 50 observations read from the data set SAVED.DEMOG.
NOTE: PROCEDURE UNIVARIATE used:
real time 0.36 seconds
cpu time 0.05 seconds
446
447 proc univariate data=saved.demogius;
448 var salary;
449 run;
NOTE: The data set WORK.QUANTS2 has 11 observations and 5 variables.
NOTE: There were 104 observations read from the data set SAVED.DEMOGIUS.
NOTE: PROCEDURE UNIVARIATE used:
real time 0.04 seconds
cpu time 0.04 seconds
450 ods output close;

MATCH_ALL Option with Datasetlist

The list of data sets can also be captured in a macro variable.

Program Editor - (Untitled)
Command ==>
00001 ods listing close;
00002
00003 ods output quantiles(match_all=datasetlist
00004 persist=proc)=work.quant;
00005 proc univariate data=saved.demog;
00006 var age children;
00007 run;
00008
00009 proc univariate data=saved.demogius;
00010 var salary;
00011 run;
00012 ods output close;
00013

Log - (Untitled)
401 ods listing close;
402
403 ods output quantiles(match_all=datasetlist
404 persist=proc)=work.quant;
405 proc univariate data=saved.demog;
406 var age children;
407 run;
NOTE: The data set WORK.QUANTS has 11 observations and 5 variables.
NOTE: The data set WORK.QUANTS1 has 11 observations and 5 variables.
NOTE: There were 50 observations read from the data set SAVED.DEMOG.
NOTE: PROCEDURE UNIVARIATE used:
real time 0.25 seconds
cpu time 0.03 seconds
408
409 proc univariate data=saved.demogius;
410 var salary;
411 run;
NOTE: The data set WORK.QUANTS2 has 11 observations and 5 variables.
NOTE: There were 104 observations read from the data set SAVED.DEMOGIUS.
NOTE: PROCEDURE UNIVARIATE used:
real time 0.10 seconds
cpu time 0.04 seconds
412 ods output close;

MATCH_ALL Option with Persist=Run

The PERSIST=RUN option acts on Procedures where the step boundary is the Quit statement.

```

Program Editor - (Untitled)
Command ==>
00001 ods listing close;
00002
00003 ods output variablesalpha(match_all
00004                                persist=run)=work.varlist;
00005
00006 proc datasets library=saved;
00007     contents data=demog;
00008 run;
00009     contents data=demogius;
00010 run;
00011 quit;
00012 ods output close;

```

```

Log - (Untitled)
Command ==>
476 ods listing close;
477
478 ods output variablesalpha(match_all
479                                persist=run)=work.varlist;
480
481 proc datasets library=saved;
482     contents data=demog;
483 run;
484     contents data=demogius;
485 run;
486 quit;
NOTE: The data set WORK.VARLIST has 10 observations and 7
variables.
NOTE: The data set WORK.VARLIST1 has 16 observations and 7
variables.
NOTE: PROCEDURE DATASETS used:
      real time    0.02 seconds
      cpu time     0.13 seconds
486 ods output close;

```

Introduction to ODS Printer

ODS Printer syntax allows output to be sent to a high-resolution printer.

As illustrated earlier, the ODS statements used in routing output to a printer destination both open and close the request.

This module focuses on creating output and routing it directly to a printer at the time of program submission.

Also, ODS Printer syntax can create a file (e.g., *.ps, *.rtf), which can be used for printing from disk at a later point in time.

Direct Printing

By default, the ODS Printer syntax sends output to a printer device for presentation on paper.

```

Program Editor - (Untitled)
Command ==>
00001 ods listing close;
00002
00003 ods printer;
00004 proc sql;
00005     select *
00006     from saved.demog;
00007 quit;
00008 ods printer close;
00009
00010 ods listing;
00011

```

The Log window confirms the syntax action.

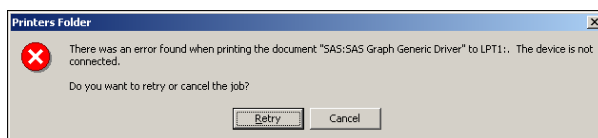
```

Log - (Untitled)
Command ==>
545 ods listing close;
546
547 ods printer;
NOTE: Sending ODS PRINTER output to printer "Apple LaserWriter II NT v47.0".
548 proc sql;
549     select *
550     from saved.demog;
551 quit;
NOTE: PROCEDURE SQL used:
      real time    0.15 seconds
      cpu time     0.15 seconds
552 ods printer close;
NOTE: ODS PRINTER printed 2 pages.
553
554 ods listing;

```

The output is sent to the default printer, specified in the Note message in the log.

Should this device be off line, an error appears on the screen.



Additional programming can enhance results with ODS Printer output.

The following list of options are included in the opening ODS Printer statement.

Fontscale =	Specify a percent value. Default is 100. Font size can be adjusted as needed.
Nocolor Color	Use information supplied in the style. Default is Nocolor.
SAS	Use the printer drivers supplied by SAS. Used only for Windows OS.
Uniform	Print output uniformly across pages.
Printer =	Names a printer device.
Style =	Select a style by name (e.g., D3D, serifPrinter)

Example syntax is displayed below.

```

Program Editor - (Untitled)
Command ==>
00001 ods listing close;
00002
00003 ods printer uniform color style=beige
00004                                printer = 'HP DeskJet 850 Series';
00005 proc sql;
00006     select *
00007     from saved.demog;
00008 quit;
00009 ods printer close;
00010
00011 ods listing;
00012

```

Printing to Disk - PostScript

Rather than send ODS Printer output to a print device, the output can be stored to disk in a variety of document formats, including the following.

- PostScript – PS
- Printer Control Language – PCL
- Portable Document Files – PDF
- Rich Text Format – RTF

First, we will store the output as a Postscript file.

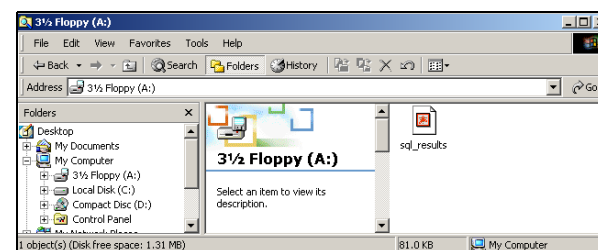
```

Program Editor - (Untitled)
Command ==>
00001 ods listing close;
00002
00003 ods printer file="a:\sql_results.ps";
00004 proc sql;
00005     select *
00006     from saved.demog;
00007 quit;
00008 ods printer close;
00009
00010 ods listing;
00011
00012

```

To direct SAS to use the generic postscript printer provided by SAS we need to modify line 3 in the code displayed above, as follows:

```
ods printer file==a:\sql_results.ps= ps;
```



```

Log - (Untitled)
Command ==>
586 ods listing close;
587
588 ods printer file="a:\sql_results.ps";
NOTE: Writing ODS PRINTER output to file "a:\sql_results.ps" (DISK),
printer "Apple LaserWriter II NT VMT.0".
589 proc sql;
590 select *
591 from saved.demog;
592 quit;
NOTE: PROCEDURE SQL used:
real time 9.31 seconds
cpu time 0.17 seconds
593 ods printer close;
NOTE: ODS PRINTER printed 2 pages to a:\sql_results.ps.
594
595 ods listing;

```

Printing to Disk - Portable Document Files

ODS Printer can create PDF output. The process depends on the version of SAS in use, as well as converting RTF to PDF.

In SAS Version 8, generating a Postscript document using the syntax shown previously creates PDF output.

This PS document is then distilled using the fill version of Adobe Acrobat (not Adobe Reader).

In SAS Version 8e (8.1), the process to create PDF is more direct.

```

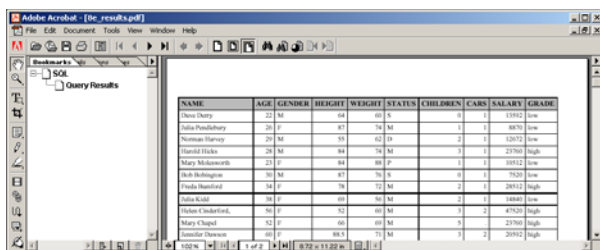
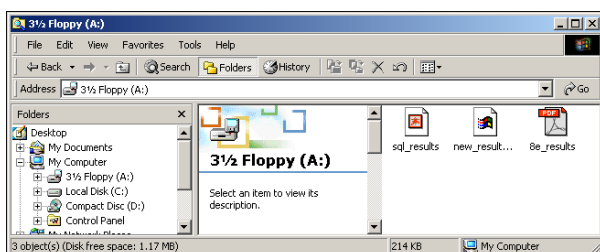
Program Editor - (Untitled)
Command ==>
00001 ods listing close;
00002
00003 ods printer file="a:\8e_results.pdf" pdf;
00004 proc sql;
00005 select *
00006 from saved.demog;
00007 quit;
00008 ods printer close;
00009
00010 ods listing;
00011

```

```

Log - (Untitled)
Command ==>
609 ods listing close;
610
611 ods printer file="a:\8e_results.pdf" pdf;
WARNING: The PDF option is experimental in this release.
NOTE: Writing ODS PRINTER output to file "a:\8e_results.pdf" (DISK), printer
612 proc sql;
613 select *
614 from saved.demog;
615 quit;
NOTE: PROCEDURE SQL used:
real time 0.41 seconds
cpu time 0.11 seconds
616 ods printer close;
NOTE: ODS PRINTER printed 2 pages to a:\8e_results.pdf.
617
618 ods listing;

```



Printing to Disk - Rich Text Format

ODS can create Rich Text Format (*.rtf) for use in a word-processing application.

The ODS RTF output was 'experimental' in earlier versions; Version 8e has dropped the warning.

The results are satisfactory.

```

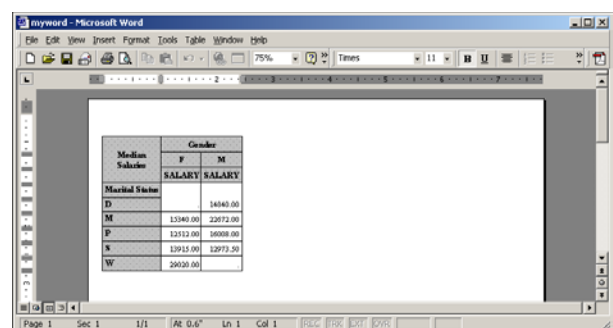
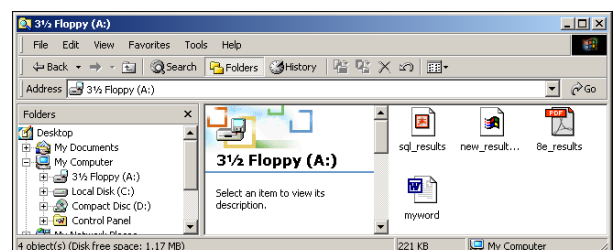
Program Editor - (Untitled)
Command ==>
00001 ods rtf file="a:\myword.rtf";
00002
00003 proc tabulate data=saved.demog;
00004 class status gender;
00005 var salary;
00006 table status,gender*salary*median /
00007 rts=17 box="Median Salaries";
00008 label status = "Marital Status"
00009 gender = "Gender"
00010 salary = " ";
00011 keylabel median = " ";
00012 run;
00013 ods rtf close;
00014

```

```

Log - (Untitled)
632 ods rtf file="a:\myword.rtf";
NOTE: Writing RTF Body file: a:\myword.rtf
633
634 proc tabulate data=saved.demog;
635 class status gender;
636 var salary;
637 table status,gender*salary*median /
638 rts=17 box="Median Salaries";
639 label status = "Marital Status"
640 gender = "Gender"
641 salary = " ";
642 keylabel median = " ";
643 run;
NOTE: There were 50 observations read from the data set SAVED.DEMOG.
NOTE: PROCEDURE TABULATE used:
real time 0.23 seconds
cpu time 0.09 seconds
644 ods rtf close;

```



Using Styles to Control Colors and Fonts

Controlling colors and fonts can enhance the appearance of Proc Tabulate output to an HTML file.

Colors can be either foreground or background. Fonts include face, size, style, weight, and width.

```

Command ==>
00001 ods html file = "c:\mystyle.html"
00002 style = default;
00003
00004 proc tabulate data=saved.demog f=dollar11.2;
00005 class status gender;
00006 var salary;
00007 table status,gender*salary=" *median /
00008 misstext = "N/A"
00009 box = "Median Salaries";
00010 keylabel median=" ";
00011 run;
00012 ods html close;
00013

```

Median Salaries	GENDER	
	F	M
STATUS		
D	N/A	\$14,840.00
M	\$15,340.00	\$22,672.00
P	\$12,512.00	\$16,008.00
S	\$13,915.00	\$12,973.50
W	\$29,020.00	N/A

Median Salaries	GENDER	
	F	M
STATUS		
D	N/A	\$14,840.00
M	\$15,340.00	\$22,672.00
P	\$12,512.00	\$16,008.00
S	\$13,915.00	\$12,973.50
W	\$29,020.00	N/A

Additional font modifications can include the following:

- Font_face Times, Courier, Helvetica
- Font_size 12pt, 8pt, 10pt
- Font_style Roman, Italic
- Font_weight Medium, Bold
- Font_width Wide, Narrow

Using Style = { }

```

Command ==>
00001 ods html file = "c:\mystyle.html"
00002 style = default;
00003
00004 proc tabulate data=saved.demog f=dollar11.2;
00005 class status gender / style={background=red};
00006 var salary;
00007 table status,gender*salary=" *median /
00008 misstext = "N/A"
00009 box = "Median Salaries";
00010 keylabel median=" ";
00011 run;
00012 ods html close;
00013

```

Median Salaries	GENDER	
	F	M
STATUS		
D	N/A	\$14,840.00
M	\$15,340.00	\$22,672.00
P	\$12,512.00	\$16,008.00
S	\$13,915.00	\$12,973.50
W	\$29,020.00	N/A

Using Proc Tabulate to Modify the Table of Contents

ODS output can be enhanced by using HTML frames to hold Proc Tabulate output.

```

Command ==>
00001 proc sort data=saved.demog out=work.demog;
00002 by status;
00003 run;
00004 ods listing close;
00005 ods html body = "c:\proctab.html"
00006 contents = "c:\toc.html"
00007 frame = "c:\findings.html"
00008 style = statdoc;
00009 proc tabulate data=work.demog;
00010 by status;
00011 var age children salary;
00012 class grade gender;
00013 table grade,gender*(age children salary)*mean;
00014 run;
00015 ods html close;
00016 ods listing;
00017

```

Notice the Table of Contents headings for each value of Status: "Cross-tabular summary report" and "Table 1".

The string "Table of Contents" also appears at the top of the Contents file.

Additional colors for use in the foreground and/or background include: red, pink, orange, yellow, yellow-green, green, blue, purple, black, white and cyan.

```

Command ==>
00001 ods html file = "c:\mystyle.html"
00002 style = default;
00003
00004 proc tabulate data=saved.demog f=dollar11.2;
00005 class status gender / style={font_style=italic};
00006 var salary;
00007 table status,gender*salary=" *median /
00008 misstext = "N/A"
00009 box = "Median Salaries";
00010 keylabel median=" ";
00011 run;
00012 ods html close;
00013

```

Table of Contents						
1. The Tabulate Procedure						
-STATUS=*						
-Cross tabular summary report						
-Table 1						
-STATUS=D						
-Cross tabular summary report						
-Table 1						
-STATUS=M						
-Cross tabular summary report						
-Table 1						
-STATUS=P						

To remove the string "Table of Contents", submit syntax using the Replace statement.

```

Command ==>
00001 proc template;
00002   define style styles.mylab;
00003     parent = styles.statdoc;
00004     replace ContentTitle from Index;
00005   end;
00006 run;
00007
00008 ods listing close;
00009 ods html body = "c:\proctab.html"
00010   contents = "c:\toc.html"
00011   frame = "c:\findings.html"
00012   style = mylab;
00013 proc tabulate data=work.demog;
00014   by status;
00015   var age children salary;
00016   class grade gender;
00017   table grade,gender*(age children salary)*mean;
00018 run;
00019 ods html close;
00020 ods listing;
00021

```

SAS Output Frame - Microsoft Internet Explorer

Address: C:\findings.html

1. The Tabulate
Procedure
-STATUS=

STATUS=M						
GRADE	GENDER					
	F			M		
	AGE	CHILDREN	SALARY	AGE	CHILDREN	SALARY
Mean	Mean	Mean	Mean	Mean	Mean	Mean
high	52.38	3.13	26887.00	38.00	2.75	25073.75
low	42.88	1.25	12796.50	35.00	3.00	12840.00

Alternative syntax for changing the procedure name string uses macro variables.

In this syntax, understand that HTML indicates a comment by starting the string with '`<!--`' and ending it with '`-->`'.

SAS Output Frame - Microsoft Internet Explorer

Address: C:\findings.html

1. The Tabulate
Procedure
-STATUS=

STATUS=M						
GRADE	GENDER					
	F			M		
	AGE	CHILDREN	SALARY	AGE	CHILDREN	SALARY
Mean	Mean	Mean	Mean	Mean	Mean	Mean
high	52.38	3.13	26887.00	38.00	2.75	25073.75
low	42.88	1.25	12796.50	35.00	3.00	12840.00

To remove bullets (i.e. points in front of the hyperlinks) modify the template as follows:

```

Command ==>
00001 proc template;
00002   define style styles.mylab;
00003     parent = styles.statdoc;
00004     replace ContentTitle from Index;
00005     replace Indexitem from container /
00006     listentryanchor = on
00007     bullet = none;
00008   end;
00009 run;
00010

```

```

Command ==>
00001 proc template;
00002   define style styles.mylab;
00003     parent = styles.statdoc;
00004     replace ContentTitle from Index;
00005     replace Indexitem from container /
00006     listentryanchor = on
00007     bullet = none;
00008   end;
00009 replace ContentProcname from Index /
00010   bullet = none
00011   pretext = symget ("start")
00012   posttext = symget ("end");
00013 run;
00014
00015 %let start = <!--;
00016 %let end = -->;
00017
00018 ods listing close;
00019 ods html body = "c:\proctab.html"
00020   contents = "c:\toc.html"
00021   frame = "c:\findings.html"
00022   style = mylab;
00023 proc tabulate data=work.demog;
00024   by status;
00025   var age children salary;
00026   class grade gender;
00027   table grade,gender*(age children salary)*mean;
00028 run;
00029 ods html close;
00030 ods listing;

```

SAS Output Frame - Microsoft Internet Explorer

Address: C:\findings.html

1. The Tabulate
Procedure
-STATUS=

STATUS=M						
GRADE	GENDER					
	F			M		
	AGE	CHILDREN	SALARY	AGE	CHILDREN	SALARY
Mean	Mean	Mean	Mean	Mean	Mean	Mean
high	52.38	3.13	26887.00	38.00	2.75	25073.75
low	42.88	1.25	12796.50	35.00	3.00	12840.00

SAS Output Frame - Microsoft Internet Explorer

Address: C:\findings.html

1. The Tabulate
Procedure
-STATUS=

STATUS=D			
GRADE	GENDER		
	M		
	AGE	CHILDREN	SALARY
Mean	Mean	Mean	Mean
high	34.50	2.50	34905.00
low	30.75	2.00	13357.33

To remove or modify the title of the procedure, use the ODS Proclabel statement.

```

Command ==>
00011 ods proclabel "Current Tabular Reports";
00012
00013 ods listing close;
00014 ods html body = "c:\proctab.html"
00015   contents = "c:\toc.html"
00016   frame = "c:\findings.html"
00017   style = mylab;
00018 proc tabulate data=work.demog;
00019   by status;
00020   var age children salary;
00021   class grade gender;
00022   table grade,gender*(age children salary)*mean;
00023 run;
00024 ods html close;
00025 ods listing;

```

Traffic Lighting in Proc Tabulate

Traffic lighting is a method of presenting data. It uses colors to show relationships.

Instead of assigning a label to a group, the output displays a group with a specified color. Below all average salaries that are 15000 or lower are displayed in green, the others are presented in red.

```

Program Editor - tabtraffic
Command ==>
00001 proc format;
00002   value $statfmt
00003     'D' = 'Divorced'
00004     'M' = 'Married'
00005     'W' = 'Widowed'
00006     'S' = 'Single'
00007     'SEP' = 'Separated';
00008
00009   value $genfmt
00010     'F' = 'Female'
00011     'M' = 'Male';
00012
00013   value salfmt
00014     low-15000 = 'cx006600'
00015     other = 'red';
00016 run;
00017
00018 title "Traffic Lighting in Proc Tabulate ";
00019 ods html file = "c:\temp\tab.html";
00020 proc tabulate data = saved.demograp
00021   style = {foreground=salfmt.};
00022   class status gender;
00023   var salary;
00024   table status, gender*salary*mean="*f=dollar10.2
00025     / box = "Mean Salary";
00026   format status $statfmt. gender $genfmt.;
00027 run;
00028 ods html close;

```

Traffic Lighting in Proc Tabulate

Mean Salary	gender	
	Female	Male
status	salary	salary
Divorced	\$8,000.00	\$10,000.00
Married	\$10,123.08	\$19,716.67
Single	\$8,500.00	\$3,560.00
Separated	\$18,000.00	\$12,000.00
Widowed	\$30,000.00	

The values of 19716.67, 18000, and 30000 are in red, the rest are in green.

Creating an HTML Output with ODS

ODS offers interesting possibilities for output created with Proc Report, including HTML output.

```

Program Editor - (Untitled)
Command ==>
00001 filename myhtml "c:\ods output\";
00002
00003 ods html file="demog_report.html"
00004   path=myhtml;
00005
00006 proc report data=saved.demog
00007   nowindows colwidth=10;
00008   column gender status salary children cars;
00009   define gender / group
00010     format=$genfmt.
00011     "Gender" " ";
00012   define status / group
00013     format=$statfmt.
00014     "Status" " ";
00015   define salary / analysis mean
00016     format=dollar10.2
00017     "Annual" "Salary";
00018   define children / analysis mean
00019     format=5.2
00020     width=10
00021     "Mean" "Children";
00022   define cars / analysis mean
00023     format=5.2
00024     "Mean" "Cars";
00025 run;
00026
00027 ods html close;

```

Results Viewer - SAS Output

Gender	Status	Annual Salary	Mean Children	Mean Cars
Female	Married	\$19,841.75	2.19	1.07
	Separated	\$12,512.00	1.50	1.00
	Single	\$13,915.00	0.00	1.00
	Widowed	\$29,020.00	2.00	1.00
Male	Divorced	\$21,976.40	2.17	1.00
	Married	\$21,737.27	2.82	1.55
	Separated	\$16,008.00	1.50	0.50
	Single	\$19,844.38	0.00	0.50

Creating Click-Through Web Pages

By adjusting the formatted values of the grouping variables, Proc Report web output can contain click-through features linking them to other HTML pages.

```

Program Editor - (Untitled)
Command ==>
00001 proc format;
00002   value $genfmt
00003     'F' = '<a href="female.html">Female</a>'
00004     'M' = '<a href="male.html">Male</a>';
00005   value $statfmt
00006     'M' = '<a href="married.html">Married</a>'
00007     'W' = '<a href="widowed.html">Widowed</a>'
00008     'D' = '<a href="divorced.html">Divorced</a>'
00009     'S' = '<a href="single.html">Single</a>'
00010     'P' = '<a href="separated.html">Separated</a>';
00011 run;
00012
00013 ods html file="c:\demog_report.html";
00014
00015 proc report data=saved.demog
00016   nowindows colwidth=10;
00017   column gender status salary children cars;
00018   define gender / group format=$genfmt. "Gender" " ";
00019   define status / group format=$statfmt. "Status" " ";
00020   define salary / analysis mean format=dollar10.2
00021     "Annual" "Salary";
00022   define children / analysis mean format=5.2
00023     width=10 "Mean" "Children";
00024   define cars / analysis mean format=5.2
00025     "Mean" "Cars";
00026 run;
00027
00028 ods html close;
00029

```

Results Viewer - SAS Output

Gender	Status	Annual Salary	Mean Children	Mean Cars
Female	Married	\$19,841.75	2.19	1.07
	Separated	\$12,512.00	1.50	1.00
	Single	\$13,915.00	0.00	1.00
	Widowed	\$29,020.00	2.00	1.00
Male	Divorced	\$21,976.40	2.17	1.00
	Married	\$21,737.27	2.82	1.55
	Separated	\$16,008.00	1.50	0.50
	Single	\$19,844.38	0.00	0.50

Traffic Lighting in Proc Report

Traffic lighting is a method of presenting data. It uses colors to show relationships.

Instead of assigning a label to a group, the output displays a group with a specified color. Below all average salaries that are 15000 or lower are displayed in green, the others are presented in red.

Traffic Lighting was introduced in the Proc Tabulate chapter with an example.

```

Program Editor - reporttraffic
Command ==>
00001 proc format;
00002   value $statfmt
00003     'D' = 'Divorced'
00004     'M' = 'Married'
00005     'W' = 'Widowed'
00006     'S' = 'Single'
00007     'SEP' = 'Separated';
00008
00009   value $genfmt
00010     'F' = 'Female'
00011     'M' = 'Male';
00012
00013   value salfmt
00014     low-15000 = 'cx006600'
00015     other = 'red';
00016 run;
00017
00018 title "Traffic Lighting in Proc Report ";
00019 ods html file = "c:\temp\report.html";
00020
00021 proc report data = saved.demograp nowindows;
00022   column gender status salary;
00023   define status / group format = $statfmt.;
00024   define gender / group format = $genfmt.;
00025   define salary / mean format = dollar10.2
00026     style = {foreground = salfmt.};
00027 run;
00028
00029 ods html close;

```

Traffic Lighting in Proc Report

gender	status	salary
Female	Divorced	\$8,000.00
	Married	\$10,123.06
	Separated	\$18,000.00
	Single	\$8,600.00
	Widowed	\$30,000.00
Male	Divorced	\$10,000.00
	Married	\$19,716.67
	Separated	\$12,000.00
	Single	\$3,560.00

The values of 19716.67, 18000, and 30000 are in red, the rest are in green.

Introduction to Data _Null_

ODS syntax can be used to output data values to the Listing window by using `_ODS_` in the Put statement.

A simple example of Data _Null_ output is shown below.

```

Command ==>
00001 data _null_;
00002   set saved.demog;
00003   file print ods;
00004   put _ods_;
00005 run;
00006

```

The results of the syntax are shown in the Results window.

Results

- Results
 - Datasep
 - FilePrint4
 - FilePrint4
 - FilePrint4

Additional syntax can be used to produce more precise output. The following syntax subsets the data using the Keep option. Note that the output resembles a Proc Print statement.

```

Command ==>
00001 title "Display of Data from Saved.Demog";
00002 title2 " ";
00003
00004 data _null_;
00005   set saved.demog;
00006   (keep = name gender salary grade);
00007   file print ods;
00008   put _ods_;
00009 run;
00010

```

Display of Data from Saved.Demog

NAME	GENDER	SALARY	GRADE
Dave Derry	M	13592	low
Julia Pendlebury	F	8870	low
Norman Harvey	M	12672	low
Harold Hicks	M	23760	high
Mary Molesworth	F	10512	low
Bob Bobington	M	7520	low
Freda Bamford	F	28512	high
Julia Kidd	F	14840	low
Helen Cinderford,	F	47520	high
Mary Chapel	F	23760	high
Jennifer Dawson	F	20592	high
Julio Jennings	F	13760	low
Shirley Walters	F	47520	high
Diane Duley	F	13840	low

Enclosing the syntax in ODS HTML statements creates browser-ready documents.

```

Command ==>
00001 ods html body="a:\demog.html";
00002 title "Display of Data from Saved.Demog";
00003 title2 " ";
00004
00005 data _null_;
00006   set saved.demog;
00007   (keep = name gender salary grade);
00008   file print ods;
00009   put _ods_;
00010 run;
00011 ods html close;
00012

```

Display of Data from Saved.Demog

NAME	GENDER	SALARY	GRADE
Dave Derry	M	13592	low
Julia Pendlebury	F	8870	low
Norman Harvey	M	12672	low
Harold Hicks	M	23760	high
Mary Molesworth	F	10512	low
Bob Bobington	M	7520	low
Freda Bamford	F	28512	high

Modifying the File Statement

The File Statement can be used to request various enhancements.

Consider another way to subset the data without using the Keep option.

```

Command ==>
00001 ods html body="a:\demog.html";
00002 title "Display of Data from Saved.Demog";
00003 title2 " ";
00004
00005 data _null_;
00006   set saved.demog;
00007   file print
00008     ods=(variables=(name gender salary grade));
00009   put _ods_;
00010 run;
00011 ods html close;
00012

```

Display of Data from Saved.Demog

NAME	GENDER	SALARY	GRADE
Dave Derry	M	13592	low
Julia Pendlebury	F	8870	low
Norman Harvey	M	12672	low
Harold Hicks	M	23760	high
Mary Molesworth	F	10512	low
Bob Bobington	M	7520	low
Freda Bamford	F	28512	high

Now consider a full-scale modification of the File statement using ODS= suboptions.

```

Command ==>
00001 proc format;
00002   value $sexfmt "M" = "Male" "F" = "Female";
00003 run;
00004
00005 ods html body="a:\demog.html";
00006 title "Display of Data from Saved.Demog";
00007 title2 " ";
00008
00009 data _null_;
00010   set saved.demog;
00011   file print
00012     ods=(variables=(name (label="Employee")
00013                       gender (label="Gender"
00014                               format=$sexfmt.)
00015                       salary (label="Annual Salary"
00016                               format=dollar12.2)
00017                       grade (label="Grade"))));
00018   put _ods_;
00019 run;
00020 ods html close;
00021

```

Results Viewer - a:\demo\demo.html

Display of Data from Saved.Demog

Employee	Gender	Annual Salary	Grade
Dave Derry	Male	\$13,592.00	low
Julia Pendlebury	Female	\$8,870.00	low
Norman Harvey	Male	\$12,672.00	low
Harold Hicks	Male	\$23,760.00	high
Mary Molesworth	Female	\$10,512.00	low
Bob Bobington	Male	\$7,520.00	low
Freda Bamford	Female	\$28,512.00	high

Display of Data from Saved.Demog

Employee	Gender	Earning Scale
Dave Derry	Male	low
Julia Pendlebury	Female	low
Norman Harvey	Male	low
Harold Hicks	Male	\$23,760.00
Mary Molesworth	Female	low
Bob Bobington	Male	low
Freda Bamford	Female	\$28,512.00

We can create the same output using a Label statement.

Combining Columns of Data

The syntax holds interesting possibilities for combining separate data set columns into the same output column.

Let's combine the Salary and Grade variables as follows:

- If the individual's Grade holds a value of low, show the Grade value.
- Otherwise, show the Salary value in the column.

```

Program Editor - (Untitled)
Command ==>
00001 ods html file="a:\report.html";
00002
00003 data _null_;
00004   set saved.demog
00005   (keep=name gender salary grade);
00006   where salary gt 0;
00007   file print
00008   ods=(variables=(name (label="Employee")
00009                   gender (label="Gender"
00010                          format=$sexfmt.)
00011                   grade (label="Earning Scale")));
00012   if grade = "low" then put _ods_;
00013   else put _ods_ @3 salary;
00014 run;
00015
00016 ods html close;
00017
nnn17

```

```

Program Editor - (Untitled)
Command ==>
00001 ods html file="a:\report.html";
00002
00003 data _null_;
00004   length grade $ 15;
00005   set saved.demog
00006   (keep=name gender salary grade);
00007   where salary gt 0;
00008   c_salary = put(salary, dollar12.2);
00009   file print
00010   ods=(variables=(name gender grade));
00011   label name = "Employee's Name"
00012         gender = "Gender"
00013         grade = "Income Level";
00014   format gender $sexfmt.;
00015   if grade = "low" then put _ods_;
00016   else put _ods_ @"grade" c_salary;
00017 run;
00018
00019 ods html close;
00020
nnn20

```

Results Viewer - a:\report.html

Display of Data from Saved.Demog

Employee	Gender	Earning Scale
Dave Derry	Male	low
Julia Pendlebury	Female	low
Norman Harvey	Male	low
Harold Hicks	Male	23760
Mary Molesworth	Female	low
Bob Bobington	Male	low
Freda Bamford	Female	28512

Additional data step syntax can enhance presentation of output.

```

Program Editor - (Untitled)
Command ==>
00001 ods html file="a:\report.html";
00002
00003 data _null_;
00004   length grade $ 15;
00005   set saved.demog
00006   (keep=name gender salary grade);
00007   where salary gt 0;
00008   c_salary = put(salary, dollar12.2);
00009   file print
00010   ods=(variables=(name (label="Employee")
00011                   gender (label="Gender"
00012                          format=$sexfmt.)
00013                   grade (label="Earning Scale")));
00014   if grade = "low" then put _ods_;
00015   else put _ods_ @3 c_salary;
00016 run;
00017
00018 ods html close;
00019
nnn19

```