Paper 095-30

Web Reporting Using the ODS

By Jeffery D. Gilbert, Steelcase, Inc., Grand Rapids, MI

ABSTRACT

This paper will explore HTML, and some introductory HTML coding, and then dig into the use of the Output Delivery System (ODS). In doing this, we will dive into create drillable reports. Of particular interest, we will dig into using labels and footnotes to create links, leading to a dynamic, drillable reporting structure. This paper assumes no previous knowledge of HTML reporting or ODS, but will build quickly to dynamic HTML reports.

INTRODUCTION to HTML

HyperText Markup Language (HTML) is simply a text formatting language. It is understood by most internet browsers, including Internet Explorer, Netscape, and Mozilla.

HTML is a language made up of formatting tags. These tags are markers that indicate the beginning and ending spots for each format. The code is always enclosed in arrows (the less than and greater than symbols). The ending format uses the same code, except it is preceded by a forward slash. For example, if you want to bold a line of output, the HTML code will look like this:

```
<b>This line is going to be in bold format</b>
```

Internet browsers will interpret this to format the line as:

This line is going to be in bold format

In addition, some formats have options, which would be specified after the beginning format. For example, the tag indicating that the preceding text is part of the body of the document is <BODY>. To change the background color, the option BGCOLOR can be used. For example:

```
<BODY BGCOLOR=TAN>
```

will change the background color of the document's body to tan.

Another useful HTML tag with an important option is the <a> tag, which controls links. Use the option HREF set to point to the SAS web site, as such:

```
<a href=www.sas.com>Link to www.sas.com</a>
```

This produces a link in your web page pointing to www.sas.com, with the text being "Link to www.sas.com".

As an example, you can easily create your own basic HTML file. Simply try these steps:

1. Opening Notepad
2. Entering the following text:

```
<head> This is the page header</head>
<body bgcolor=tan>
<h1> This will create a large header line </h1>
<h3> This is the default header line </h3>
<h6> This is the smallest header line </h6>
<b> tag creates text in bold type </b>
<p> Indicates a new paragraph
<i> tag creates text in italics </i>
<p>
<font size=7 color=RED face=arial> Creates text in the <u>largest</u> size available, in red, in
Arial font </font>
</body>
<a href=www.sas.com>Link to the SAS Web Site</a>
```

3. Save the file with an extension of "htm".
4. Find the file on your computer, and double-click on it. Your browser will launch, and you will see results similar to the figure 1. This example shows some common, useful tags and their results:

This is the page header

This will create a large header line

This is the default header line

This is the smallest header line

tag creates text in bold type
Indicates a new paragraph *tag creates text in italics*

Creates text in the largest size available, in red, in Arial font

Link to the SAS web site

If you wanted to change something, simply make the change using Notepad, save it, and refresh your browser.

Now that you've created an HTML file, albeit a fairly useless file, you can see how valuable this could be for formatting output from SAS.

USING HTML IN SAS CODE

Using HTML code within SAS titles, footnotes, and labels will produce results on your output file, viewed from your brow using that code to generate the formatting. For example, when using the ODS (explained later) to produce HTML, and using a title statement like the following title:

```
title "<b>This is the title</b><I> of this report</I>"
```

Will produce a web page with a title that looks like this:

This is the title *of this report*

Likewise, if you wanted your footnote to include a link to the SAS web site, as discussed above, the SAS code below will take care of it all.

```
footnote "<a href=www.sas.com>Link to www.sas.com<a>"
```

That's as tricky as the coding gets. The next logical step, therefore, is to integrate the use of HTML into the SAS Output Delivery System using ODS HTML. But before we combine our HTML formatting capabilities with the ODS, we will step back and give a brief introduction to using the ODS HTML statement.

USING ODS HTML

The ODS statement brought BASE SAS from into the 21st Century by allowing programmers to easily create HTML, Excel, and other types of output that is easily shareable. Of interest to this paper, of course, is the ODS HTML statement, to create HTML output from procedures.

When using the ODS HTML statement, the basic call contains two parts: the ODS call statement, and the ODS close statement.

The ODS call defines the type of ODS statement called, options and file names. In its most basic form, contains simply a BODY statement, or alternatively a FILE statement, which works the same the BODY statement in the ODS HTML statement. For example:

```
ODS HTML body='c:\temp.htm';
```

The ODS Close statement essentially tells SAS to close the files created by the ODS call, and that these files are complete. This statement looks like:

```
ODS HTML CLOSE;
```

So a basic PROC PRINT like the one below, on the table SASUSER.ORANGES, will produce output like that displayed in Figure 1.

```
ods html body='c:\orange1.htm';
proc print data=sasuser.oranges noobs
label;
  var store day price1 price2 sales1
sales2;
  where store='1';
run;
ods html close;
```

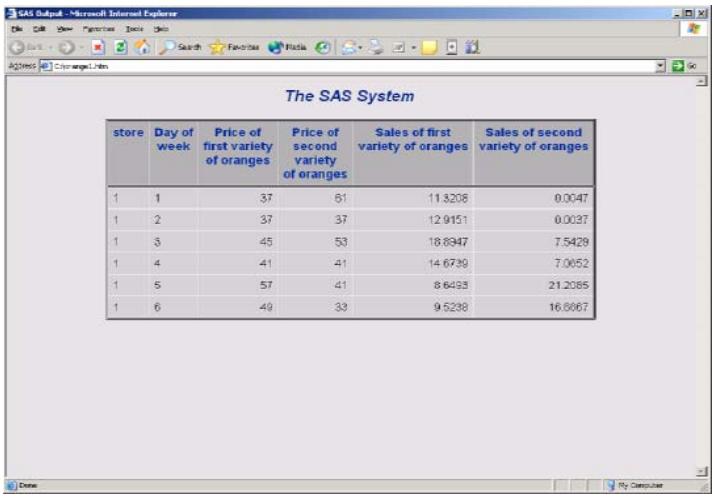


Figure 1 - A basic ODS HTML call creates a file named orange1.htm

The ODS statement can also produce an index page and a contents page. Changing the first line to include a frame statement and a contents statement, as below, produces the results in Figure 2.

```
ods html body='c:\orange1.htm'
         frame='c:\orange_frame.htm
'
         contents='c:\orange_conten
ts.htm';
```

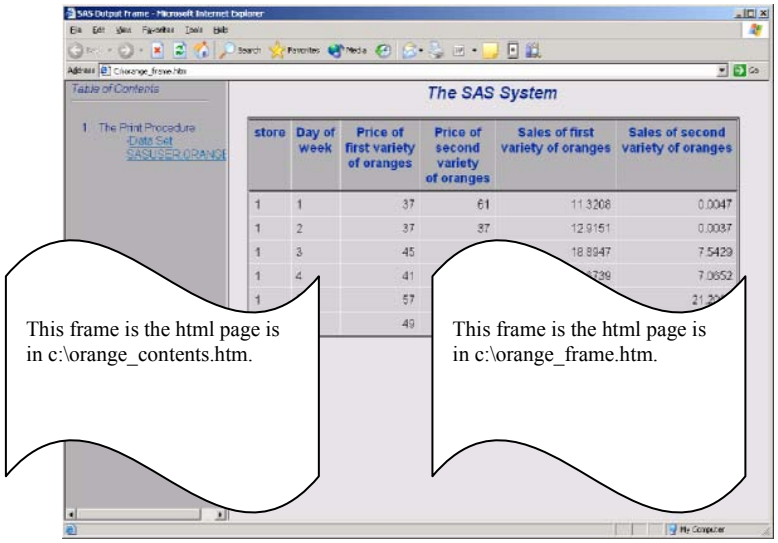
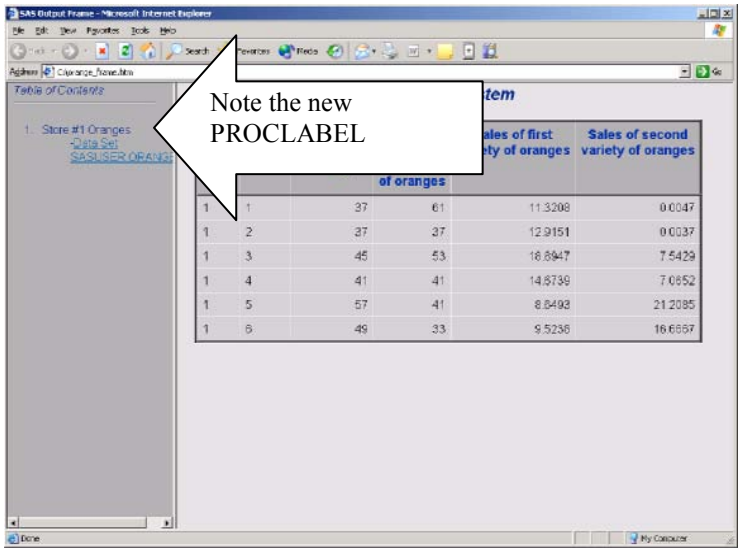


Figure 2 - ODS Body is orange1.htm, with both a frame and contents page.

To change the procedure label from "The Print Procedure" to something meaningful, you will want to use the following statement. Note that this statement ONLY applies to the next procedure, and will be reset to the default after that next procedure is finished running.

```
ods proclabel 'Store #1 Orange';
```



We could continue with many tips and tricks, and we did not even touch on the subject of styles, which could be an entire paper in itself. I would encourage you to obtain a copy of the references, listed below, as they contain many more great capabilities than covered in this paper. The intent of this paper was to address is the ability to put together the ODS HTML statement, with HTML programming, to create drillable reports. Therefore, we will now turn our attention to the ODS with HTML code.

COMBINING HTML PROGRAMMING WITH THE ODS

A most exciting development of the new generation of The SAS System, and specifically BASE SAS, is the ability to combine HTML programming with the ODS to format reports, and manipulate an HTML file. The below example shows two basic ODS HTML calls, with the addition of a footnote statement in each. The footnotes contain links to the other file created, so that any user can flip between the two reports. The results are in Figure 3 and Figure 4.

```
ods html body='c:\orange_store1.htm';
proc print data=sasuser.oranges noobs
label;
  var store day price1 price2 sales1
sales2;
  where store='1';
  footnote "<a
href='c:\orange_store2.htm'>View Store
#2 Sales<a>";
run;
ods html close;

ods html body='c:\orange_store2.htm';
proc print data=sasuser.oranges noobs
label;
  var store day price1 price2 sales1
sales2;
  where store='2';
  footnote "<a
href='c:\orange_store1.htm'>View Store #1
Sales<a>";
run;
ods html close;
```

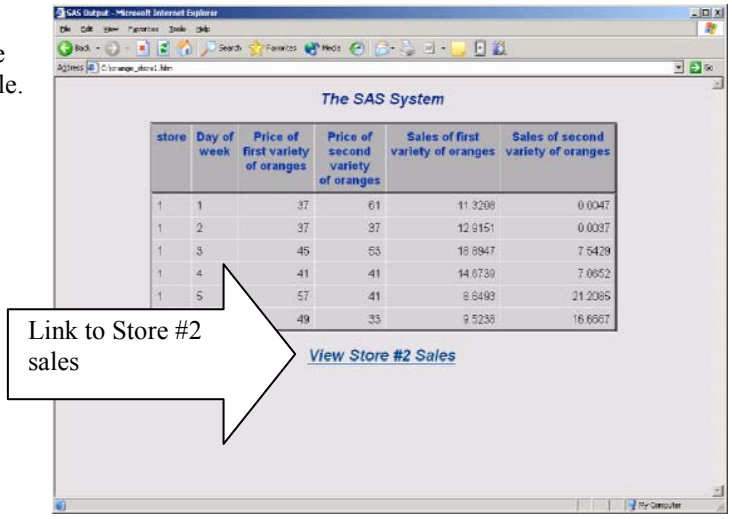


Figure 3 - Report for "Store 1", with a link to the report for "Store 2".

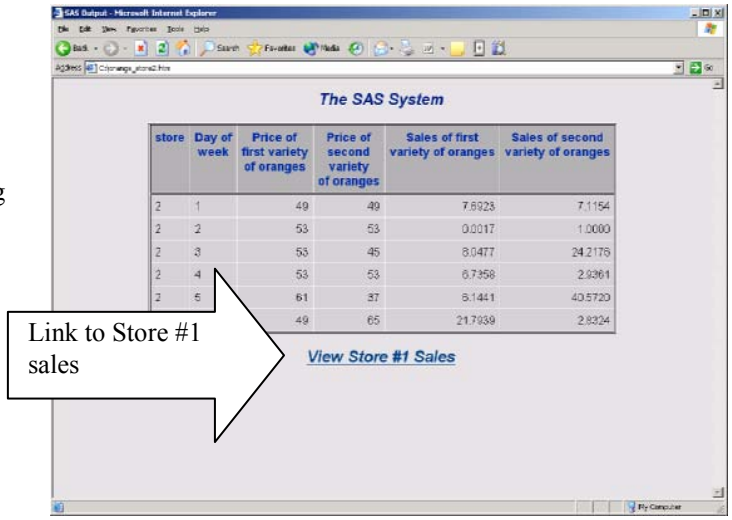


Figure 4 - Report for "Store 2", with a link back to the report for "Store 1"

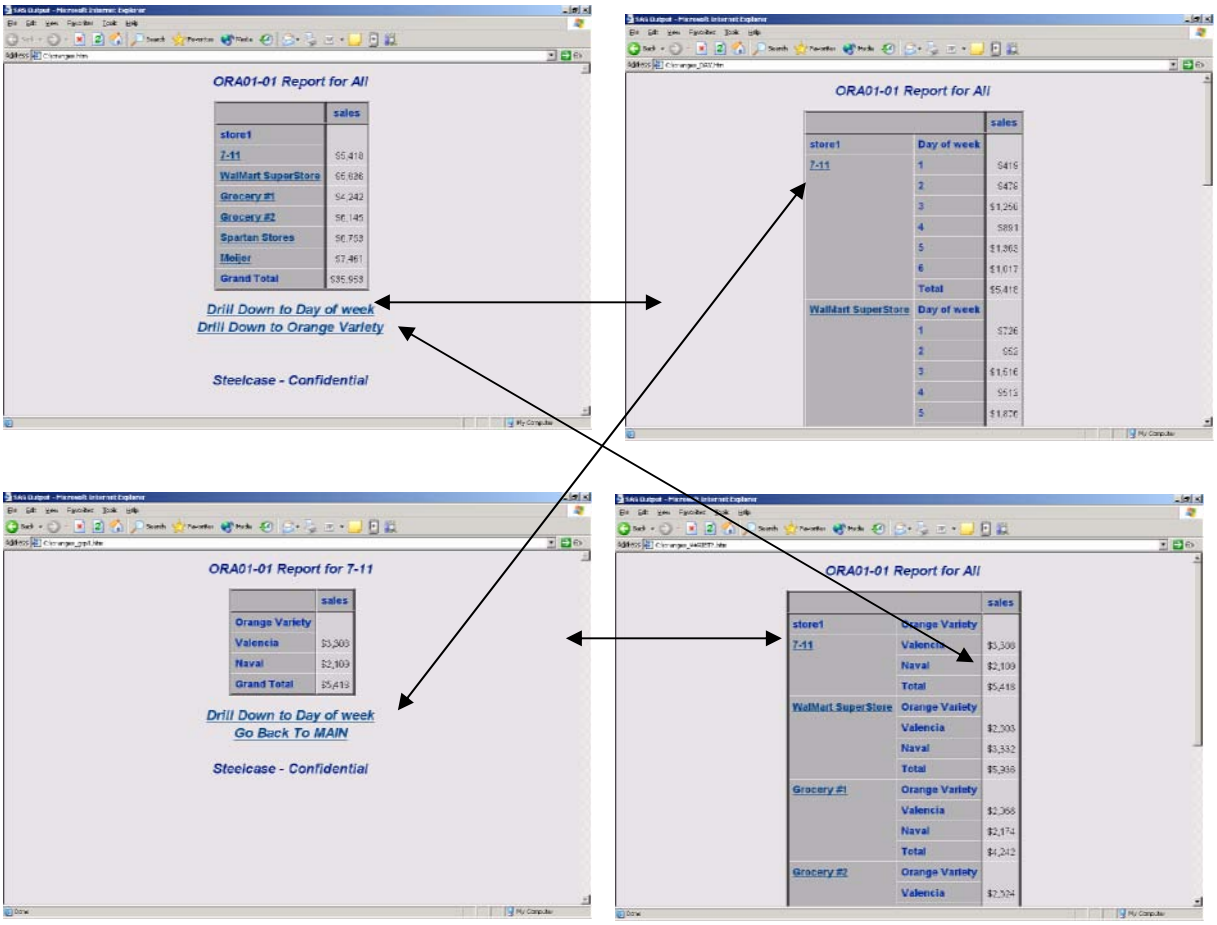
VIEWING THE RESULTS

With this as a foundation for web reporting, exciting opportunities abound. The Appendix of this paper contains macro code to create automatic drilldowns from any table. This code, appropriately named the "drilldown" macro, utilizes the links in both footnotes and formats.

This drilldown macro will create many HTML files that interconnect. Below is documentation, along with a few of the pages resulting from the macro calls:

Drilldown Macro Parameter	Description
IN	The name of the input SAS table.
FIRSTVAR	The first classification field to be included in the report. This is the starting place for all drill-downs.
NEXTVARS	This is a list of all additional classification fields. The first two will have drill-down capabilities.
SUMVARS	The numeric fields that are being summarized.
FILE_PREFIX	All file names will begin with the prefix specified here.

```
%drilldown(in=ORANGES, firstvar=store,
           nextvars=variety day,
           sumvars=sales,
           file_prefix=oranges);
```



CONCLUSION

Using HTML code within BASE SAS coding, any programmer can set up dynamic reports with drill-down capabilities. The possibilities are endless. Taking this to the next level, a programmer can use this knowledge to build an intricate framework for drillable reports, right from BASE SAS.

REFERENCES

An excellent book, as mentioned earlier in this paper, is “Output Delivery System: The Basics”, by Lauren Haworth. This book covers HTML programming and using the ODS in much greater detail than any short paper and presentation can, and I highly recommend it. It is available from SAS Publishing.

In addition, several past SUGI papers contain excellent tips, and are great resources for those wanting to learn more about using HTML with The SAS System. Just a few of these papers are:

- SUGI 26, Paper 185, “HTML for the SAS Programmer”, by Lauren Haworth. Or go to the following link: http://www2.sas.com/proceedings/sugi26/p185-26.pdf
- SUGI 26, Paper 156, “ODS, YES! Odious, NO! – An Introduction to the SAS Output Delivery System”, by Lara Bryant, Sally Muller, and Ray Pass. Or go to the following link: http://www2.sas.com/proceedings/sugi26/p156-26.pdf
- SUGI 26, Paper 1, “Using Styles and Templates to Customize SAS ODS Output”, by Sunil Gupta. Or go to the following link: http://www2.sas.com/proceedings/sugi26/p001-26.pdf

CONTACT INFORMATION

Jeffery D. Gilbert
Steelcase, Inc.
901 44th Street
Grand Rapids, MI 49508
(616) 698-4185
jgilbert@steelcase.com

APPENDIX – DRILLDOWN MACRO

Please note that you are welcome to use and modify this macro in any way, shape, or form, without my permission, as long as you agree to release the author and his company from any liability. You know, the standard legal mumbo jumbo.

```
proc format;
  value orngvar 1='Valencia' 2='Naval';
  value $store 1='7-11' 2='WalMart SuperStore' 3='Grocery #1'
               4='Grocery #2' 5='Spartan Stores' 6='Meijer';
run;

data oranges(drop=price1 price2 sales1 sales2);
  set sasuser.oranges;

  variety=1;      sales=price1*sales1;       output;

  variety=2;      sales=price2*sales2;       output;

  format store $store.   variety orngvar.;
  label variety='Orange Variety';
run;

%macro drilldown2(in=, firstvar=, nextvars=, sumvars=, file_prefix=, goback=, title1=);
  * Drilldown macro;
  * Written for SUGI 30 presentation by Jeffery D. Gilbert, Steelcase, Inc.;
  * Permission is granted to use or modify this series of macros in any way, ;
  * as long as the programmer agrees to release the author and author's company;
  * from any liability.;

  * Put the specified fields in NEXTVARS into a table.;
  data ddvarlist;
    length varname $50;
    %let i=1;
    %let varname=%scan(&nextvars, &i, %str( ) );
    %do %while (%length(&varname));
      varname="%upcase(&varname)";
      output;

      %let i=%eval(&i + 1);
      %let varname=%scan(&nextvars, &i, %str( ) );

    %end;
  run;
  %let nobs=%eval(&i - 1);

  * Get the labels and footnotes attached to the new table, as you would expect.;
  proc sort data=ddvarlist;
    by varname;
  run;
  data ddlabels;
    length lblstatement fmtstatement $1000;
    retain lblstatement ' ' fmtstatement ' ';
    merge ddvarlist(in=ofinterest) ddcontents;
    by varname;
    if ofinterest or
       varname in (%let i=1;
                   %do %while (%length(%scan(&sumvars,&i,%str( ) ) ));
                     "%scan(%upcase(&sumvars), &i, %str( ) )"
                     %let i=%eval( &i + 1 );
                   %end;
                   );
    if label ne ' ' then lblstatement=trim(lblstatement) || ' ' || trim(varname) || '="' ||
trim(label) || '"';
    if not (format in ('$' ' ')) then fmtstatement=trim(fmtstatement) || ' ' || trim(varname) || ' '
|| trim(format) || '.';

    call symput('lblstatement', lblstatement);
```

```
    call symput('fmtstatement', fmtstatement);

    if ofinterest;
  run;
  %put LBL: &lblstatement;
  %put FMT: &fmtstatement;

  proc datasets nolist;
    modify &in;
      %if %length(&lblstatement)>1 %then label &lblstatement; %str(;)
      %if %length(&fmtstatement)>1 %then format &fmtstatement; %str(;)
  quit;

  * Create the FOOTNOTE statements for the master table.;
  %do i=1 %to &nobs;
    data _null_;
      set ddlabels;
      if _n_=&i;

      call symput('varname_label', trim(label));
      call symput('varname', trim(varname));
    run;
        %put NOTE: Now creating footnote &i.;
    footnote&i "<A HREF='&file_prefix._&varname..htm' > Drill Down to &varname_label </A>";
  %end;

  %if %length(&goback)>0 %then %do;
%put NOTE: Attempting to produce footnote &i.;
    footnote&i "<A HREF='&goback..htm'> Go Back To MAIN </A>"; %str(;)
  %end;

%put NOTE: Finish by creating the last two footnotes.;
  %let i=%eval(&i+1);
  %let k=%eval(&i+1);
  footnote&i " ";
  footnote&k "Steelcase - Confidential";

  * Get the summary variables and build the table statement for these fields for the upcoming proc
tabulate.;
  %let i=1;
  %let tablevars=;
  %let varname=%scan(&sumvars, &i, %str( ) );
  %do %while (%length(&varname));
    %let tablevars=&tablevars &varname=' ';

    %let i=%eval(&i + 1);
    %let varname=%scan(&sumvars, &i, %str( ) );
  %end;

  title1 "ORA01-01  "
      %if %length(&title1) >0 %then "%trim(&title1)";;

  * Run first summary.;
  ods html body="&file_prefix..htm" path="c:\";
  proc tabulate data=&in missing format=dollar12.;
    class &firstvar;
    var &sumvars;
    table &firstvar all='Grand Total', (&sumvars)*all=' ' / rts=60;
    keylabel sum=' ';
  run;

  * Run subsequent summaries;
  %do i=1 %to &nobs;
    data _null_;
      set ddvarlist;
      if _n_=&i;
      call symput('currentvar', trim(varname));
    run;
    %put NOTE: Now processing drilldown for &firstvar and &currentvar ;

    ods html body="&file_prefix._&currentvar..htm" path="c:\";
    proc tabulate data=&in missing format=dollar12.;
      class &firstvar &currentvar;
      var &sumvars;
      table (&firstvar*(&currentvar all='Total')) , (&sumvars)*all=' ' / rts=60;
      keylabel sum=' ';

      footnote1 "<A HREF='&file_prefix..htm'> Go back to main level </A>";
      footnote2 " ";
      footnote3 "Steelcase - Confidential";
```

```
    run;

  %end;

  ods html close;

%mend drilldown2;

%macro drilldown(in=, firstvar=, nextvars=, sumvars=, file_prefix=);

  * First summarize the incoming table.;
  proc summary data=&in nway missing;
    class &firstvar &nextvars;
         var &sumvars;
         output out=ddone(drop=_freq_ _type_) sum=;
  run;

  * Get the label for FIRSTVAR and make it HTML ready.;
  proc contents data=&in out=ddcontents noprint;
  run;
  data ddcontents;
    set ddcontents;
    name=upcase(name);

    if name="%upcase(&firstvar)" then do;
      call symput('mainlbl', label);
      call symput('mainfmt', trim(format));
    end;
  run;
  %put MAINLBL: &mainlbl;
  %put MAINFMT: &mainfmt;
  proc sort data=ddcontents(keep=name label format) out=ddcontents(rename=(name=varname));
    by name;
  run;

  proc format cntlout=ddfmt;
    select &mainfmt;
  run;
  data ddone;
    set ddone;

    &firstvar.1="<A HREF='&file_prefix._grp" || trim(left(&firstvar)) || ".htm' > " || put(&firstvar,
&mainfmt..) || " </A>";
    label &firstvar.1="&mainlbl";
  run;

  * Now do the drilldown for the main level.;
  %drilldown2(in=ddone, firstvar=&firstvar.1,
              nextvars=&nextvars,
              sumvars=&sumvars,
              title1=%quote( Report for All &mainlbl ),
              file_prefix=&file_prefix);

  proc sort data=ddone(keep=&firstvar) out=ddtwo nodupkey;
    by &firstvar;
  run;

  %let i=1;
  %let varname=%scan(&nextvars, &i, %str( ) );
  %let secondvarlist=;
  %do %while (%length(&varname));

    %if (&i=1) %then %let secondvar=%upcase(&varname);
    %else %let secondvarlist=&secondvarlist &varname;

    %let i=%eval(&i + 1);
    %let varname=%scan(&nextvars, &i, %str( ) );

  %end;

  proc sql noprint;
    select nobs
           into: num
           from dictionary.tables
      where upcase(memname)='DDTWO';
  quit;
  %put NOTE: DDTWO has &num observations.;

  * Now produce a report for each value of FIRSTVAR.;
  %do j=1 %to &num;
    data _null_;
```

```
      set ddtwo;
      if _n_=&j;

      call symput('current_value', trim(&firstvar) );
      call symput('current_label', trim(put(&firstvar, &mainfmt.. )) );
    run;

    data ddthree;
      set ddone;
      where &firstvar="&current_value";
    run;
    %drilldown2(in=ddthree, firstvar=&secondvar,
                nextvars=&secondvarlist,
                sumvars=&sumvars,
                file_prefix=&file_prefix._grp&current_value,
                title1=%quote( Report for &current_label ),
                goback=&file_prefix);

  %end;

%mend drilldown;

options mprint merror;
%drilldown(in=ORANGES, firstvar=store,
           nextvars=variety day,
           sumvars=sales,
           file_prefix=oranges);
```