Paper GH-03

My Reporting Requires a Full Staff—Help!

Erin Lynch, Daniel O'Connor, Himesh Patel, SAS Institute Inc., Cary, NC

ABSTRACT

With cost cutting and reduced staff, everyone is feeling the pressure to finish their jobs. Preparing reports of any kind can be time-consuming. Not to worry; SAS® is here to the rescue. This presentation goes over a real-world example of generating complex reports for schools. You see how SAS software can create polished and detail-rich academic charts and graphs that can be automatically customized by school. Using the power of ODS and SAS/GRAPH® procedures, you can easily display data in a custom report that has never looked so good.

Examples and code in this presentation help you generalize the usage. These same techniques can be applied to any industry.

INTRODUCTION

Sometimes our programming practices or business processes impede efficient reporting. Our culture demands real-time data—any time; therefore, reporting cannot take weeks or months. Consider the following workflow:

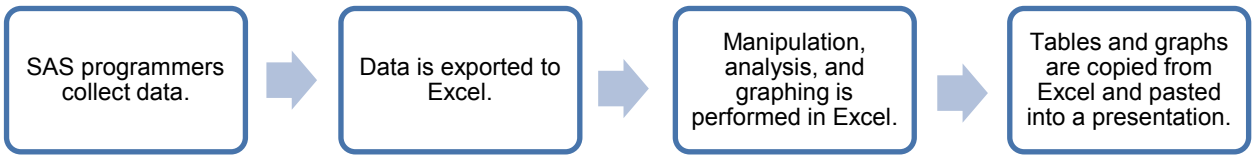


Figure 1. Excel Dependent Reporting

This flow is laborious. It's prone to inconsistencies and errors, and too many resources—a full staff— are wasted preparing and generating reports.

This paper describes a method for delivering consistent presentation-quality output for six school districts. The code will generate six PDF documents, one for each district. By keeping all the reporting effort in SAS, time is saved, resources are reduced, and reports are generated uniformly and fast. Figure 2 illustrates the coding method covered in this paper.

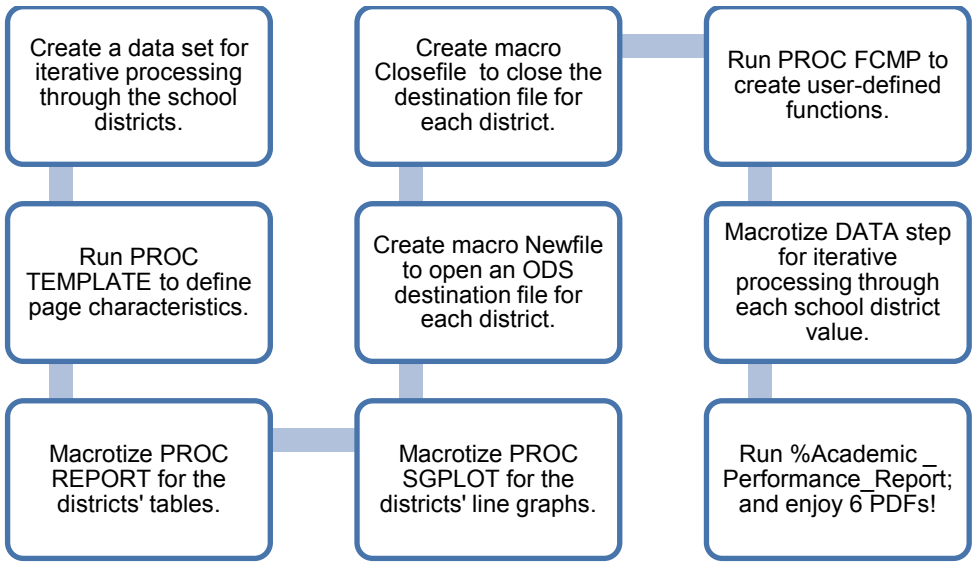


Figure 2. SAS Dependent Reporting

INPUT AND EXPECTED OUTPUT

Two input data sets are used (partial display):

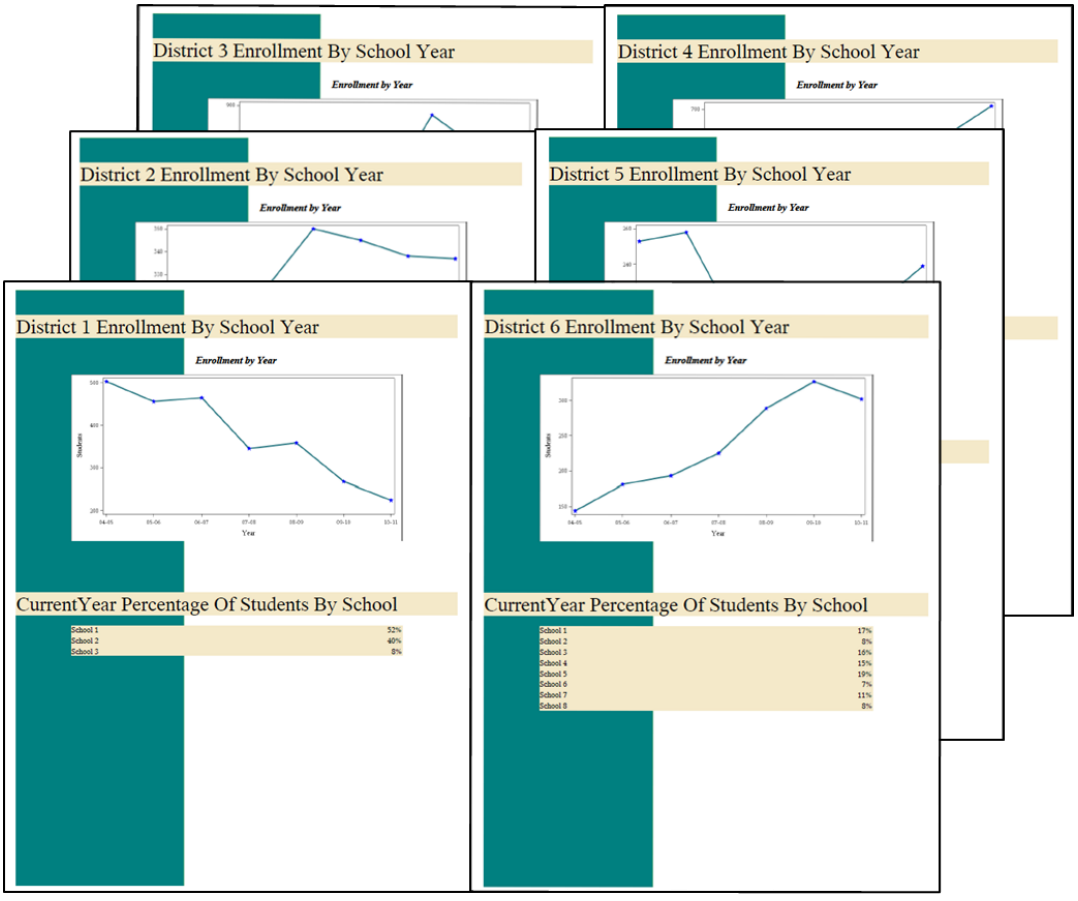
Linedata.sas7bdat

	District	Year	Students
1	District 1	04-05	503
2	District 1	05-06	456
3	District 1	06-07	465
4	District 1	07-08	345
5	District 1	08-09	358
6	District 1	09-10	267
7	District 1	10-11	225
8	District 2	04-05	316
9	District 2	05-06	294
10	District 2	06-07	324
11	District 2	07-08	350
12	District 2	08-09	345
13	District 2	09-10	338
14	District 2	10-11	337
15	District 3	04-05	673
16	District 3	05-06	594

Tabledata.sas7bdat

	District	School	Students	Total_Enrollment	Percent
1	District 1	School 1	117	225	52%
2	District 1	School 2	91	225	40%
3	District 1	School 3	17	225	8%
4	District 2	School 1	122	337	36%
5	District 2	School 2	84	337	25%
6	District 2	School 3	61	337	18%
7	District 2	School 4	32	337	9%
8	District 2	School 5	38	337	11%
9	District 3	School 1	150	518	29%
10	District 3	School 2	132	518	25%
11	District 3	School 3	95	518	18%
12	District 3	School 4	97	518	19%
13	District 3	School 5	25	518	5%
14	District 3	School 6	19	518	4%
15	District 4	School 1	146	704	21%
16	District 4	School 2	88	704	13%

The expected result is:



The SAS Output Delivery System can efficiently output a separate PDF for each district's enrollment line graph, and a separate PDF member for each district's data table. The challenge, however, is to run code that creates a presentation-ready PDF containing a combination of graphical and text data.

CODE HIGHLIGHTS

Each step through the code (Figure 2) encloses parts of the code in macro for PROC FCMP. Refer to the Appendix for the full sample code.

PROC FCMP will create user-defined functions Newfile, Enrollment, District_Makeup, and Closefile that will be called at the end of the code by macro %Academic_Performance_Report.

Macro %Academic_Performance_Report will deliver a PDF for each school district ready for presentation and delivery. Let's step through the code.

DATA SET FOR ITERATIVE PROCESSING

The data set districts.sas7bdat is created from the input data set linedata.sas7bdat and contains the distinct values for school district. This data set is used throughout the code to define iterative values for the macro variable &district. Since this data set drives the report generation, adding a school district will automatically generate a PDF. This method increases efficiency by eliminating a lot of potential code maintenance.

PROC TEMPLATE

PROC TEMPLATE is applied to set up the appearance of the page. Create or use your own image file when running the code. Within PROC TEMPLATE, adjust backgroundimage="Your.png" with the physical location of your image file. The code creates an ODS Style called Charter, which is saved in your SASUSER.TEMPLAT and is permanent until you delete it.

```
proc template;
 define style Styles.Charter;
 parent = styles.printer;
 style Body from Document
   "Undef margins so we get the margins from the printer or SYS option"
    /
    marginbottom = _undef_
    margintop = _undef_
    marginright = _undef_
    marginleft = _undef_
    pagebreakhtml = html('PageBreakLine')
   backgroundimage="Your.png";
end;
run;
```

MACRO DISTRICT_MAKEUP AND IN-LINE FORMATTING

The macro District_Makeup is created around PROC REPORT for presenting the table. Just before the PROC TEMPLATE, there is one line of code:

```
ods escapechar="~";
```

This tells ODS, "When you encounter this character, in-line formatting is beginning." In-line formatting controls the look and content of titles above the table as well as orientation on the page.

```
%macro District_Makeup;

%let district=%sysfunc(dequote(&district.));

   ods text="~{newline 6}";

   ods text="~{style [width=100pct font_size=26pt background=CXf4e9c9]
   Current Year Percentage Of Students By School}";
```

```
    proc report data=tabledata(where=(district="&district")) nowd

    style(report)={frame=void font_size=12pt rules=none backgroundcolor=CXF4E9C9
    cellpadding=0 cellspacing=0};
       define district / noprint;
       define students / noprint;
       define total_enrollment / noprint;
       define school / '' style(column)={width=5.5in};
       define percent / '' style(column)={width=.5in} right;
    run;
%mend;
```

MACRO ENROLLMENT AND PROC SGPLOT

The macro Enrollment is created around PROC SGPLOT. If you have not explored PROC SGPLOT, do so! Again, notice the in-line formatting before the PROC.

```
%macro Enrollment;
%let district=%sysfunc(dequote(&district.));

  ods text="~{newline 3}";
  ods text="~{style [width=100pct font_size=26pt background=CXf4e9c9]
  &district Enrollment By School Year}";
  ods text="~{newline 2}";
  ods text="~{style systemtitle [just=center]Enrollment by Year}";
  ods graphics / height=3in width=6in;
  proc sgplot data=sorted_linedata(where=(district="&district"));
     series x=year y=students / markers
     lineattrs=(color=CX39828C pattern=SOLID thickness=3)
     markerattrs=(color=CX0000FF symbol=STARFILLED) name='series';
  run;
%mend;
```

MACROS NEWFILE AND CLOSEFILE

The macro Newfile is created to instruct ODS to open a declared (&destination) destination file, generate output with standard naming (&file), use a specific style (&style), use specified path (&path) and gpath (&gpath) if destination is HTML, and extension (&extension) if destination is RTF. The parameters passed into the macro Academic_Performance_Report will be used by Newfile to generate a PDF, HTML, or RTF for each value in the districts.sas7bdat data set.

The macro Closefile closes the declared destination file after each iterative run through the values in the districts.sas7bdat data set.

These two macros are great reusable code for your personal collection.

PROC FCMP

This is where everything comes together. PROC FCMP is the SAS Function Compiler procedure; it enables you to create and store SAS functions before they are used in the Academic_Performance_Reporting macro. PROC FCMP uses the macros Enrollment, District_Makeup, Newfile, and Closefile (including the parameters each relies on) to create user-defined functions. These functions will be called in the macro Academic_Performance_Report.

```
proc fcmp outlib=work.fncs.submit;
function Enrollment(district $);
 rc = run_macro('Enrollment', district );
 return(rc);
 endsub;
function District_Makeup(district $);
 rc = run_macro('District_Makeup', district );
 return(rc);
 endsub;
function Newfile( destination $, path $, gpath $, file $, extension $,
styleparm $ );
 rc = run_macro('Newfile', destination, path, gpath, file, extension,
 styleparm );
```

```
 return(rc);
 endsub;
function Closefile( destination $ );
 rc = run_macro('CloseFile', destination );
 return(rc);
 endsub;
run;
quit;
```

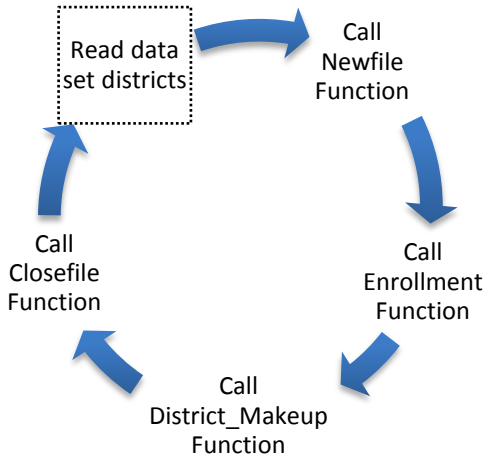
MACRO ACADEMIC_PERFORMANCE_REPORT

Now, let's generate some PDF documents—one for each district! Here is the macro Academic_Performance_Report:

```
%macro Academic_Performance_Report (destination=pdf, path=, gpath=, extension=,
   style=Charter );
   %if ( "&extension" eq "" ) and ( &destination ne "" ) %then
   %let extension =&destination;
   options cmplib=work.fncs;
   data _null_;
   set districts;
    rc = Newfile( symget('destination'), symget('path'), symget('gpath'),
   cats(district, "_Annual_Performance"), symget('extension'), symget('style'));
   if ( rc eq 0 ) then do;
    rc = Enrollment( district );
    rc = District_Makeup( district );
    rc = Closefile(symget('destination'));
   end;
   ;run;
   %mend;
```

The macro Academic_Performance_Report declares parameter values destination=pdf and style=Charter that are used by the macros Newfile and Closefile. The data set districts.sas7bdat is used to iterate through the code from the first value of 'District 1' through last value of 'District 6.'

This is what happens:



The macro results in six PDFs named District_1_Annual_Performance.pdf through District_6_Annual_Performance.pdf.

CONCLUSION

Wouldn't it be grand to run code any day of a fiscal year, quarter, month, or even week and have presentation-ready performance reports generated in a snap? It is possible using Base SAS procedures, ODS, and SAS/GRAPH.

On the surface, this coding method may appear to be a lot more work than simply exporting to Excel and having "the full staff" manipulate, analyze, and graph data, school district by school district.

We have stepped through a coding method that will save time, reduce resources, and ensure consistent reporting. If the input data, macros, and user-defined functions were stored in or run to permanent locations, you could simply run:

```
%Academic_Performance_Report;
```

Maintenance would be making certain the input data was refreshed, and rerunning PROC FCMP should there be changes in the macros. So, instead of this being a conclusion, we consider it a beginning!

Please try out the full sample code, complete with comprehensive comments, located in the Appendix.

ACKNOWLEDGMENTS

Thanks to Sanjay Matange, Timothy McBride, and Robin Angley for their input and reviews.

RECOMMENDED READING

- Haworth, Lauren, Cynthia L. Zender, and Michele Burlew. 2009. *Output Delivery System: The Basics and Beyond*. Cary, NC: SAS Press.
- SAS Institute Inc. 2009. *SAS® 9.2 Output Delivery System: User's Guide*. Cary, NC: SAS Institute Inc.
- SAS Institute Inc. 2010. *SAS/GRAPH® Statistical Graphics Procedures Guide*, 2nd ed. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Erin Lynch
Building U, SAS Campus Drive
SAS Institute Inc.
Cary, NC 27513
E-mail: Erin.Lynch@sas.com

Daniel O'Connor
Building R, SAS Campus Drive
SAS Institute Inc.
Cary, NC 27513
E-mail: Dan.OConnor@sas.com

APPENDIX

Sample Code

```
/* Create datasets linedata and tabledata.                                    */
data linedata;
input District $12. Year $ Students;
cards;
District1     04-05  503
District1     05-06  456
District1     06-07  465
District1     07-08  345
District1     08-09  358
District1     09-10  267
District1     10-11  225
District2     04-05  316
District2     05-06  294
District2     06-07  324
District2     07-08  350
District2     08-09  345
District2     09-10  338
District2     10-11  337
District3     04-05  673
District3     05-06  594
District3     06-07  810
District3     07-08  628
District3     08-09  872
District3     09-10  767
District3     10-11  518
District4     04-05  545
District4     05-06  596
District4     06-07  638
District4     07-08  648
District4     08-09  665
District4     09-10  666
District4     10-11  704
District5     04-05  253
District5     05-06  258
District5     06-07  209
District5     07-08  212
District5     08-09  188
District5     09-10  217
District5     10-11  239
District6     04-05  144
District6     05-06  181
District6     06-07  193
District6     07-08  225
District6     08-09  289
District6     09-10  327
District6     10-11  302
;
run;
data linedata;
set linedata;
district=substr(district,1,8)||' '||substr(district,9,1);
run;

data tabledata;
input District $12. School $8. Students Total_Enrollment Percent $;
cards;
District1     School1       117     225     52%
District1     School2       91      225     40%
```

```
District1     School3       17     225    8%
District2     School1       122    337    36%
District2     School2       84     337    25%
District2     School3       61     337    18%
District2     School4       32     337    9%
District2     School5       38     337    11%
District3     School1       150    518    29%
District3     School2       132    518    25%
District3     School3       95     518    18%
District3     School4       97     518    19%
District3     School5       25     518    5%
District3     School6       19     518    4%
District4     School1       146    704    21%
District4     School2       88     704    13%
District4     School3       73     704    10%
District4     School4       59     704    8%
District4     School5       33     704    5%
District4     School6       111    704    16%
District4     School7       98     704    14%
District4     School8       56     704    8%
District4     School9       40     704    6%
District5     School1       87     239    36%
District5     School2       46     239    19%
District5     School3       60     239    25%
District5     School4       46     239    19%
District6     School1       50     302    17%
District6     School2       24     302    8%
District6     School3       47     302    16%
District6     School4       45     302    15%
District6     School5       57     302    19%
District6     School6       22     302    7%
District6     School7       32     302    11%
District6     School8       25     302    8%
;
run;
data tabledata;
set tabledata;
district=substr(district,1,8)||' '||substr(district,9,1);
school=substr(school,1,6)||' '||substr(school,7,1);
run;

/* Begin Sample Code.                                                    */

ods _all_ close;

proc sort data=linedata out=sorted_linedata;
by district year;
run;

/* Create a data set of all the unique school districts that will be used   */
/* to iterate through the generation of a report for each district.         */

proc sort data=linedata out=districts(keep=district) nodupkey;
by district;
run;

/* Set the margins for the page.                                         */
options topmargin=.125in bottommargin=.125in leftmargin=.25in rightmargin=.25in;

/* Clear the system title, and turn date and page numbering off.         */
title;
options nodate nonumber;
```

```
/* The ODS ESCAPE Character is a special character that is embedded within  */
/* the text, and provides ODS a way of looking to apply additional          */
/* characteristics to the text.                                             */

ods escapechar="~";

/* Create a style that puts a background image on each page of the report.  */
proc template;
  define style Styles.Charter;
  parent = styles.printer;
  style Body from Document
    "Undef margins so we get the margins from the printer or SYS option"
     /
     marginbottom = _undef_
     margintop = _undef_
     marginright = _undef_
     marginleft = _undef_
     pagebreakhtml = html('PageBreakLine')
     backgroundimage="Your.png";
  end;
run;

/* Create a table for each school district.                                 */
%macro District_Makeup;

/* The input macro argument gets quoted coming through the FCMP interface   */
/* and we need to strip the quotes off.  Use the dequote system function    */
/* to get our internal district macro variable to look right.               */

%let district=%sysfunc(dequote(&district.));

   /* Use ODS in-line formatting syntax to insert 6 blank lines. The escape */
   /* character followed by a { indicates special instructions are coming.  */
   /* The next word (newline) is considered a function which takes a        */
   /* numeric argument being the number of blank lines you want to insert.  */

       ods text="~{newline 6}";

   /* Use ODS in-line syntax to apply stylistic attributes to the text.     */
   /* In this example we are giving the field a width of 100% of the page,  */
   /* telling ODS to use a 26pt font, and a background color. The key thing */
   /* here to notice is that the text is the argument to the function.      */
   /* The attributes will be applied to the text until you reach the }.     */

ods text="~{style [width=100pct font_size=26pt background=CXf4e9c9]Current
Year Percentage Of Students By School}";

   proc report data=tabledata(where=(district="&district")) nowd

/* In-line formatting can be used within PROC Report to control the look of */
/* the entire report: turn the outer boarder of the table off frame=void,   */
/* turn the inner cell boarders off rules=none, use a 12pt font, a          */
/* background color, and turn the padding inside the cell off as well as    */
/* spacing between cells off.                                               */

style(report)={frame=void font_size=12pt rules=none backgroundcolor=CXF4E9C9
cellpadding=0 cellspacing=0};
        define district / noprint;
        define students / noprint;
        define total_enrollment / noprint;
        define school / '' style(column)={width=5.5in};
        define percent / '' style(column)={width=.5in} right;
   run;
```

```
%mend;

/* Create a graph for each school district.                              */
%macro Enrollment;
%let district=%sysfunc(dequote(&district.));

   ods text="~{newline 3}";
   ods text="~{style [width=100pct font_size=26pt background=CXf4e9c9]&district
Enrollment By School Year}";
   ods text="~{newline 2}";
   ods text="~{style systemtitle [just=center]Enrollment by Year}";
   ods graphics / height=3in width=6in;
   proc sgplot data=sorted_linedata(where=(district="&district"));
        series x=year y=students / markers
        lineattrs=(color=CX39828C pattern=SOLID thickness=3)
        markerattrs=(color=CX0000FF symbol=STARFILLED) name='series';
   run;
%mend;

/* Turn ON the requested output destination (PDF, HTML, RTF, or TAGSETS.RTF)*/
/* with the appropriate options.                                         */

%macro Newfile;

/* The PATH and GPATH macro variables are only used for the HTML         */
/* destination to indicate where to put the HTML and graph files.  The   */
/* (url=none) says do not include the physical pathname in the HTML      */
/* output because we would not be able to move it to our web server.     */

       %if &path ne '' %then
        %let pathopt= path=&path(url=none);
   %else
        %let pathopt=;

   %if &gpath ne '' %then
        %let gpathopt= gpath=&gpath(url=none);
   %else
        %let gpathopt=;
   %let path=%sysfunc(dequote(&path.));
   %let gpath=%sysfunc(dequote(&gpath.));

   /* The DESTINATION macro variable controls with ODS Destination we are */
   /* starting (PDF, HTML, RTF, or TAGSETS.RTF).                          */
   %let destination=%sysfunc(dequote(&destination.));

   /* The FILE macro variable is the filename that we are going to create. */
   %let file=%sysfunc(translate(%sysfunc(dequote(&file.)), "_", " "));

   /* The EXTENSION macro variable is the extension on the file that we    */
   /* are going to create.                                                 */
   %let extension=%sysfunc(dequote(&extension));

   /* The STYLEPARM macro variable is the name of the style that we want   */
   /* to use for our report.  In the default case we are using the         */
   /* STYLES.CHARTER style that we created at the beginning of the program */
   /* that will put a background image on every page.                      */
   %if &styleparm ne '' %then
      %let styleopt= style=%sysfunc(dequote(&styleparm.));
   %else
      %let styleopt=;

   /* Put it all together to create an ODS statement, and turn on the      */
   /* ODS destination.                                                     */
```

```
   %if ( %upcase(&destination) eq PDF ) %then %do;
       ods &destination file="&path.&file..&extension" notoc startpage=no
       &styleopt;
   %end;
   %else %if (( %upcase(&destination) eq RTF ) or ( %upcase(&destination) eq
   TAGSETS.RTF )) %then %do;
       ods &destination file="&path.&file..&extension" startpage=no &styleopt;
   %end;
   %else %if ( %upcase(&destination) eq HTML ) %then %do;
       ods &destination file="&file..&extension" &pathopt &gpathopt &styleopt;
   %end;

%mend;

/* Turn OFF the requested output destination (PDF, HTML, RTF, or            */
/* TAGSETS.RTF).                                                            */
%macro Closefile;
   %let destination=%sysfunc(dequote(&destination.));

   ods &destination close;
%mend;


/* PROC FCMP is the TRICK PONY that makes all this possible. FCMP stands    */
/* function compiler.  Basically all we are doing here is creating a user   */
/* defined data step function that knows how to call a previously defined   */
/* macro function. It creates the definition of the function and the        */
/* arguments that are going to be passed in from the execution of the data  */
/* step.                                                                    */

proc fcmp outlib=work.fncs.submit;
  function Enrollment(district $);
    rc = run_macro('Enrollment', district );
    return(rc);
  endsub;
  function District_Makeup(district $);
    rc = run_macro('District_Makeup', district );
    return(rc);
  endsub;
  function Newfile( destination $, path $, gpath $, file $, extension $,
  styleparm $ );
    rc = run_macro('Newfile', destination, path, gpath, file, extension,
    styleparm );
    return(rc);
  endsub;
  function Closefile( destination $ );
    rc = run_macro('CloseFile', destination );
    return(rc);
  endsub;
run;
quit;


/* Iterate through our input data set of unique school districts and        */
/* produce a report for each.                                               */
/* Set some default values for our input macro variables:                   */
/* DESTINATION = PDF                                                        */
/* STYLE = Charter                                                          */

%macro Academic_Performance_Report (destination=pdf, path=, gpath=, extension=,
style=Charter );

/* If the EXTENSION macro variable has not been defined then set it to the  */
```

```
/* output destination name. In most cases they are the same.  The only      */
/* difference is the TAGSETS.RTF destination, in which case you would       */
/* want to set the EXTENSION macro variable to RTF.                         */
%if ( "&extension" eq "" ) and ( &destination ne "" ) %then
   %let extension =&destination;

/* Tell the SAS System where to find the compiled user defined functions.   */
options cmplib=work.fncs;

/* Use our unique school district data set to drive the report              */
/* generation for each school district.                                     */
data _null_;
set districts;

 /* Call our user defined data step function that will execute the NEWFILE  */
 /* MACRO, and will start an ODS Destination with a data dependent          */
 /* file name.                                                              */
 /* (i.e. District1_Annual_Performance.pdf,                                 */
 /* District<n>_Annual_Performance.pdf), so on.                             */
 rc = Newfile( symget('destination'), symget('path'), symget('gpath'),
      cats(district, "_Annual_Performance"), symget('extension'),
      symget('style'));

 /* If the NEWFILE function executed without an error then execute the      */
 /* additional user defined functions.                                      */
 if ( rc eq 0 ) then do;

    /* Call the user defined function to produce a GRAPH for the current    */
    /* observation(district).                                               */
    rc = Enrollment( district );

    /* Call our user defined function to produce a TABLE for the current    */
    /* observation(district).                                               */
    rc = District_Makeup( district );

    /* Call our user defined function to close the ODS DESTINATION.         */
    rc = Closefile(symget('destination'));
 end;

;run;
%mend;

%Academic_Performance_Report;

/* If you wanted to create different output types all you would have to do  */
/* is the following:                                                        */
/* %Academic_Performance_Report(destination=HTML, path=<physical path>,     */
/* gpath=<physical path for graphs>);                                       */
/* %Academic_Performance_Report(destination=RTF);                           */
/* %Academic_Performance_Report(destination=TAGSETS.RTF, extension=RTF);    */
```