Version 9 ODS - Intermediate

Prepared by

DESTINY CORPORATION

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

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.

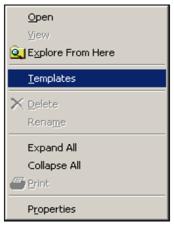
Inspecting Templates

The two templates work together, but have different functionalities.

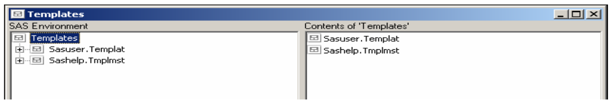
Both templates may be examined, edited, or created from scratch.

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.

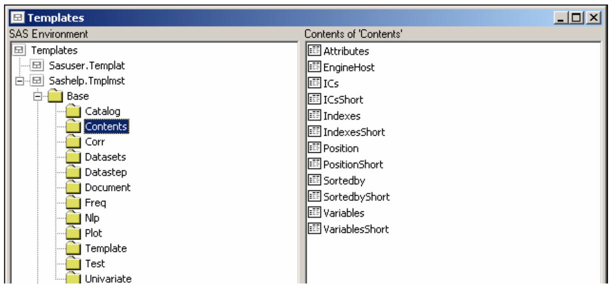


Notice that the Templates Folder has two subfolders:

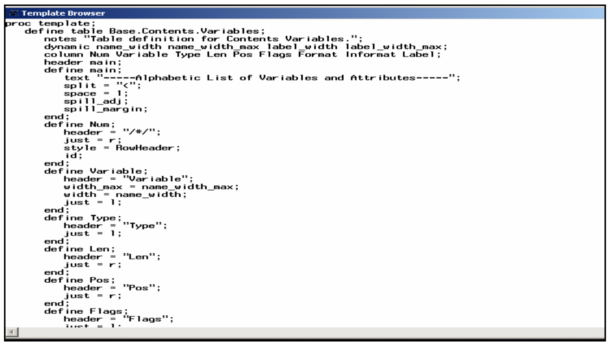
- Sasuser.Templat
- Sashelp.Tmplmst.

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.



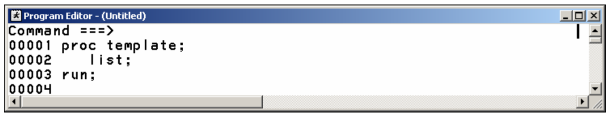
Double clicking on any template in the right pane will open a Template Browser window displaying the syntax of the table template.



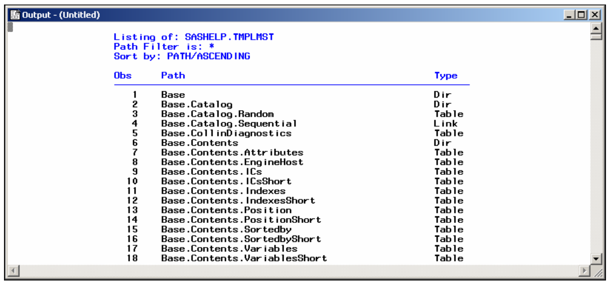
```
proc template;
   define table Base.Contents.Variables;
      notes "Table definition for Contents Variables.";
      dynamic name_width name_width_max label_width label_width_max;
      column Num Variable Type Len Pos Flags Format Informat Label;
      header main;
      define main;
         text "-----Alphabetic List of Variables and Attributes-----";
         split = "<";
         space = 1;
         spill_adj;
         spill_margin;
      end;
      define Num;
         header = "/#/";
         just = r;
         style = RowHeader;
         id;
      end;
      define Variable;
         header = "Variable";
         width_max = name_width_max;
         width = name_width;
         just = l;
      end;
      define Type;
         header = "Type";
         just = l;
      end;
      define Len;
         header = "Len";
         just = r;
      end;
      define Pos;
         header = "Pos";
         just = r;
      end;
      define Flags;
         header = "Flags";
```

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

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



```
Command ===>
proc template;
   list;
run;
```



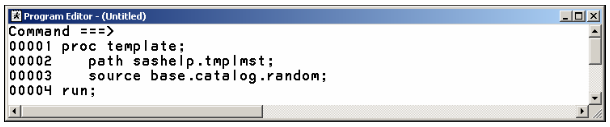
```
Listing of: SASHELP.TMPLMST
Path Filter is: *
Sort by: PATH/ASCENDING

Obs    Path                                  Type
-------------------------------------------------
  1    Base                                  Dir
  2    Base.Catalog                          Dir
  3    Base.Catalog.Random                   Table
  4    Base.Catalog.Sequential               Link
  5    Base.CollinDiagnostics                Table
  6    Base.Contents                         Dir
  7    Base.Contents.Attributes              Table
  8    Base.Contents.EngineHost              Table
  9    Base.Contents.ICs                     Table
 10    Base.Contents.ICsShort                Table
 11    Base.Contents.Indexes                 Table
 12    Base.Contents.IndexesShort            Table
 13    Base.Contents.Position                Table
 14    Base.Contents.PositionShort           Table
 15    Base.Contents.Sortedby                Table
 16    Base.Contents.SortedbyShort           Table
 17    Base.Contents.Variables               Table
 18    Base.Contents.VariablesShort          Table
```

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

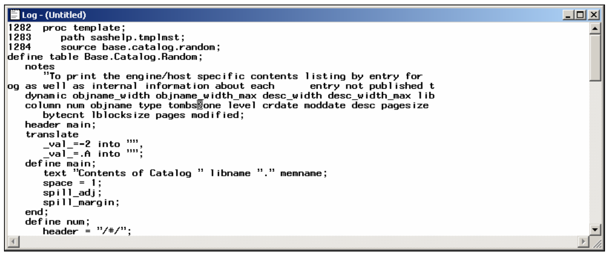
Use any full-path or partial-path information such as:

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

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

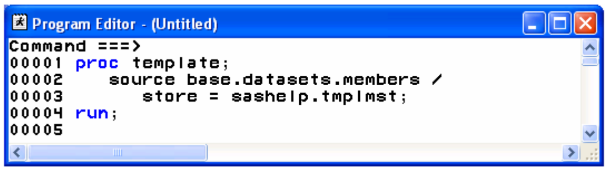


```
Command ===>
proc template;
   path sashelp.tmplmst;
   source base.catalog.random;
run;
```



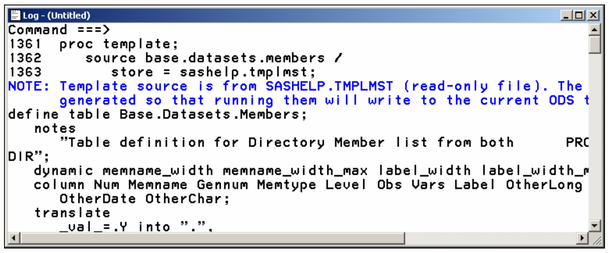
```
proc template;
   path sashelp.tmplmst;
   source base.catalog.random;
define table Base.Catalog.Random;
   notes
      "To print the engine/host specific contents listing by entry for
og as well as internal information about each      entry not published t
   dynamic objname_width objname_width_max desc_width desc_width_max lib
   column num objname type tombstone level crdate moddate desc pagesize
      bytecnt lblocksize pages modified;
   header main;
   translate
      _val_=-2 into "",
      _val_=.A into "";
   define main;
      text "Contents of Catalog " libname "." memname;
      space = 1;
      spill_adj;
      spill_margin;
   end;
   define num;
      header = "/#/";
```

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



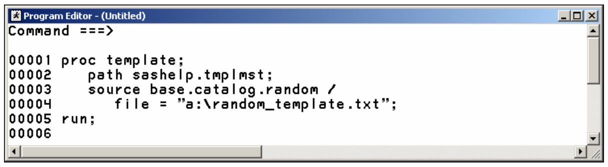
```
Command ===>
proc template;
   source base.datasets.members /
      store = sashelp.tmplmst;
run;

```

This program produces the following Log results.

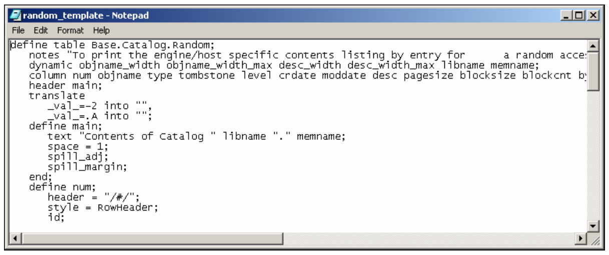


```
Command ===>
proc template;
   source base.datasets.members /
      store = sashelp.tmplmst;
NOTE: Template source is from SASHELP.TMPLMST (read-only file). The
      generated so that running them will write to the current ODS t
define table Base.Datasets.Members;
   notes
      "Table definition for Directory Member list from both      PRC
DIR";
   dynamic memname_width memname_width_max label_width label_width_m
   column Num Memname Gennum Memtype Level Obs Vars Label OtherLong
      OtherDate OtherChar;
   translate
      _val_=.Y into ".",
```

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



```
Command ===>

proc template;
   path sashelp.tmplmst;
   source base.catalog.random /
      file = "a:\random_template.txt";
run;

```



```
define table Base.Catalog.Random;
   notes "To print the engine/host specific contents listing by entry for      a random acce
   dynamic objname_width objname_width_max desc_width desc_width_max libname memname;
   column num objname type tombstone level crdate moddate desc pagesize blocksize blockcnt by
   header main;
   translate
      _val_=-2 into "",
      _val_=.A into "";
   define main;
      text "Contents of Catalog " libname "." memname;
      space = 1;
      spill_adj;
      spill_margin;
   end;
   define num;
      header = "/#/";
      style = RowHeader;
      id;
```

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

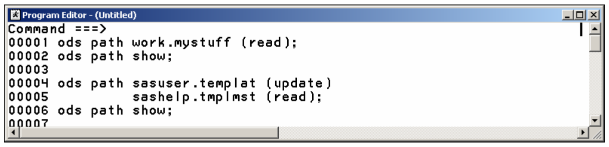
The Sashelp.Tmplmst store holds templates provided by the SAS Institute.

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



```
Command ===>
ods path work.mystuff (read);
ods path show;

ods path sasuser.templat (update)
         sashelp.tmplmst (read);
ods path show;
```

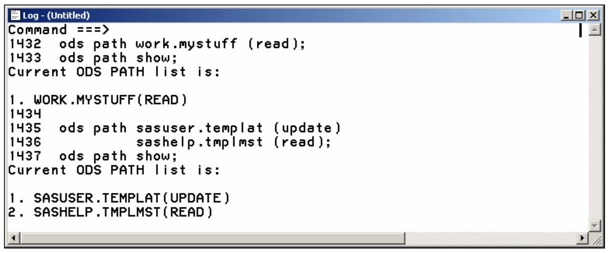
The ODS Path statement has two roles:

- 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.

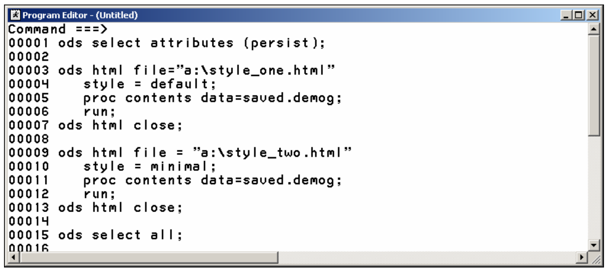


```
Command ===>
ods path work.mystuff (read);
ods path show;
Current ODS PATH list is:

1. WORK.MYSTUFF(READ)

ods path sasuser.templat (update)
         sashelp.tmplmst (read);
ods path show;
Current ODS PATH list is:

1. SASUSER.TEMPLAT(UPDATE)
2. SASHELP.TMPLMST(READ)
```

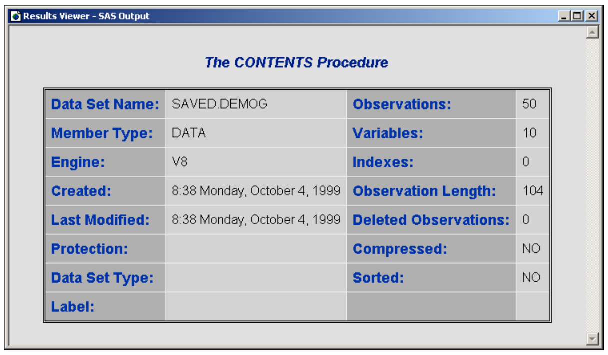
Table vs. Style Templates

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.

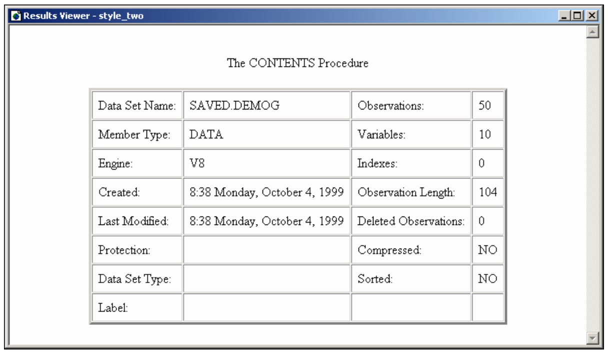


```
Command ===>
ods select attributes (persist);

ods html file="a:\style_one.html"
   style = default;
   proc contents data=saved.demog;
   run;
ods html close;

ods html file = "a:\style_two.html"
   style = minimal;
   proc contents data=saved.demog;
   run;
ods html close;

ods select all;
```



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.

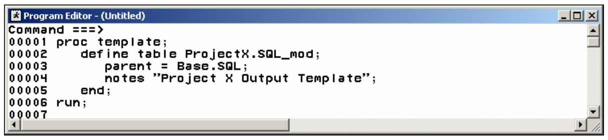
The programmer must edit or create the one that handles the aspect of the output to be manipulated.

Editing a Table Template

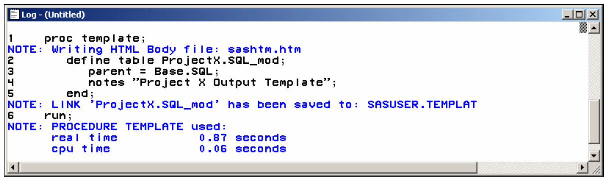
Templates can be edited or created from scratch.

It is easier to edit an existing template.

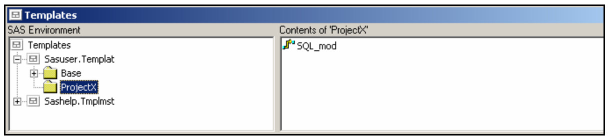
In either case, templates not provided by SAS are not located in the Sashelp.Tmplmst store.

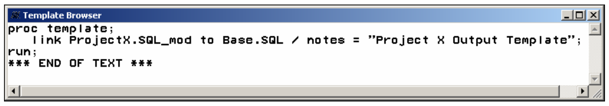


```
Command ===>
proc template;
   define table ProjectX.SQL_mod;
      parent = Base.SQL;
      notes "Project X Output Template";
   end;
run;
```



```
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           0.87 seconds
      cpu time            0.06 seconds
```





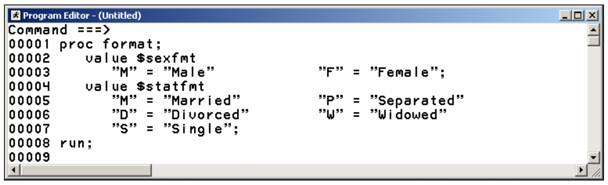
```
proc template;
   link ProjectX.SQL_mod to Base.SQL / notes = "Project X Output Template";
run;
*** END OF TEXT ***
```

Creating a Table Template

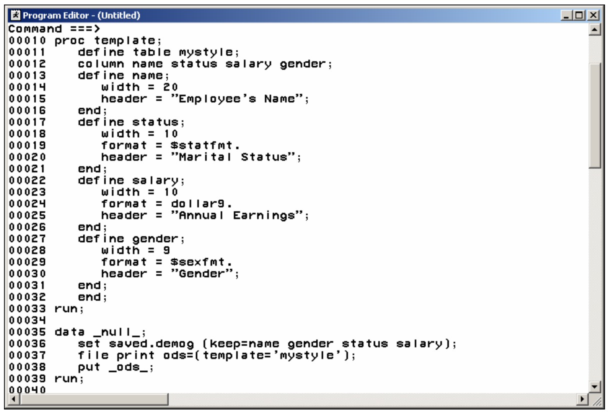
Proc Template can be used to edit and create a template.

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

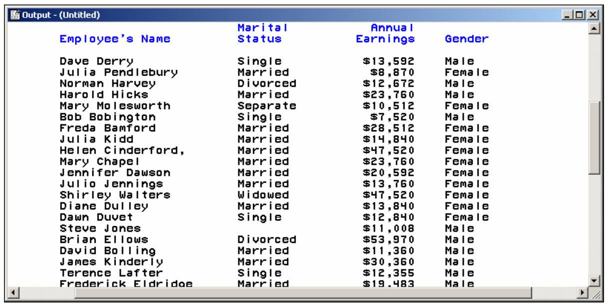
The following example shows how to create a new template for specific output created through Data _Null_ Step.



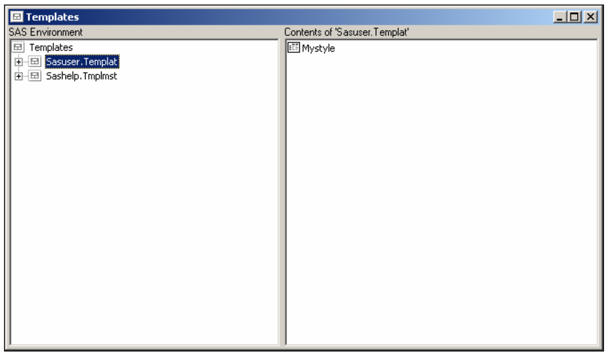
```
Command ===>
proc format;
   value $sexfmt
      "M" = "Male"              "F" = "Female";
   value $statfmt
      "M" = "Married"           "P" = "Separated"
      "D" = "Divorced"          "W" = "Widowed"
      "S" = "Single";
run;
```



```
Command ===>
proc template;
   define table mystyle;
   column name status salary gender;
   define name;
      width = 20
      header = "Employee's Name";
   end;
   define status;
      width = 10
      format = $statfmt.
      header = "Marital Status";
   end;
   define salary;
      width = 10
      format = dollar9.
      header = "Annual Earnings";
   end;
   define gender;
      width = 9
      format = $sexfmt.
      header = "Gender";
   end;
   end;
run;

data _null_;
   set saved.demog (keep=name gender status salary);
   file print ods=(template='mystyle');
   put _ods_;
run;
```



```
                         Marital         Annual
 Employee's Name         Status         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         Separate        $10,512    Female
 Bob Bobington           Single           $7,520    Male
 Freda Bamford           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
 Julio Jennings          Married         $13,760    Female
 Shirley Walters         Widowed         $47,520    Female
 Diane Dulley            Married         $13,840    Female
 Dawn Duvet              Single          $12,840    Female
 Steve Jones                             $11,008    Male
 Brian Ellows            Divorced        $53,970    Male
 David Bolling           Married         $11,360    Male
 James Kinderly          Married         $30,360    Male
 Terence Lafter          Single          $12,355    Male
 Frederick Eldridge      Married         $19,483    Male
```

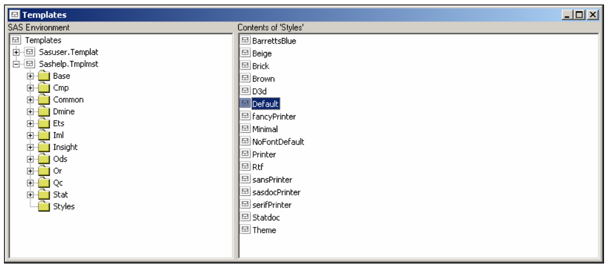


Proc Template gives us the ability to create specific structures.

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

ODS Style Templates

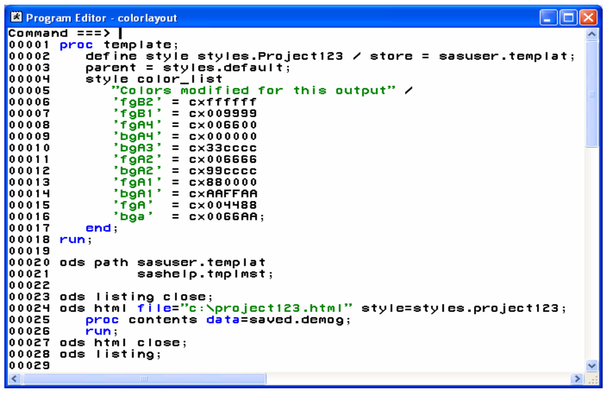
The basis for all ODS styles is the Styles.Default template.



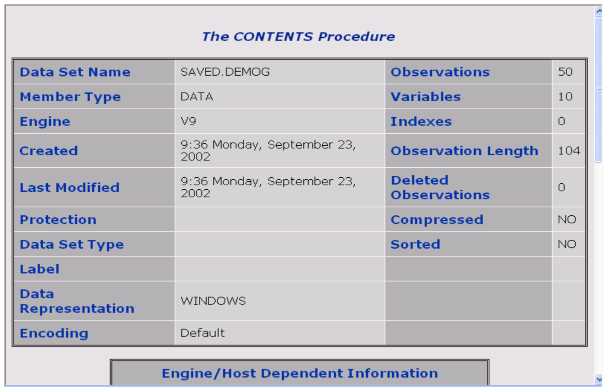
The Style templates can be modified as needed by the user.

A few illustrations are displayed in the following sections.

Let's change the color layout.



```
Command ===>
00001 proc template;
00002     define style styles.Project123 / store = sasuser.templat;
00003     parent = styles.default;
00004     style color_list
00005         "Colors modified for this output" /
00006         'fgB2'  = cxffffff
00007         'fgB1'  = cx009999
00008         'fgA4'  = cx006600
00009         'bgA4'  = cx000000
00010         'bgA3'  = cx33cccc
00011         'fgA2'  = cx006666
00012         'bgA2'  = cx99cccc
00013         'fgA1'  = cx880000
00014         'bgA1'  = cxAAFFAA
00015         'fgA'   = cx004488
00016         'bga'   = cx0066AA;
00017     end;
00018 run;
00019
00020 ods path sasuser.templat
00021          sashelp.tmplmst;
00022
00023 ods listing close;
00024 ods html file="c:\project123.html" style=styles.project123;
00025     proc contents data=saved.demog;
00026     run;
00027 ods html close;
00028 ods listing;
00029
```



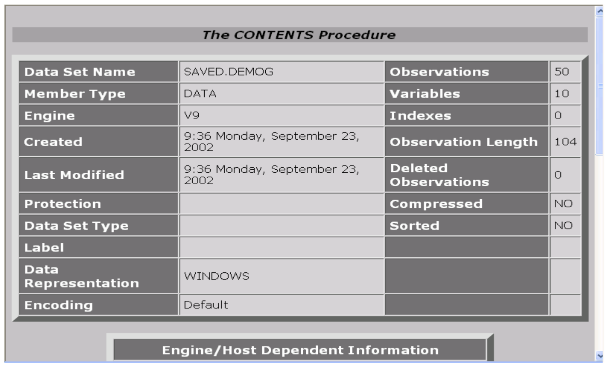
The CONTENTS Procedure

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:



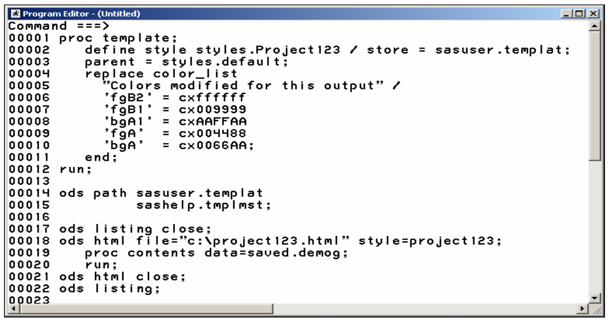
The CONTENTS Procedure

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		

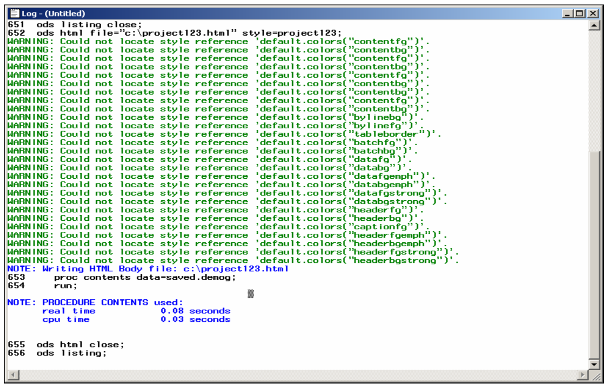
Engine/Host Dependent Information

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.



```
Command ===>
00001 proc template;
00002     define style styles.Project123 / store = sasuser.templat;
00003     parent = styles.default;
00004     replace color_list
00005         "Colors modified for this output" /
00006         'fgB2' = cxffffff
00007         'fgB1' = cx009999
00008         'bgA1' = cxAAFFAA
00009         'fgA'  = cx004488
00010         'bgA'  = cx0066AA;
00011     end;
00012 run;
00013
00014 ods path sasuser.templat
00015          sashelp.tmplmst;
00016
00017 ods listing close;
00018 ods html file="c:\project123.html" style=project123;
00019     proc contents data=saved.demog;
00020     run;
00021 ods html close;
00022 ods listing;
00023
```



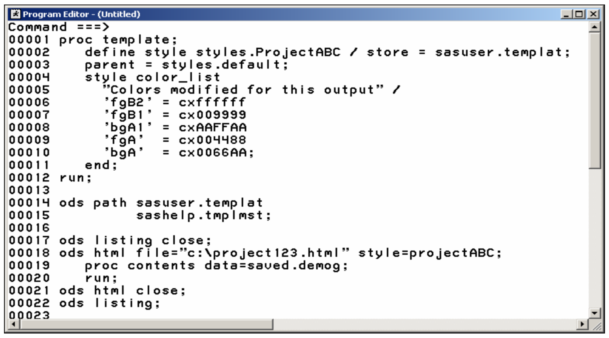
```
651  ods listing close;
652  ods html file="c:\project123.html" style=project123;
WARNING: Could not locate style reference 'default.colors("contentfg")'.
WARNING: Could not locate style reference 'default.colors("contentbg")'.
WARNING: Could not locate style reference 'default.colors("contentfg")'.
WARNING: Could not locate style reference 'default.colors("contentbg")'.
WARNING: Could not locate style reference 'default.colors("contentfg")'.
WARNING: Could not locate style reference 'default.colors("contentbg")'.
WARNING: Could not locate style reference 'default.colors("contentbg")'.
WARNING: Could not locate style reference 'default.colors("contentfg")'.
WARNING: Could not locate style reference 'default.colors("contentbg")'.
WARNING: Could not locate style reference 'default.colors("bylinebg")'.
WARNING: Could not locate style reference 'default.colors("bylinefg")'.
WARNING: Could not locate style reference 'default.colors("tableborder")'.
WARNING: Could not locate style reference 'default.colors("batchfg")'.
WARNING: Could not locate style reference 'default.colors("batchbg")'.
WARNING: Could not locate style reference 'default.colors("datafg")'.
WARNING: Could not locate style reference 'default.colors("databg")'.
WARNING: Could not locate style reference 'default.colors("datafgemph")'.
WARNING: Could not locate style reference 'default.colors("databgemph")'.
WARNING: Could not locate style reference 'default.colors("datafgstrong")'.
WARNING: Could not locate style reference 'default.colors("databgstrong")'.
WARNING: Could not locate style reference 'default.colors("headerfg")'.
WARNING: Could not locate style reference 'default.colors("headerbg")'.
WARNING: Could not locate style reference 'default.colors("captionfg")'.
WARNING: Could not locate style reference 'default.colors("headerfgemph")'.
WARNING: Could not locate style reference 'default.colors("headerbgemph")'.
WARNING: Could not locate style reference 'default.colors("headerfgstrong")'.
WARNING: Could not locate style reference 'default.colors("headerbgstrong")'.
NOTE: Writing HTML Body file: c:\project123.html
653      proc contents data=saved.demog;
654      run;

NOTE: PROCEDURE CONTENTS used:
      real time           0.08 seconds
      cpu time            0.03 seconds


655  ods html close;
656  ods listing;
```

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

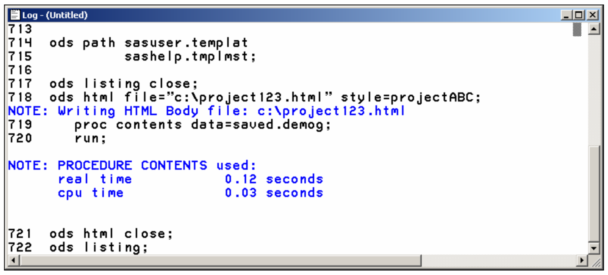
Simply changing the colors requiring modification does not work. SAS does not have a color reference for any color missing from the Replace statement.

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.



```
Command ===>
proc template;
   define style styles.ProjectABC / store = sasuser.templat;
   parent = styles.default;
   style color_list
      "Colors modified for this output" /
      'fgB2' = cxffffff
      'fgB1' = cx009999
      'bgA1' = cxAAFFAA
      'fgA'  = cx004488
      'bgA'  = cx0066AA;
   end;
run;

ods path sasuser.templat
         sashelp.tmplmst;

ods listing close;
ods html file="c:\project123.html" style=projectABC;
   proc contents data=saved.demog;
   run;
ods html close;
ods listing;
```



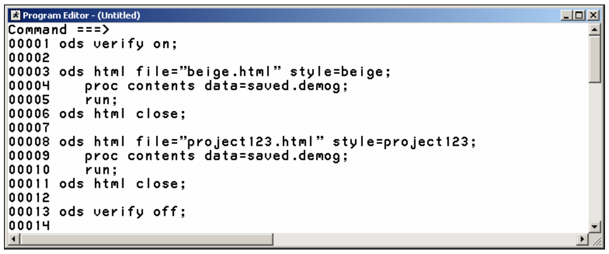
```
713
714   ods path sasuser.templat
               sashelp.tmplmst;
716
717   ods listing close;
718   ods html file="c:\project123.html" style=projectABC;
NOTE: Writing HTML Body file: c:\project123.html
719      proc contents data=saved.demog;
720      run;

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).

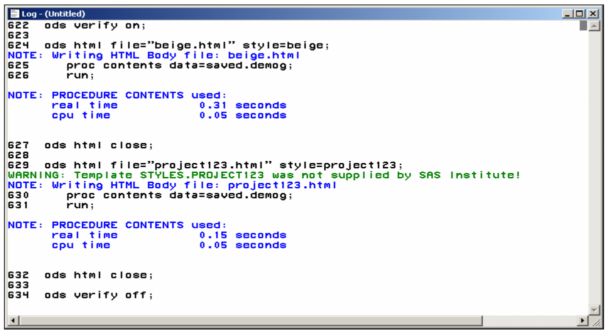
ODS VERIFY

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'.



```
Command ===>
ods verify on;

ods html file="beige.html" style=beige;
   proc contents data=saved.demog;
   run;
ods html close;

ods html file="project123.html" style=project123;
   proc contents data=saved.demog;
   run;
ods html close;

ods verify off;
```



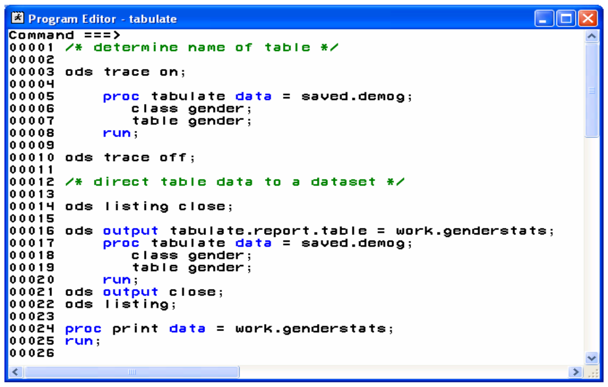
```
622   ods verify on;
623
624   ods html file="beige.html" style=beige;
NOTE: Writing HTML Body file: beige.html
625      proc contents data=saved.demog;
626      run;

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.

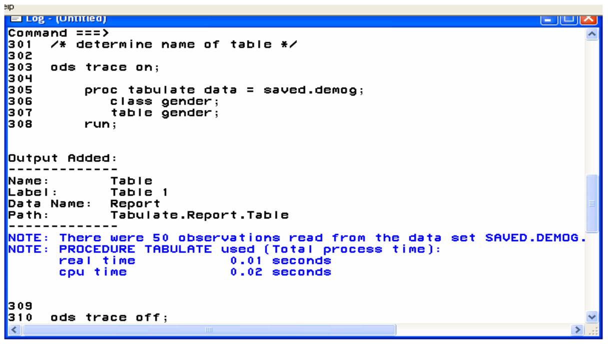
This means that the statistics are created using Proc Tabulate or Proc Means. Output is directed into a SAS data set as variables and observations.

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



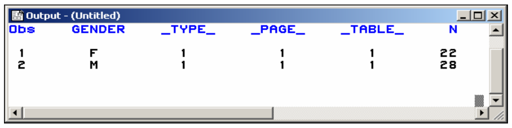
```
Command ===>
/* determine name of table */

ods trace on;

     proc tabulate data = saved.demog;
        class gender;
        table gender;
     run;

ods trace off;

/* direct table data to a dataset */

ods listing close;

ods output tabulate.report.table = work.genderstats;
     proc tabulate data = saved.demog;
        class gender;
        table gender;
     run;
ods output close;
ods listing;

proc print data = work.genderstats;
run;
```

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



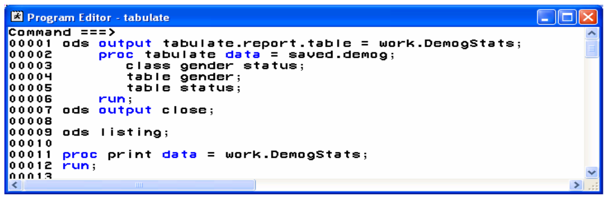
```
Command ===>
301   /* determine name of table */
302
303   ods trace on;
304
305        proc tabulate data = saved.demog;
306           class gender;
307           table gender;
308        run;


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.

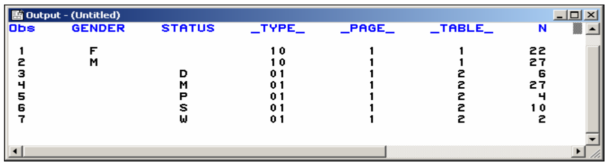


```
Obs    GENDER    _TYPE_    _PAGE_    _TABLE_     N

 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.



```
Command ===>
ods output tabulate.report.table = work.DemogStats;
     proc tabulate data = saved.demog;
          class gender status;
          table gender;
          table status;
     run;
ods output close;

ods listing;

proc print data = work.DemogStats;
run;
```

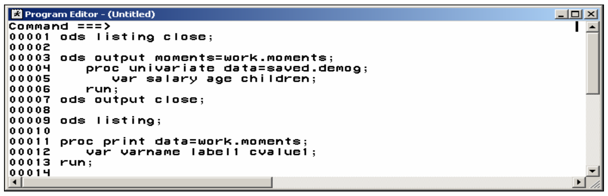


```
Obs    GENDER    STATUS    _TYPE_    _PAGE_    _TABLE_     N

 1       F                   10         1         1       22
 2       M                   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
```

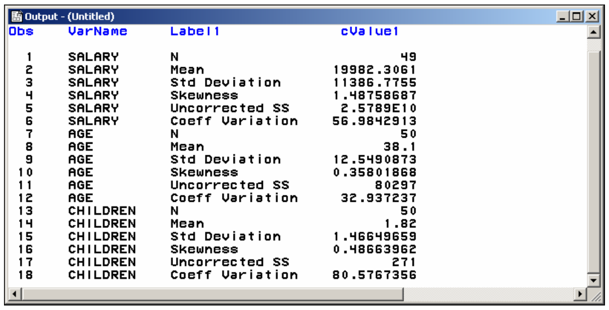
MATCH_ALL Option with Data Set Output

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.

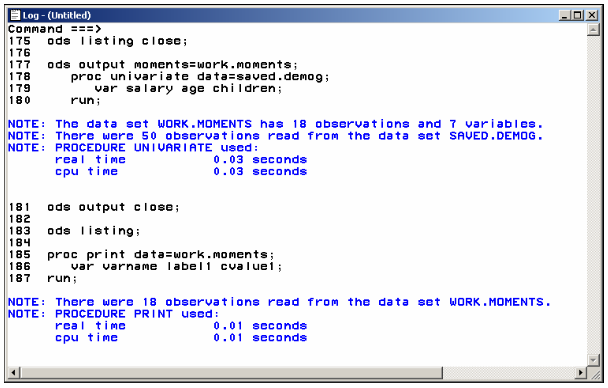


```
Command ===>
ods listing close;

ods output moments=work.moments;
   proc univariate data=saved.demog;
      var salary age children;
   run;
ods output close;

ods listing;

proc print data=work.moments;
   var varname label1 cvalue1;
run;
```



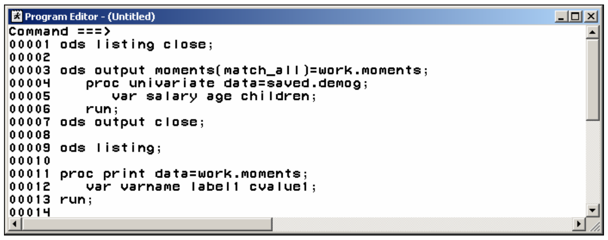
```
Obs    VarName     Label1                     cValue1

  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**.**

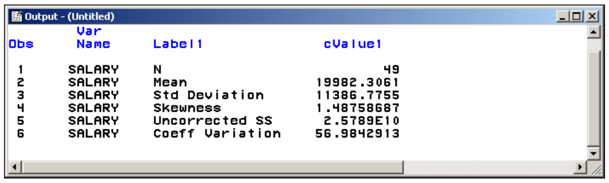


```
Command ===>
175  ods listing close;
176
177  ods output moments=work.moments;
178     proc univariate data=saved.demog;
179        var salary age children;
180     run;

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
```

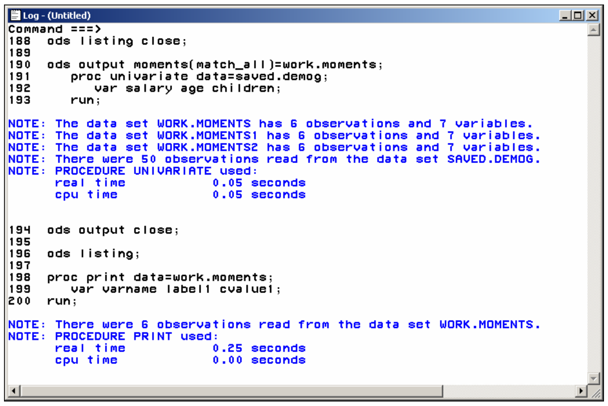
On the other hand, the MATCH_ALL option will separate each variable's data to flow into a different SAS dataset.



```
Command ===>
ods listing close;

ods output moments(match_all)=work.moments;
   proc univariate data=saved.demog;
      var salary age children;
   run;
ods output close;

ods listing;

proc print data=work.moments;
   var varname label1 cvalue1;
run;
```



```
         Var
Obs      Name      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
```

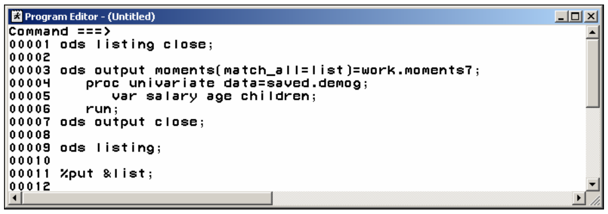


```
Command ===>
188  ods listing close;
189
190  ods output moments(match_all)=work.moments;
191     proc univariate data=saved.demog;
192        var salary age children;
193     run;

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
```

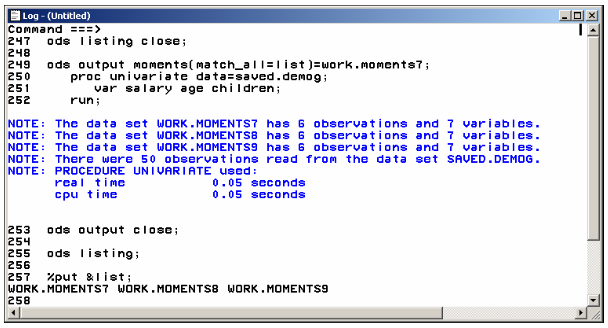
MATCH_ALL option with Macro Variable Output

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.



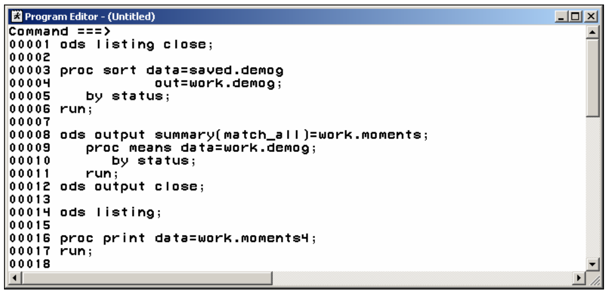
```
Command ===>
ods listing close;

ods output moments(match_all=list)=work.moments7;
   proc univariate data=saved.demog;
      var salary age children;
   run;
ods output close;

ods listing;

%put &list;
```



```
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.

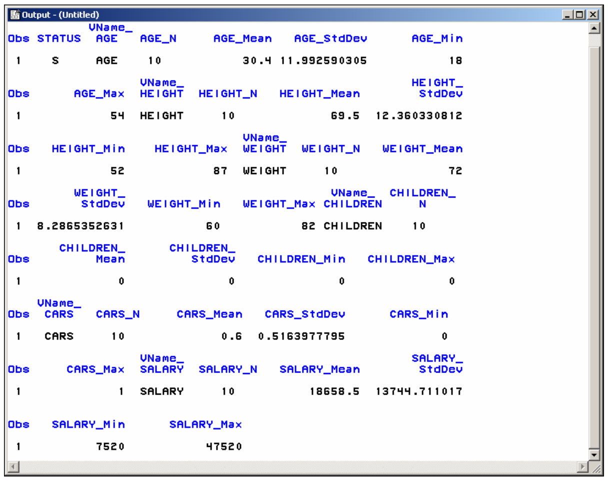


```
Command ===>
ods listing close;

proc sort data=saved.demog
              out=work.demog;
   by status;
run;

ods output summary(match_all)=work.moments;
   proc means data=work.demog;
      by status;
   run;
ods output close;

ods listing;

proc print data=work.moments4;
run;
```

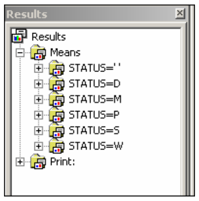
The distinct values of Status are ‘ ‘, D, M, P, S, W.

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

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



```
           VName_
Obs STATUS  AGE   AGE_N     AGE_Mean   AGE_StdDev      AGE_Min

 1    S     AGE    10           30.4 11.992590305           18

                  VName_                               HEIGHT_
Obs      AGE_Max  HEIGHT  HEIGHT_N   HEIGHT_Mean        StdDev

 1            54  HEIGHT     10             69.5  12.360330812

                                VName_
Obs   HEIGHT_Min    HEIGHT_Max  WEIGHT  WEIGHT_N   WEIGHT_Mean

 1            52            87  WEIGHT     10               72

         WEIGHT_                            VName_  CHILDREN_
Obs       StdDev   WEIGHT_Min   WEIGHT_Max CHILDREN     N

 1  8.2865352631           60           82 CHILDREN    10

       CHILDREN_      CHILDREN_
Obs         Mean         StdDev   CHILDREN_Min   CHILDREN_Max

 1             0              0              0              0

    VName_
Obs  CARS   CARS_N     CARS_Mean   CARS_StdDev      CARS_Min

 1   CARS     10             0.6  0.5163977795             0

                  VName_                               SALARY_
Obs     CARS_Max  SALARY  SALARY_N   SALARY_Mean        StdDev

 1             1  SALARY     10          18658.5  13744.711017


Obs   SALARY_Min      SALARY_Max

 1          7520           47520
```



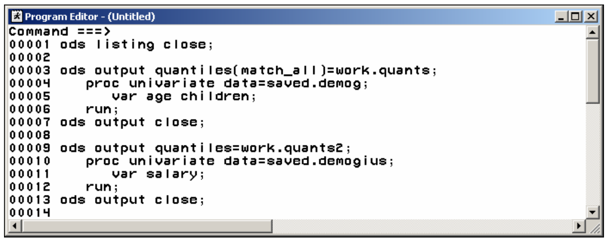
MATCH_ALL option with Persist=Proc

Consider the programming example below.

Two separate data sets are used in Proc univariate.

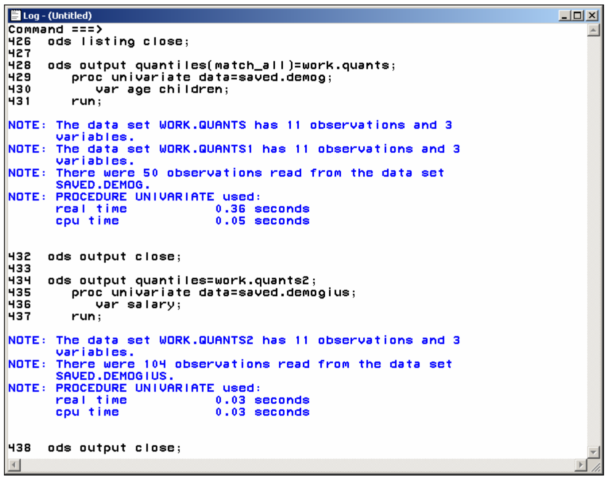
Three variables are referenced.

The resulting SAS data sets are work.quants, work.quants1, and work.quants2.



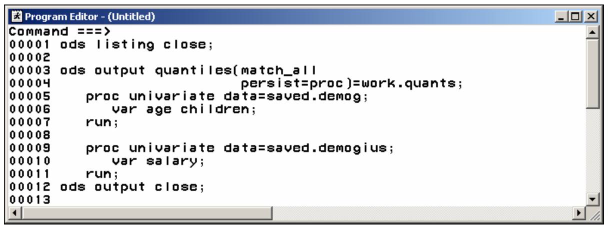
```
Command ===>
ods listing close;

ods output quantiles(match_all)=work.quants;
   proc univariate data=saved.demog;
      var age children;
   run;
ods output close;

ods output quantiles=work.quants2;
   proc univariate data=saved.demogius;
      var salary;
   run;
ods output close;
```

The Log window shows the output created.



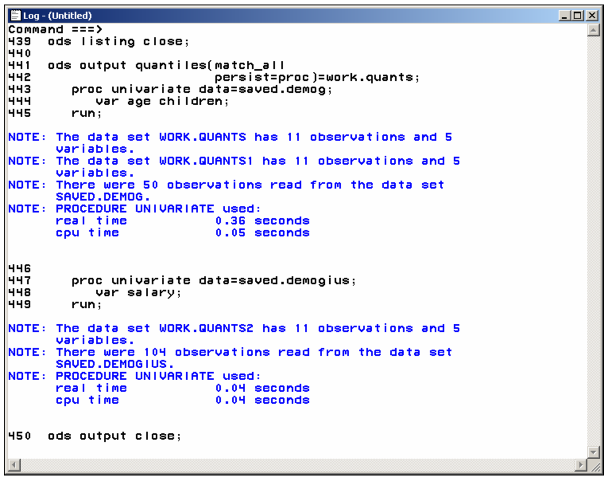
```
Command ===>
426  ods listing close;
427
428  ods output quantiles(match_all)=work.quants;
429     proc univariate data=saved.demog;
430        var age children;
431     run;

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.quants2;
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.



```
Command ===>
ods listing close;

ods output quantiles(match_all
                     persist=proc)=work.quants;
   proc univariate data=saved.demog;
      var age children;
   run;

   proc univariate data=saved.demogius;
      var salary;
   run;
ods output close;
```

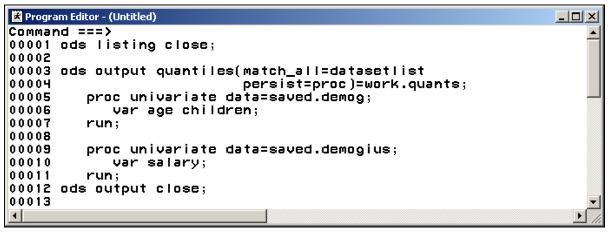
The Log window shows the creation of the same three SAS data sets.



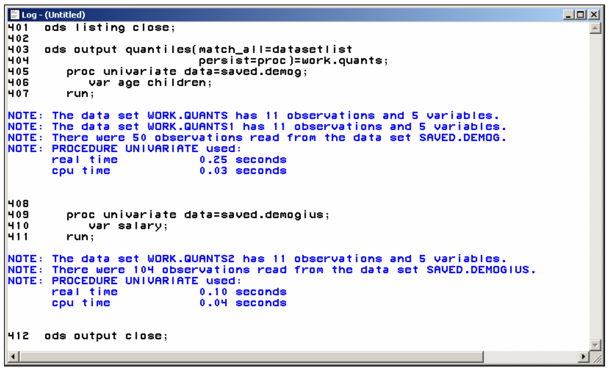
```
Command ===>
439  ods listing close;
440
441  ods output quantiles(match_all
442                       persist=proc)=work.quants;
443     proc univariate data=saved.demog;
444        var age children;
445     run;

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.



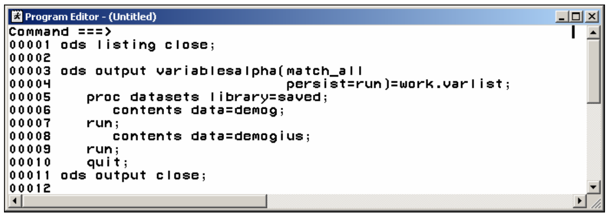
```
Command ===>
ods listing close;

ods output quantiles(match_all=datasetlist
                     persist=proc)=work.quants;
   proc univariate data=saved.demog;
      var age children;
   run;

   proc univariate data=saved.demogius;
      var salary;
   run;
ods output close;
```



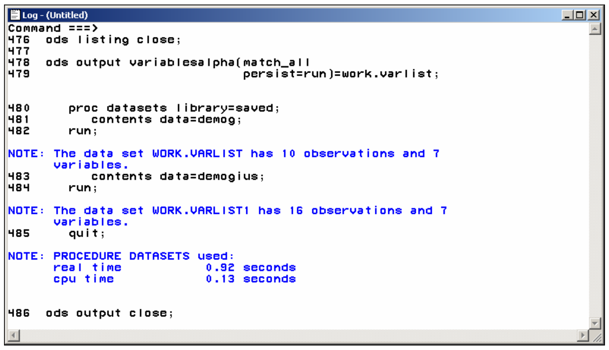
```
401  ods listing close;
402
403  ods output quantiles(match_all=datasetlist
404                       persist=proc)=work.quants;
405     proc univariate data=saved.demog;
406        var age children;
407     run;

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.



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



```
Command ===>
476  ods listing close;
477
478  ods output variablesalpha(match_all
479                            persist=run)=work.varlist;


480     proc datasets library=saved;
481        contents data=demog;
482     run;

NOTE: The data set WORK.VARLIST has 10 observations and 7
      variables.
483        contents data=demogius;
484     run;

NOTE: The data set WORK.VARLIST1 has 16 observations and 7
      variables.
485     quit;

NOTE: PROCEDURE DATASETS used:
      real time           0.92 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.

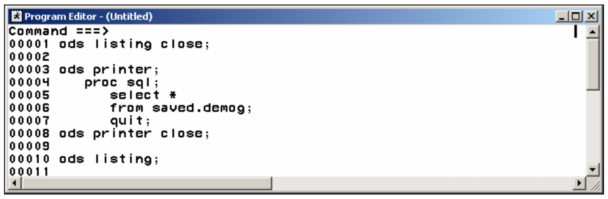
As illustrated earlier, the ODS statements used in routing output to a printer destination both open and close the request.

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.

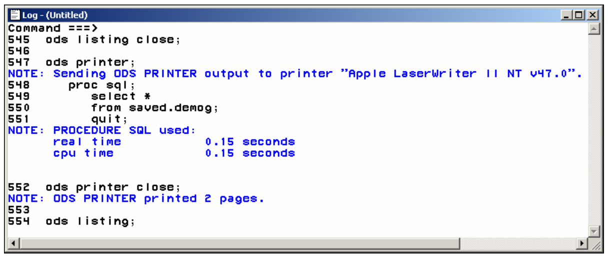
Direct Printing

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



```
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
```

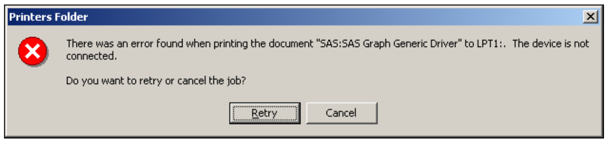
The Log window confirms the syntax action.



```
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.



There was an error found when printing the document "SAS:SAS Graph Generic Driver" to LPT1:. The device is not connected.

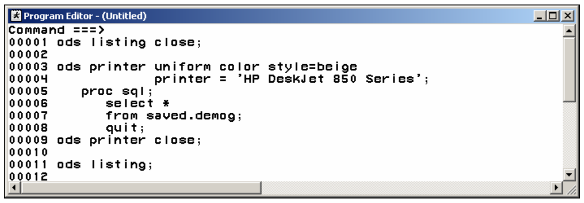
Do you want to retry or cancel the job?

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.



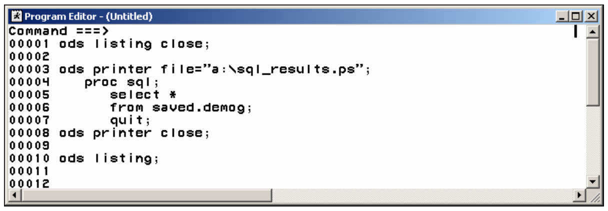
```
Command ===>
00001 ods listing close;
00002
00003 ods printer uniform color style=beige
00004             printer = 'HP DeskJet 850 Series';
00005    proc sql;
00006       select *
00007       from saved.demog;
00008       quit;
00009 ods printer close;
00010
00011 ods listing;
00012
```

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

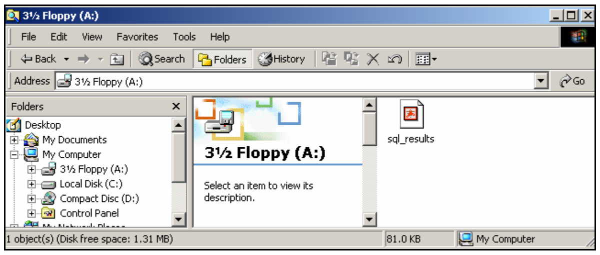
First, we will store the output as a Postscript file.

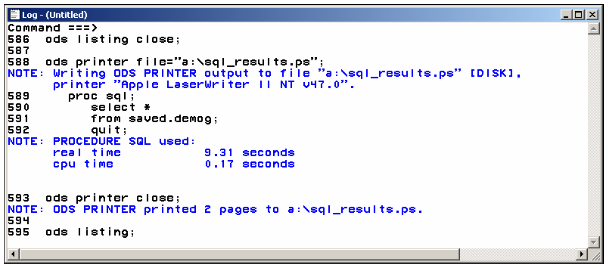


```
Command ===>
00001 ods listing close;
00002
00003 ods printer file="a:\sql_results.ps";
00004    proc sql;
00005       select *
00006       from saved.demog;
00007       quit;
00008 ods printer close;
00009
00010 ods listing;
00011
00012
```

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;
```





```
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 v47.0".
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;
```

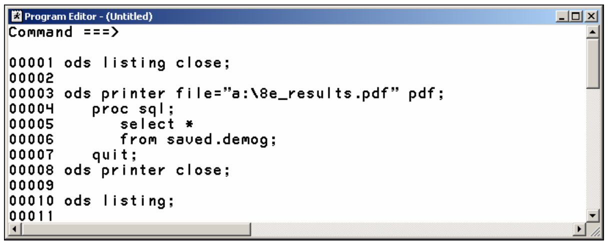
Printing to Disk - Portable Document Files

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

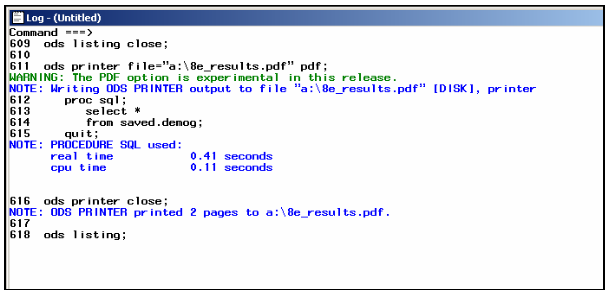
In SAS Version 8, generating a Postscript document using the syntax shown previously creates PDF output.

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

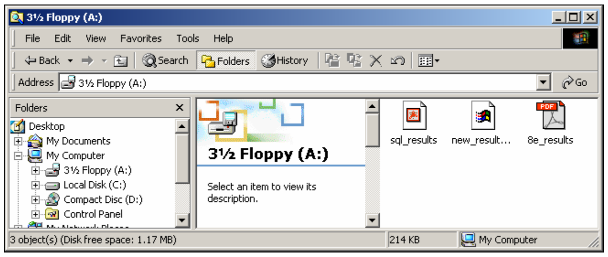
In SAS Version 8e (8.1), the process to create PDF is more direct.

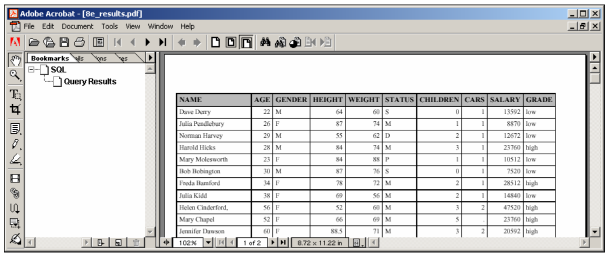


```
Command ===>

ods listing close;

ods printer file="a:\8e_results.pdf" pdf;
   proc sql;
       select *
       from saved.demog;
   quit;
ods printer close;

ods listing;
```



```
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;
```




Adobe Acrobat - [8e_results.pdf]
File Edit Document Tools View Window Help
Bookmarks
SQL
Query Results
102%
1 of 2
8.72 x 11.22 in

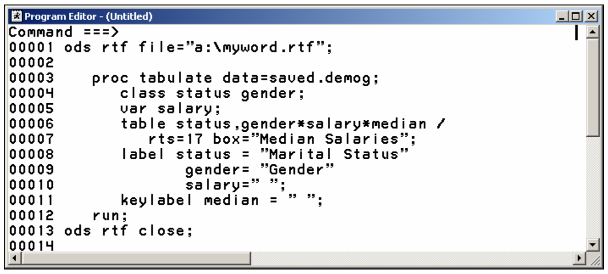
NAME	AGE	GENDER	HEIGHT	WEIGHT	STATUS	CHILDREN	CARS	SALARY	GRADE
Dave Derry	22	M	64	60	S	0	1	13592	low
Julia Pendlebury	26	F	87	74	M	1	1	8870	low
Norman Harvey	29	M	55	62	D	2	1	12672	low
Harold Hicks	28	M	84	74	M	3	1	23760	high
Mary Molesworth	23	F	84	88	P	1	1	10512	low
Bob Bobington	30	M	87	76	S	0	1	7520	low
Freda Bamford	34	F	78	72	M	2	1	28512	high
Julia Kidd	38	F	69	56	M	2	1	14840	low
Helen Cinderford,	56	F	52	60	M	3	2	47520	high
Mary Chapel	52	F	66	69	M	5	.	23760	high
Jennifer Dawson	60	F	88.5	71	M	3	2	20592	high

Printing to Disk - Rich Text Format

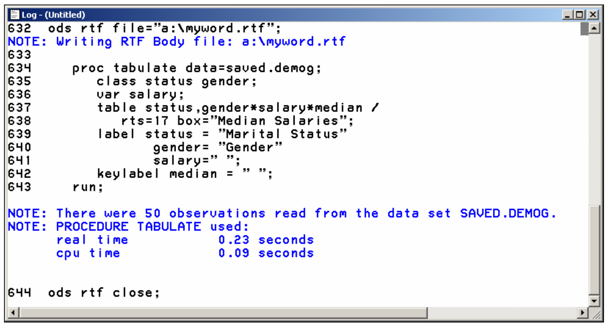
ODS can create Rich Text Format (*.rtf) for use in a word-processing application.

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

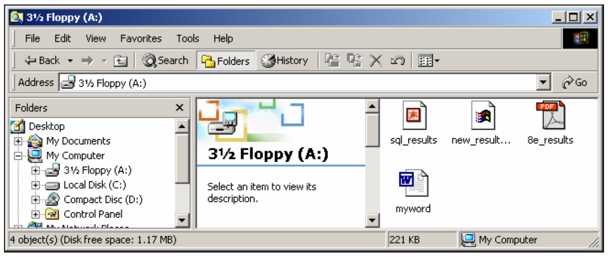
The results are satisfactory.

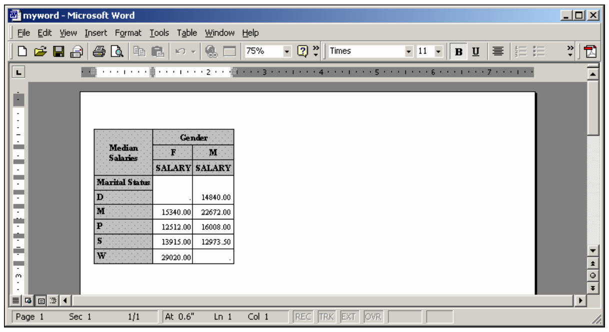


```
Command ===>
ods rtf file="a:\myword.rtf";

   proc tabulate data=saved.demog;
      class status gender;
      var salary;
      table status,gender*salary*median /
         rts=17 box="Median Salaries";
      label status = "Marital Status"
            gender= "Gender"
            salary=" ";
      keylabel median = " ";
   run;
ods rtf close;

```



```
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;
```



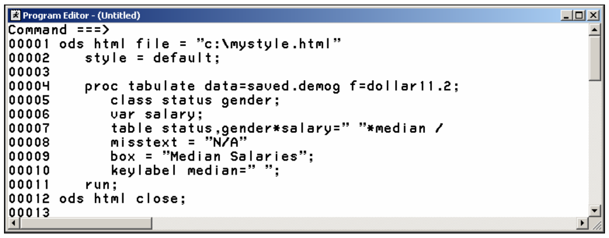


Median Salaries	Gender	
	F	M
	SALARY	SALARY
Marital Status		
D	.	14840.00
M	15340.00	22672.00
P	12512.00	16008.00
S	13915.00	12973.50
W	29020.00	.

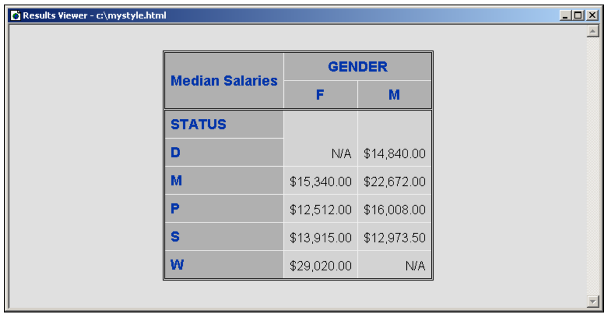
Using Styles to Control Colors and Fonts

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.

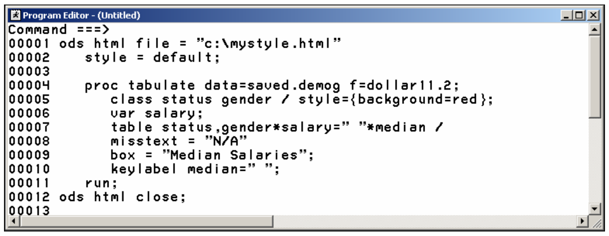


```
Command ===>
ods html file = "c:\mystyle.html"
   style = default;

   proc tabulate data=saved.demog f=dollar11.2;
      class status gender;
      var salary;
      table status,gender*salary=" "*median /
      misstext = "N/A"
      box = "Median Salaries";
      keylabel median=" ";
   run;
ods html close;
```

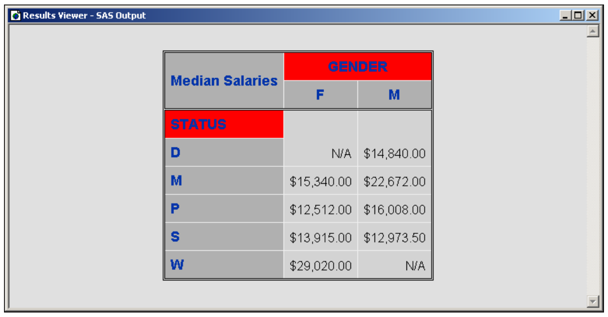


Median Salaries	GENDER	
	F	M
STATUS		
D	N/A	$14,840.00
M	$15,340.00	$22,672.00
P	$12,512.00	$16,008.00
S	$13,915.00	$12,973.50
W	$29,020.00	N/A

Using Style = { }

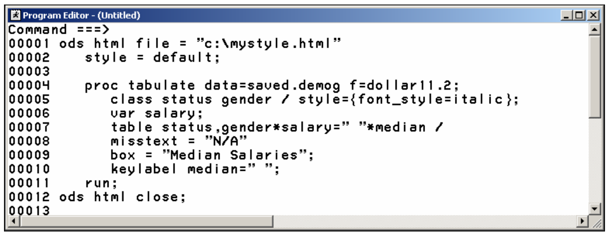


```
Command ===>
ods html file = "c:\mystyle.html"
   style = default;

   proc tabulate data=saved.demog f=dollar11.2;
      class status gender / style={background=red};
      var salary;
      table status,gender*salary=" "*median /
      misstext = "N/A"
      box = "Median Salaries";
      keylabel median=" ";
   run;
ods html close;
```

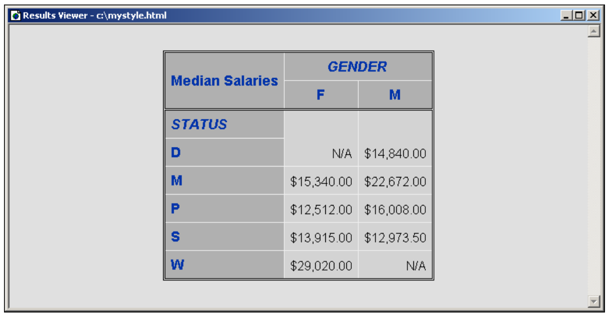


Median Salaries	GENDER	
	F	M
STATUS		
D	N/A	$14,840.00
M	$15,340.00	$22,672.00
P	$12,512.00	$16,008.00
S	$13,915.00	$12,973.50
W	$29,020.00	N/A

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



```
Command ===>
ods html file = "c:\mystyle.html"
   style = default;

   proc tabulate data=saved.demog f=dollar11.2;
      class status gender / style={font_style=italic};
      var salary;
      table status,gender*salary=" "*median /
      misstext = "N/A"
      box = "Median Salaries";
      keylabel median=" ";
   run;
ods html close;
```



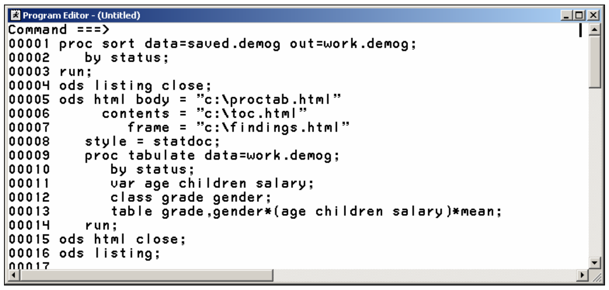
Median Salaries	GENDER	
	F	M
STATUS		
D	N/A	$14,840.00
M	$15,340.00	$22,672.00
P	$12,512.00	$16,008.00
S	$13,915.00	$12,973.50
W	$29,020.00	N/A

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 Proc Tabulate to Modify the Table of Contents

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



```
Command ===>
proc sort data=saved.demog out=work.demog;
   by status;
run;
ods listing close;
ods html body = "c:\proctab.html"
     contents = "c:\toc.html"
        frame = "c:\findings.html"
   style = statdoc;
   proc tabulate data=work.demog;
      by status;
      var age children salary;
      class grade gender;
      table grade,gender*(age children salary)*mean;
   run;
ods html close;
ods listing;
```

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.

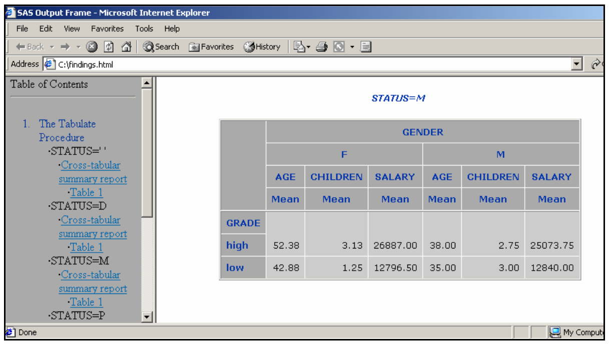


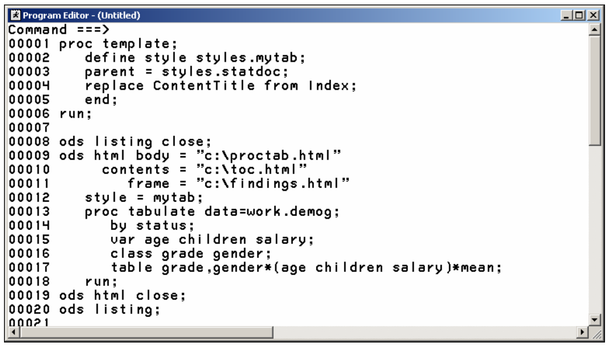
Table of Contents

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

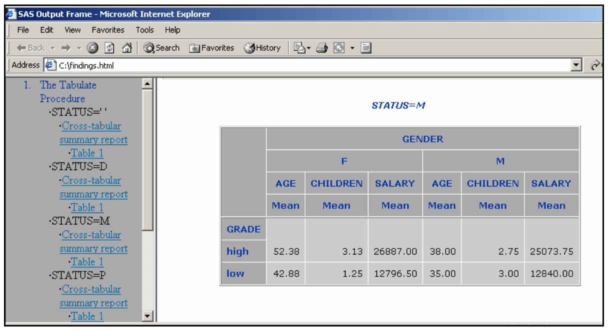
STATUS=M

	GENDER					
	F			M		
	AGE	CHILDREN	SALARY	AGE	CHILDREN	SALARY
	Mean	Mean	Mean	Mean	Mean	Mean
GRADE						
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 the string “Table of Contents”, submit syntax using the Replace statement.



```
Command ===>
proc template;
   define style styles.mytab;
   parent = styles.statdoc;
   replace ContentTitle from Index;
   end;
run;

ods listing close;
ods html body = "c:\proctab.html"
     contents = "c:\toc.html"
        frame = "c:\findings.html"
   style = mytab;
   proc tabulate data=work.demog;
      by status;
      var age children salary;
      class grade gender;
      table grade,gender*(age children salary)*mean;
   run;
ods html close;
ods listing;
```

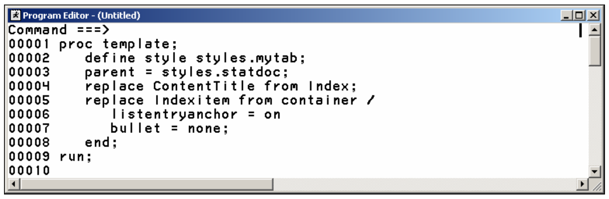


1. The Tabulate Procedure
   ·STATUS=' '
   ·Cross-tabular summary report
   ·Table 1
   ·STATUS=D
   ·Cross-tabular summary report
   ·Table 1
   ·STATUS=M
   ·Cross-tabular summary report
   ·Table 1
   ·STATUS=P
   ·Cross-tabular summary report
   ·Table 1

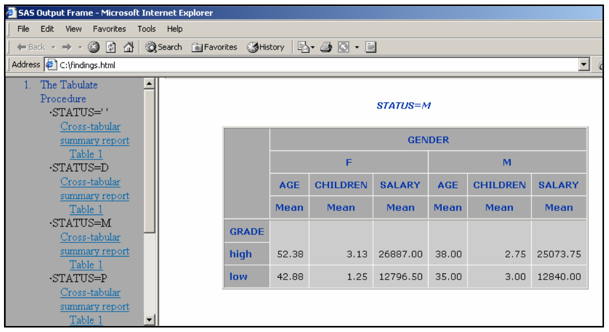
STATUS=M

	GENDER					
	F			M		
	AGE	CHILDREN	SALARY	AGE	CHILDREN	SALARY
	Mean	Mean	Mean	Mean	Mean	Mean
GRADE						
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 ===>
proc template;
   define style styles.mytab;
   parent = styles.statdoc;
   replace ContentTitle from Index;
   replace Indexitem from container /
      listentryanchor = on
      bullet = none;
   end;
run;
```

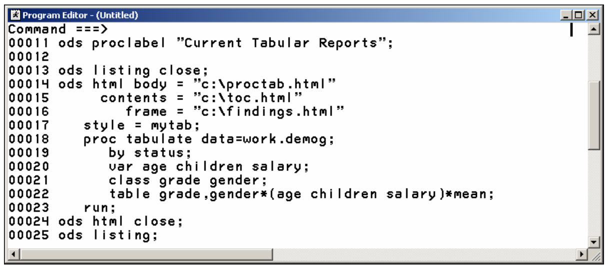


1. The Tabulate Procedure
   ·STATUS=' '
   Cross-tabular summary report
   Table 1
   ·STATUS=D
   Cross-tabular summary report
   Table 1
   ·STATUS=M
   Cross-tabular summary report
   Table 1
   ·STATUS=P
   Cross-tabular summary report
   Table 1

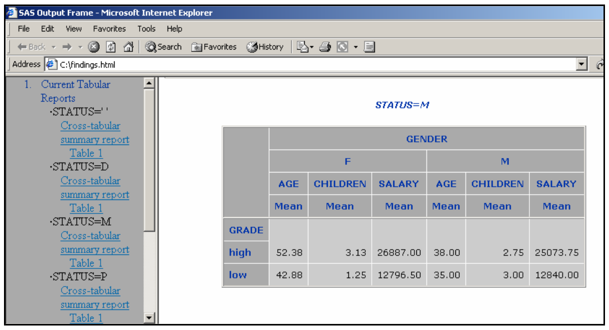
STATUS=M

	GENDER					
	F			M		
	AGE	CHILDREN	SALARY	AGE	CHILDREN	SALARY
	Mean	Mean	Mean	Mean	Mean	Mean
GRADE						
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 or modify the title of the procedure, use the ODS Proclabel statement.



```
Command ===>
ods proclabel "Current Tabular Reports";

ods listing close;
ods html body = "c:\proctab.html"
     contents = "c:\toc.html"
        frame = "c:\findings.html"
   style = mytab;
   proc tabulate data=work.demog;
      by status;
      var age children salary;
      class grade gender;
      table grade,gender*(age children salary)*mean;
   run;
ods html close;
ods listing;
```



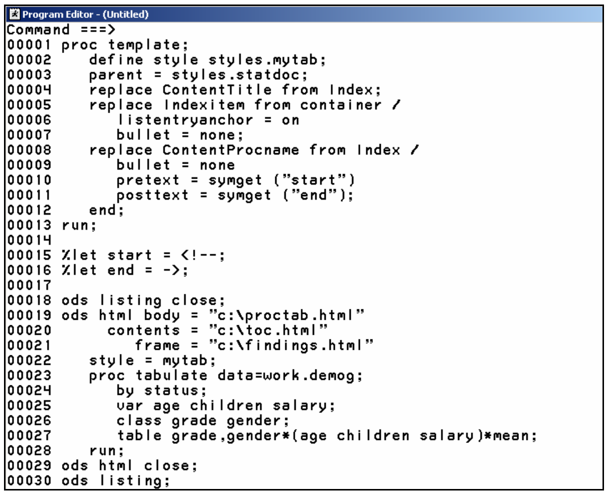
1. Current Tabular Reports
   ·STATUS=' '
   Cross-tabular summary report
   Table 1
   ·STATUS=D
   Cross-tabular summary report
   Table 1
   ·STATUS=M
   Cross-tabular summary report
   Table 1
   ·STATUS=P
   Cross-tabular summary report
   Table 1

STATUS=M

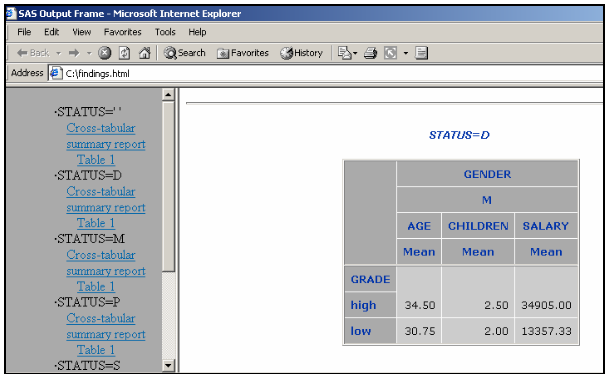
	GENDER					
	F			M		
	AGE	CHILDREN	SALARY	AGE	CHILDREN	SALARY
	Mean	Mean	Mean	Mean	Mean	Mean
GRADE						
high	52.38	3.13	26887.00	38.00	2.75	25073.75
low	42.88	1.25	12796.50	35.00	3.00	12840.00

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 '->'.



```
Command ===>
proc template;
   define style styles.mytab;
   parent = styles.statdoc;
   replace ContentTitle from Index;
   replace Indexitem from container /
      listentryanchor = on
      bullet = none;
   replace ContentProcname from Index /
      bullet = none
      pretext = symget ("start")
      posttext = symget ("end");
   end;
run;

%let start = <!--;
%let end = ->;

ods listing close;
ods html body = "c:\proctab.html"
     contents = "c:\toc.html"
        frame = "c:\findings.html"
   style = mytab;
   proc tabulate data=work.demog;
      by status;
      var age children salary;
      class grade gender;
      table grade,gender*(age children salary)*mean;
   run;
ods html close;
ods listing;
```



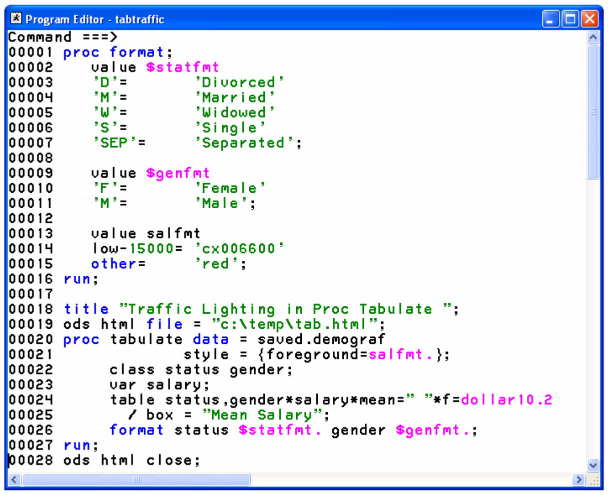
·STATUS=' '
Cross-tabular summary report
Table 1
·STATUS=D
Cross-tabular summary report
Table 1
·STATUS=M
Cross-tabular summary report
Table 1
·STATUS=P
Cross-tabular summary report
Table 1
·STATUS=S

STATUS=D

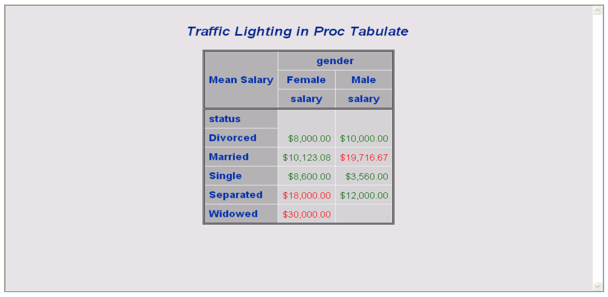
	GENDER		
	M		
	AGE	CHILDREN	SALARY
	Mean	Mean	Mean
GRADE			
high	34.50	2.50	34905.00
low	30.75	2.00	13357.33

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.



```
Command ===>
proc format;
   value $statfmt
   'D'=        'Divorced'
   'M'=        'Married'
   'W'=        'Widowed'
   'S'=        'Single'
   'SEP'=      'Separated';

   value $genfmt
   'F'=        'Female'
   'M'=        'Male';

   value salfmt
   low-15000=  'cx006600'
   other=      'red';
run;

title "Traffic Lighting in Proc Tabulate ";
ods html file = "c:\temp\tab.html";
proc tabulate data = saved.demograf
              style = {foreground=salfmt.};
     class status gender;
     var salary;
     table status,gender*salary*mean=" "*f=dollar10.2
       / box = "Mean Salary";
     format status $statfmt. gender $genfmt.;
run;
ods html close;
```



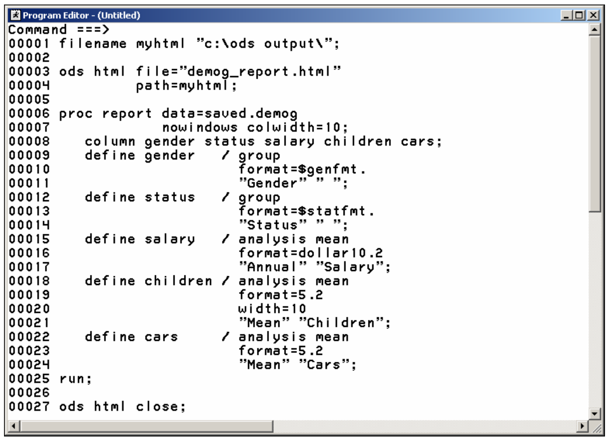
Traffic Lighting in Proc Tabulate

Mean Salary	gender	
	Female	Male
	salary	salary
status		
Divorced	$8,000.00	$10,000.00
Married	$10,123.08	$19,716.67
Single	$8,600.00	$3,560.00
Separated	$18,000.00	$12,000.00
Widowed	$30,000.00	.

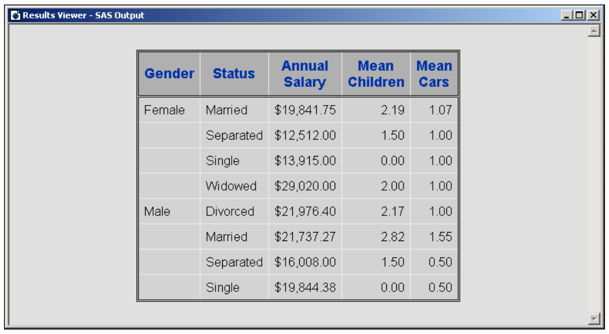
The values of 19716.67, 18000, and 30000 are in red, the rest are in green.

Creating an HTML Output with ODS

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



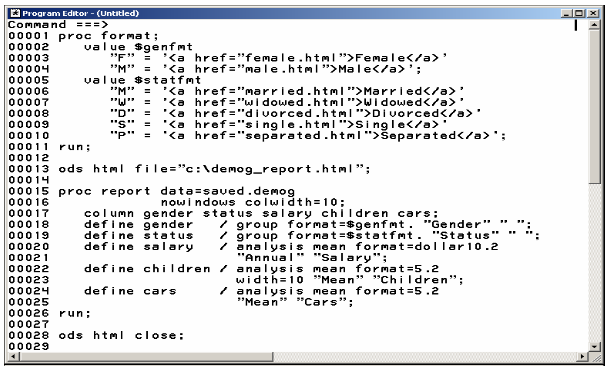
```
Command ===>
filename myhtml "c:\ods output\";

ods html file="demog_report.html"
         path=myhtml;

proc report data=saved.demog
            nowindows colwidth=10;
   column gender status salary children cars;
   define gender   / group
                     format=$genfmt.
                     "Gender" " ";
   define status   / group
                     format=$statfmt.
                     "Status" " ";
   define salary   / analysis mean
                     format=dollar10.2
                     "Annual" "Salary";
   define children / analysis mean
                     format=5.2
                     width=10
                     "Mean" "Children";
   define cars     / analysis mean
                     format=5.2
                     "Mean" "Cars";
run;

ods html close;
```



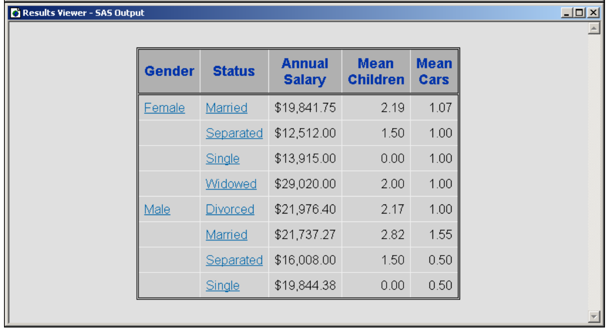
Gender	Status	Annual Salary	Mean Children	Mean Cars
Female	Married	$19,841.75	2.19	1.07
	Separated	$12,512.00	1.50	1.00
	Single	$13,915.00	0.00	1.00
	Widowed	$29,020.00	2.00	1.00
Male	Divorced	$21,976.40	2.17	1.00
	Married	$21,737.27	2.82	1.55
	Separated	$16,008.00	1.50	0.50
	Single	$19,844.38	0.00	0.50

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.



```
Command ===>
proc format;
   value $genfmt
      "F" = '<a href="female.html">Female</a>'
      "M" = '<a href="male.html">Male</a>';
   value $statfmt
      "M" = '<a href="married.html">Married</a>'
      "W" = '<a href="widowed.html">Widowed</a>'
      "D" = '<a href="divorced.html">Divorced</a>'
      "S" = '<a href="single.html">Single</a>'
      "P" = '<a href="separated.html">Separated</a>';
run;

ods html file="c:\demog_report.html";

proc report data=saved.demog
            nowindows colwidth=10;
   column gender status salary children cars;
   define gender   / group format=$genfmt. "Gender" " ";
   define status   / group format=$statfmt. "Status" " ";
   define salary   / analysis mean format=dollar10.2
                     "Annual" "Salary";
   define children / analysis mean format=5.2
                     width=10 "Mean" "Children";
   define cars     / analysis mean format=5.2
                     "Mean" "Cars";
run;

ods html close;
```



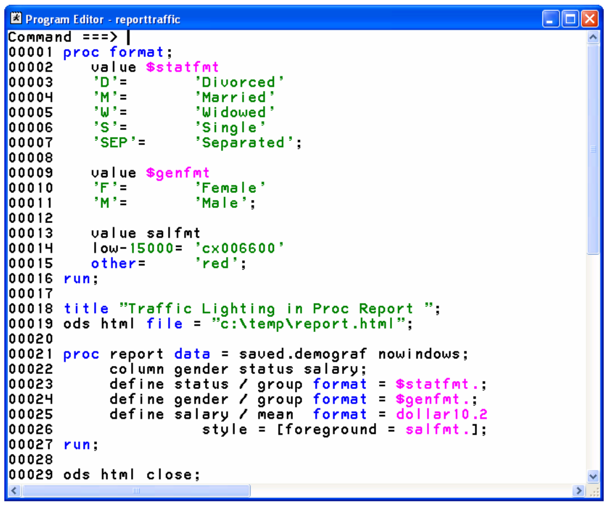
Gender	Status	Annual Salary	Mean Children	Mean Cars
Female	Married	$19,841.75	2.19	1.07
	Separated	$12,512.00	1.50	1.00
	Single	$13,915.00	0.00	1.00
	Widowed	$29,020.00	2.00	1.00
Male	Divorced	$21,976.40	2.17	1.00
	Married	$21,737.27	2.82	1.55
	Separated	$16,008.00	1.50	0.50
	Single	$19,844.38	0.00	0.50

Traffic Lighting in Proc Report

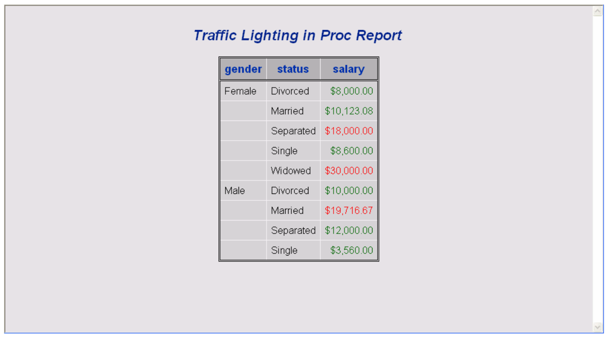
Traffic lighting is a method of presenting data. It uses colors to show relationships.

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.



```
Command ===>
proc format;
   value $statfmt
   'D'=        'Divorced'
   'M'=        'Married'
   'W'=        'Widowed'
   'S'=        'Single'
   'SEP'=      'Separated';

   value $genfmt
   'F'=        'Female'
   'M'=        'Male';

   value salfmt
   low-15000=  'cx006600'
   other=      'red';
run;

title "Traffic Lighting in Proc Report ";
ods html file = "c:\temp\report.html";

proc report data = saved.demograf nowindows;
     column gender status salary;
     define status / group format = $statfmt.;
     define gender / group format = $genfmt.;
     define salary / mean  format = dollar10.2
                style = [foreground = salfmt.];
run;

ods html close;
```



Traffic Lighting in Proc Report

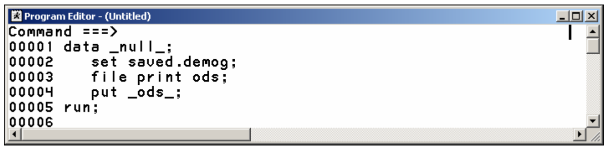
gender	status	salary
Female	Divorced	$8,000.00
	Married	$10,123.08
	Separated	$18,000.00
	Single	$8,600.00
	Widowed	$30,000.00
Male	Divorced	$10,000.00
	Married	$19,716.67
	Separated	$12,000.00
	Single	$3,560.00

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

Introduction to Data _Null_

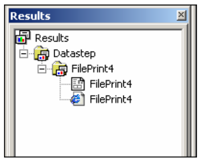
ODS syntax can be used to output data values to the Listing window by using _ODS_ in the Put statement.

A simple example of Data _Null_ output is shown below.

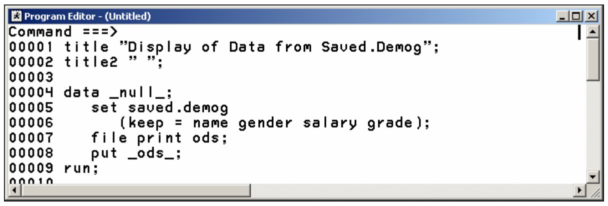


```
Command ===>
data _null_;
   set saved.demog;
   file print ods;
   put _ods_;
run;
```

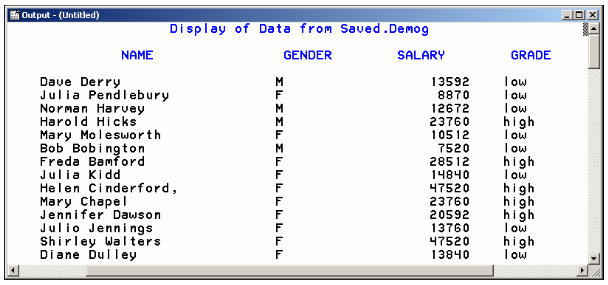
The results of the syntax are shown in the Results window.



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.

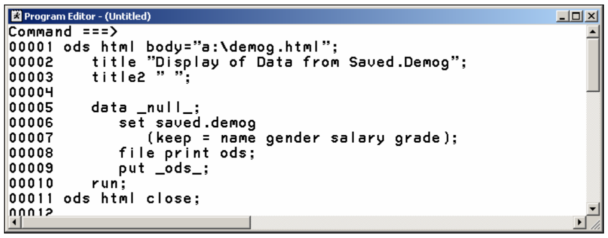


```
Command ===>
title "Display of Data from Saved.Demog";
title2 " ";

data _null_;
   set saved.demog
      (keep = name gender salary grade);
   file print ods;
   put _ods_;
run;
```

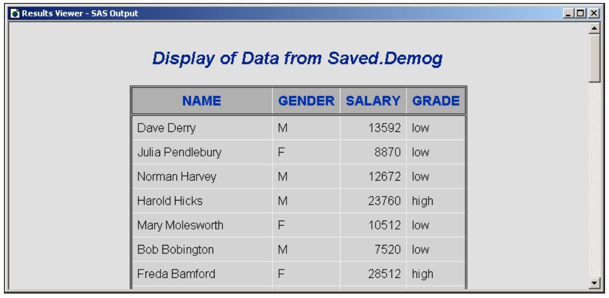


```
                 Display of Data from Saved.Demog

      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 Dulley           F              13840     low
```

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



```
Command ===>
ods html body="a:\demog.html";
   title "Display of Data from Saved.Demog";
   title2 " ";

   data _null_;
      set saved.demog
         (keep = name gender salary grade);
      file print ods;
      put _ods_;
   run;
ods html close;
```



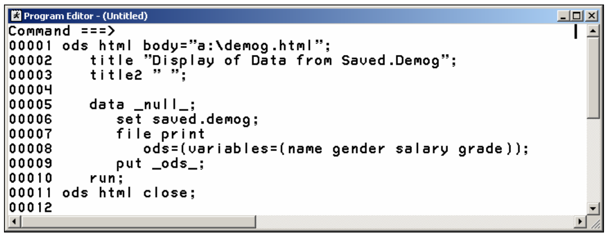
Display of Data from Saved.Demog

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

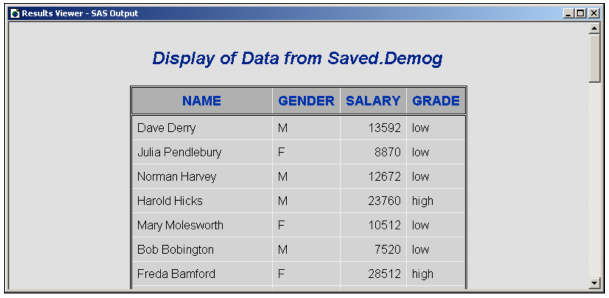
Modifying the File Statement

The File Statement can be used to request various enhancements.

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



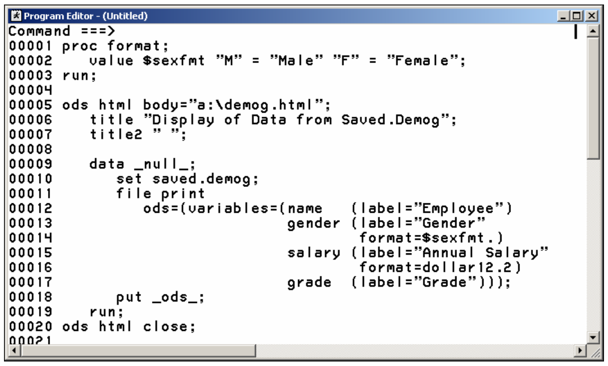
```
Command ===>
ods html body="a:\demog.html";
   title "Display of Data from Saved.Demog";
   title2 " ";

   data _null_;
      set saved.demog;
      file print
         ods=(variables=(name gender salary grade));
      put _ods_;
   run;
ods html close;
```



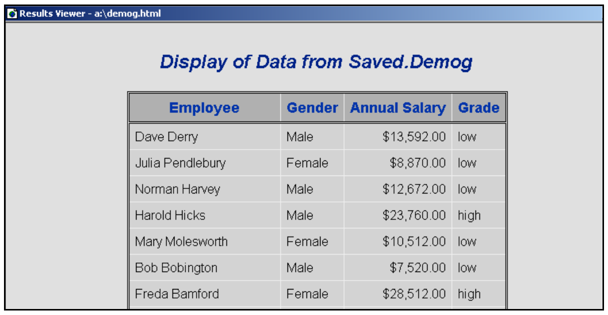
Display of Data from Saved.Demog

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 ===>
proc format;
   value $sexfmt "M" = "Male" "F" = "Female";
run;

ods html body="a:\demog.html";
   title "Display of Data from Saved.Demog";
   title2 " ";

   data _null_;
      set saved.demog;
      file print
         ods=(variables=(name   (label="Employee")
                         gender (label="Gender"
                                 format=$sexfmt.)
                         salary (label="Annual Salary"
                                 format=dollar12.2)
                         grade  (label="Grade")));
      put _ods_;
   run;
ods html close;
```


Results Viewer - a:\demog.html

Display of Data from Saved.Demog

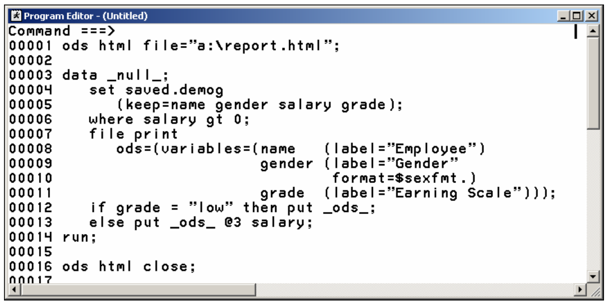
Employee	Gender	Annual Salary	Grade
Dave Derry	Male	$13,592.00	low
Julia Pendlebury	Female	$8,870.00	low
Norman Harvey	Male	$12,672.00	low
Harold Hicks	Male	$23,760.00	high
Mary Molesworth	Female	$10,512.00	low
Bob Bobington	Male	$7,520.00	low
Freda Bamford	Female	$28,512.00	high

Combining Columns of Data

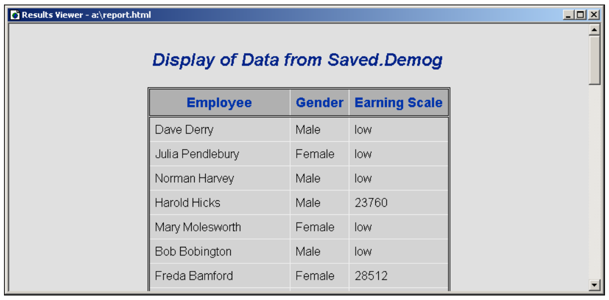
The syntax holds interesting possibilities for combining separate data set columns into the same output column.

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)

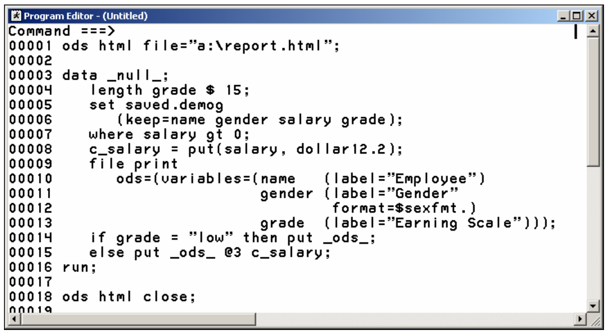
```
Command ===>
ods html file="a:\report.html";

data _null_;
   set saved.demog
      (keep=name gender salary grade);
   where salary gt 0;
   file print
      ods=(variables=(name   (label="Employee")
                      gender (label="Gender"
                              format=$sexfmt.)
                      grade  (label="Earning Scale")));
   if grade = "low" then put _ods_;
   else put _ods_ @3 salary;
run;

ods html close;
```


Results Viewer - a:\report.html

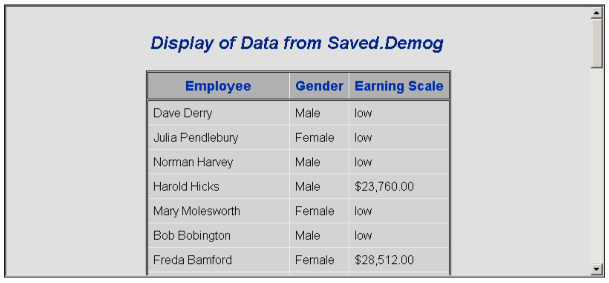
Display of Data from Saved.Demog

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)

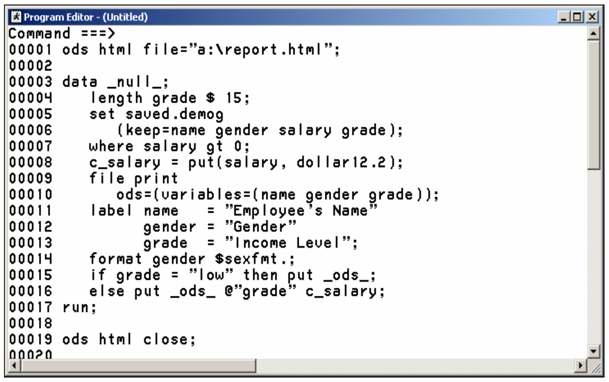
```
Command ===>
ods html file="a:\report.html";

data _null_;
   length grade $ 15;
   set saved.demog
      (keep=name gender salary grade);
   where salary gt 0;
   c_salary = put(salary, dollar12.2);
   file print
      ods=(variables=(name   (label="Employee")
                      gender (label="Gender"
                              format=$sexfmt.)
                      grade  (label="Earning Scale")));
   if grade = "low" then put _ods_;
   else put _ods_ @3 c_salary;
run;

ods html close;
```



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.


Program Editor - (Untitled)

```
Command ===>
ods html file="a:\report.html";

data _null_;
   length grade $ 15;
   set saved.demog
      (keep=name gender salary grade);
   where salary gt 0;
   c_salary = put(salary, dollar12.2);
   file print
      ods=(variables=(name gender grade));
   label name   = "Employee's Name"
         gender = "Gender"
         grade  = "Income Level";
   format gender $sexfmt.;
   if grade = "low" then put _ods_;
   else put _ods_ @"grade" c_salary;
run;

ods html close;
```