You CAN Get There from Here (and Back Again): Adding Hot-link Drill-down Capabilities to ODS HTML Output

Ray Pass
Ray Pass Consulting

ABSTRACT
The SAS Output Delivery System (ODS) can be used to create HTML pages for publishing on the Internet or an Intranet. Right out of the box (and the box is FREE too! – with Base SAS), ODS can be used to create either one page per procedure output, or a collection of HTML pages per procedure. ODS does provide within-procedure HTML navigability, but there is no wholesale provision provided for inter-procedure output linking By using simple SAS MACRO processing combined with the manipulation of **`TITLE`** statements, basic data set variable values and values of variables used as **`CLASS`** variables in TABULATE procedures, you can create whole systems of hot-linked pages with almost full drill-down capabilities. The technique is to simply substitute HTML-navigational-tag enriched data values for those displayed values that you want to act as hot links, and let ODS do the rest! Techniques and examples are shown in this paper, along with data-driven techniques for renaming ODS-HTML generated sequential body file names into more meaningful content-oriented names.

INTRODUCTION

The SAS Output Delivery System (ODS to its friends) has given us a method of preparing HTML formatted versions of output from all SAS procedures that produce output. When used with the new TEMPLATE procedure and the new STYLES feature, all aspects of procedure output, including row, column and cell elements as well as titles for the accompanying output, can be fully formatted.

ODS takes the content results from SAS procedures, or the contents of SAS data sets, and combines these data with a display definition to create an output object. This output object is then routed to an ODS destination. The destinations in production as of V8.1 include HTML, LISTING (standard SAS output), RTF, PRINTER and OUTPUT (standard SAS DATA set). Output objects that are sent to the HTML destination are rendered as HTML formatted documents which can then be read via an HTML browser. The documents consist of titles and HTML tables, and can be afforded the full range of HTML formatting enhancements, including Cascading Style Sheets.

Although the transformation from standard SAS output to HTML formatted output is a major step forward in terms of creating high information content reports which can be made available via the internet or any size intranet to large groups of information consumers, it is still rudimentary in nature because each output is static. One of the seminal definitional features of web information presentation is the ability to connect related informational displays via hot links (hypertext), or drill-down techniques. The hot link capability is accomplished by enriching an element so it can serve as a clickable launching point to a related collection of information. The main enrichment is in the form of the location, or address, of the target display. This paper presents the basis of a methodology for creating these enriched data elements, as well as for the automatic data-driven generation of all the static outputs necessary for a complete system of related informational displays (static report outputs). It also discusses techniques for amplifying (renaming) the sequentially labeled body files created by default by ODS HTML. This technique is totally data-driven and yields meaningful content-specific names for the files.

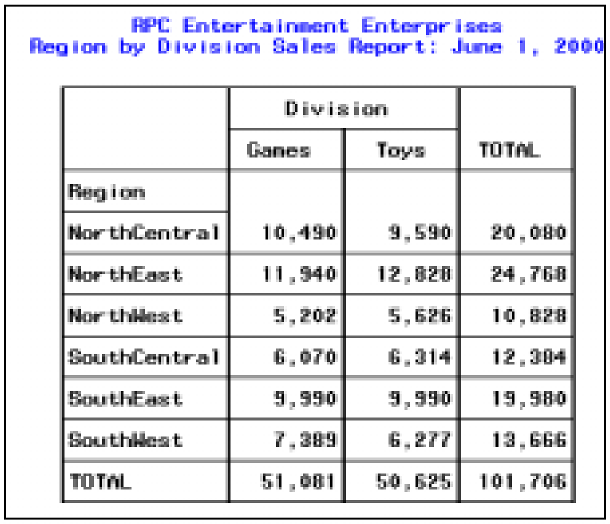
This paper assumes that the reader is already familiar with the basic concepts of ODS. It is not an introduction to these techniques, but is rather an explication of an addition to them. ODS documentation can be found in the SAS publication entitled The Complete Guide to the SAS Output Delivery System, Version 8, although this document is only current up to V8.0. There are also numerous SUGI and Regional SUG papers available on ODS.

SAMPLE DATA

The data to be used throughout his paper consists of periodic sales reports of the fictitious RPC Entertainment Enterprises Corporation. There are six geographical regions in the company (NorthEast, NorthCentral, NorthWest, SouthEast. SouthCentral, SouthWest), and each region is further broken down into the states making up the region. There are two divisions in the company (Games, Toys) and each division is broken down into various items produced by the division.

NON-HTML REPORTS

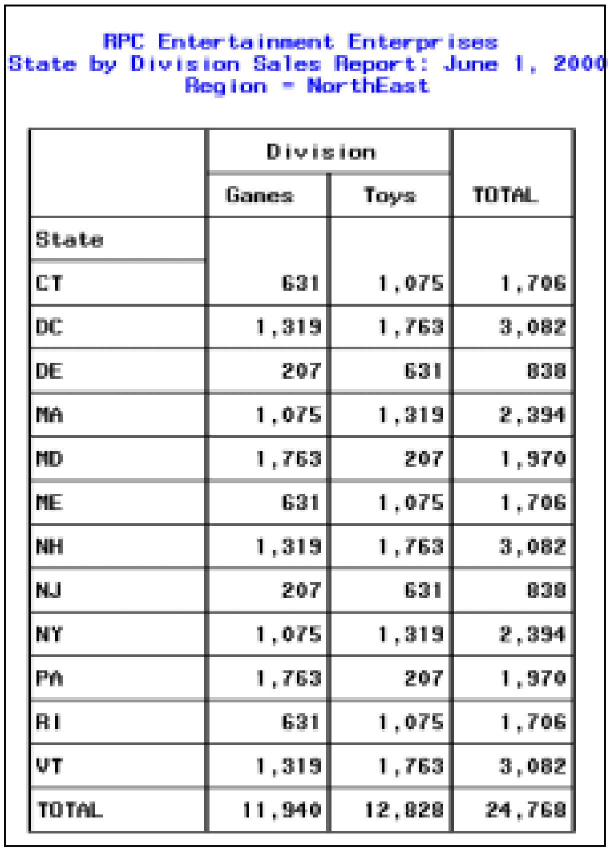
A series of PROC TABULATEs (and accompanying **`TITLE`** statements) can be used without HTML enhancement to produce the needed daily sales reports. Although all of the Region by State by Division by Item data could be presented in their most granular form in one TABULATE output, the end goal here is a hypertext system of related reports. A representative sample of non-HTML modular reports is presented here. Only one of the six (one for each Region) possible *State by Division* reports, one of the two (one for each Division) possible *Region by Item* reports, and one of the 12 (one for each Region-Division combination) possible *State by Item* reports are shown. These are found in Figs 1-4.



RPC Entertainment Enterprises
Region by Division Sales Report: June 1, 2000

	Division		
	Games	Toys	TOTAL
Region			
NorthCentral	10,490	9,590	20,080
NorthEast	11,940	12,828	24,768
NorthWest	5,202	5,626	10,828
SouthCentral	6,070	6,314	12,384
SouthEast	9,990	9,990	19,980
SouthWest	7,389	6,277	13,666
TOTAL	51,081	50,625	101,706

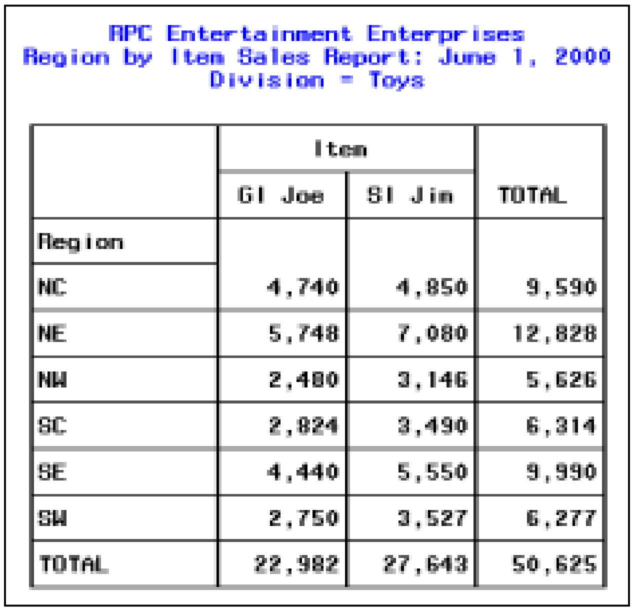
Fig 1. Non-HTML Report
Region by Division



RPC Entertainment Enterprises
State by Division Sales Report: June 1, 2000
Region = NorthEast

	Division		
	Games	Toys	TOTAL
State			
CT	631	1,075	1,706
DC	1,319	1,763	3,082
DE	207	631	838
MA	1,075	1,319	2,394
MD	1,763	207	1,970
ME	631	1,075	1,706
NH	1,319	1,763	3,082
NJ	207	631	838
NY	1,075	1,319	2,394
PA	1,763	207	1,970
RI	631	1,075	1,706
VT	1,319	1,763	3,082
TOTAL	11,940	12,828	24,768

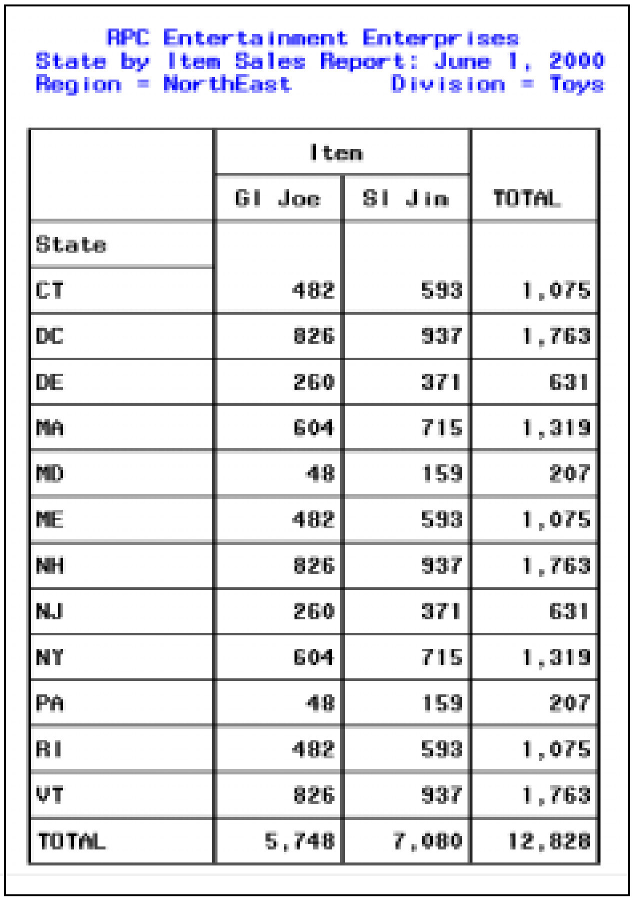
Fig 2. Non-HTML Report
State by Division (Region: NorthEast)



RPC Entertainment Enterprises
Region by Item Sales Report: June 1, 2000
Division = Toys

	Item		
	GI Joe	SI Jim	TOTAL
Region			
NC	4,740	4,850	9,590
NE	5,748	7,080	12,828
NW	2,480	3,146	5,626
SC	2,824	3,490	6,314
SE	4,440	5,550	9,990
SW	2,750	3,527	6,277
TOTAL	22,982	27,643	50,625

Fig 3. Non-HTML Report
Region by Item (Division: Toys)



RPC Entertainment Enterprises
State by Item Sales Report: June 1, 2000
Region = NorthEast Division = Toys

	Item		
	GI Joe	SI Jim	TOTAL
State			
CT	482	593	1,075
DC	826	937	1,763
DE	260	371	631
MA	604	715	1,319
MD	48	159	207
ME	482	593	1,075
NH	826	937	1,763
NJ	260	371	631
NY	604	715	1,319
PA	48	159	207
RI	482	593	1,075
VT	826	937	1,763
TOTAL	5,748	7,080	12,828

Fig 4. Non-HTML Report
State by Item (Region: NorthEast,
Division: Toys)

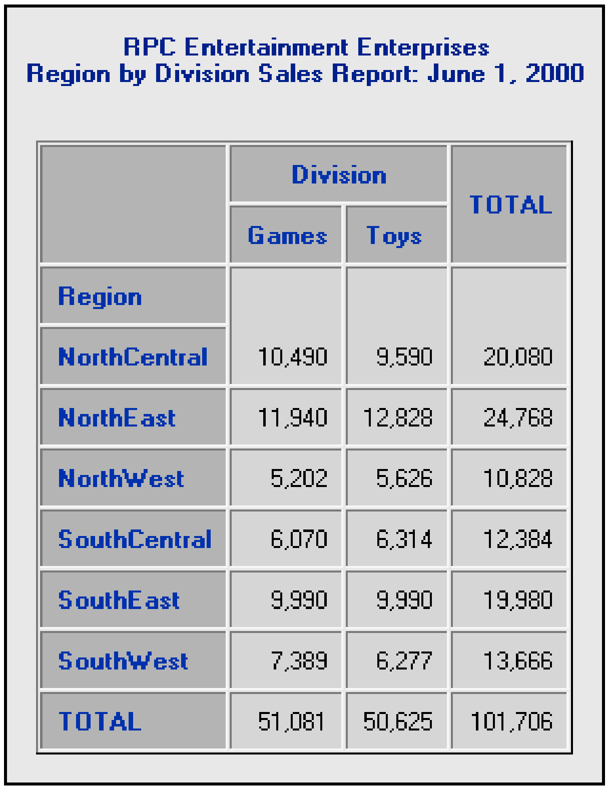
SIMPLE ODS HTML OUTPUT

To use ODS to produce HTML output instead of the standard output listings seen above, we would issue an `ODS HTML` statement before running the various TABULATE procedures. As this paper assumes the basic mechanics and syntax of ODS are known to the reader, it will not go into great detail about the ODS code used herein.

Outputs of the rudimentary use of ODS code as viewed via an HTML browser are shown in Figs 5-8. Full HTML table formatting, including font and color choices, etc. is available via the TEMPLATE procedure. The output presented here uses default formatting values.

The results of using the `ODS HTML` statement are a series of independent HTML documents containing the output from the TABULATE procedure translated into HTML tagged code. The names of these output file are contained as parameters in the ODS code. As an example, the actual names of the HTML files as displayed via browser in Figs 5-8 could be respectively:

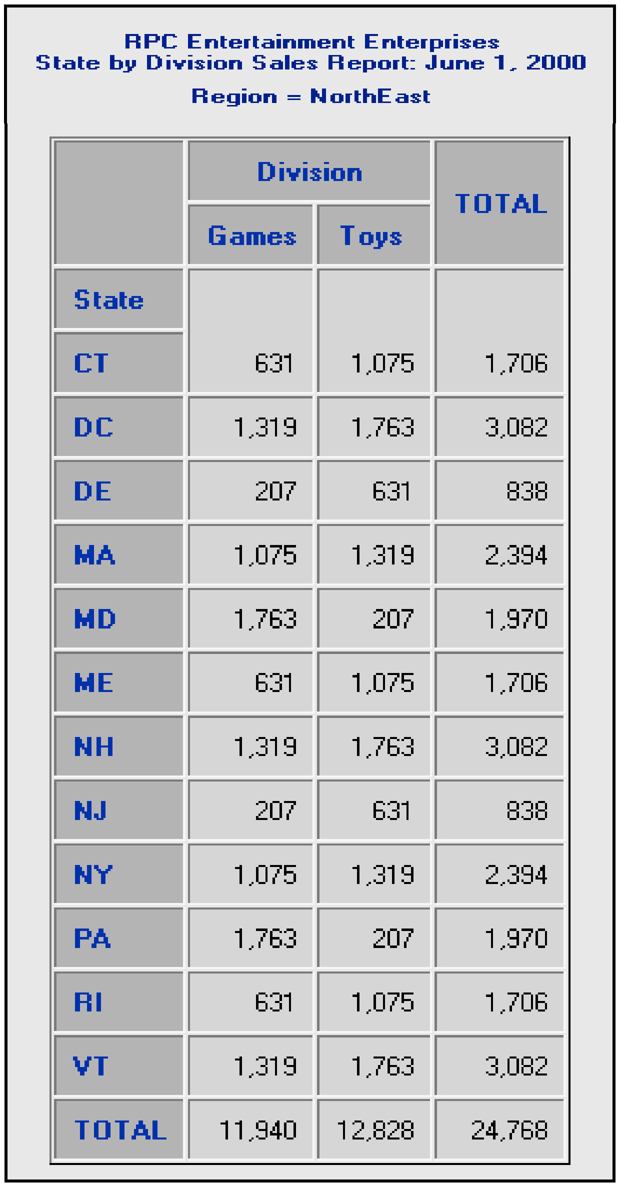
simple 2000-06-01-regxdiv.htm
simple 2000-06-01-stxdiv-NE.htm
simple 2000-06-01-regxitm-Toys.htm
simple 2000-06-01-stxitm-NE-Toys.htm.



RPC Entertainment Enterprises
Region by Division Sales Report: June 1, 2000

	Division		TOTAL
	Games	Toys	
Region			
NorthCentral	10,490	9,590	20,080
NorthEast	11,940	12,828	24,768
NorthWest	5,202	5,626	10,828
SouthCentral	6,070	6,314	12,384
SouthEast	9,990	9,990	19,980
SouthWest	7,389	6,277	13,666
TOTAL	51,081	50,625	101,706

Fig 5. Simple HTML Report Region by Division



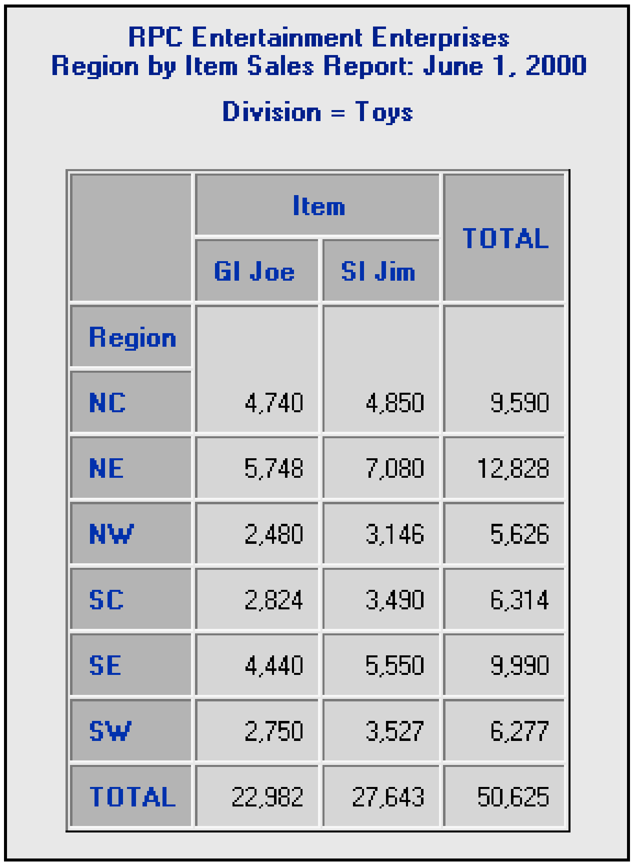
RPC Entertainment Enterprises
State by Division Sales Report: June 1, 2000
Region = NorthEast

	Division		TOTAL
	Games	Toys	
State			
CT	631	1,075	1,706
DC	1,319	1,763	3,082
DE	207	631	838
MA	1,075	1,319	2,394
MD	1,763	207	1,970
ME	631	1,075	1,706
NH	1,319	1,763	3,082
NJ	207	631	838
NY	1,075	1,319	2,394
PA	1,763	207	1,970
RI	631	1,075	1,706
VT	1,319	1,763	3,082
TOTAL	11,940	12,828	24,768

Fig 6. Simple HTML Report State by Divison (Region: NorthEast)

ENHANCED ODS HTML OUTPUT

Now that we have the individual reports rendered as HTML documents, the next step is to create the navigational tools to be able to go from one report to another by clicking on a hot link. This part is the crux of the whole method presented here, and is actually really very simple. All that needs to be done is to create a set of alternate variables from which the tables are constructed. The change that is necessary is to enhance each data item (that is to be displayed as a hot link) with additional location information contained in HTML tags, specifically HREF tags. As an example, each region is



RPC Entertainment Enterprises
Region by Item Sales Report: June 1, 2000

Division = Toys

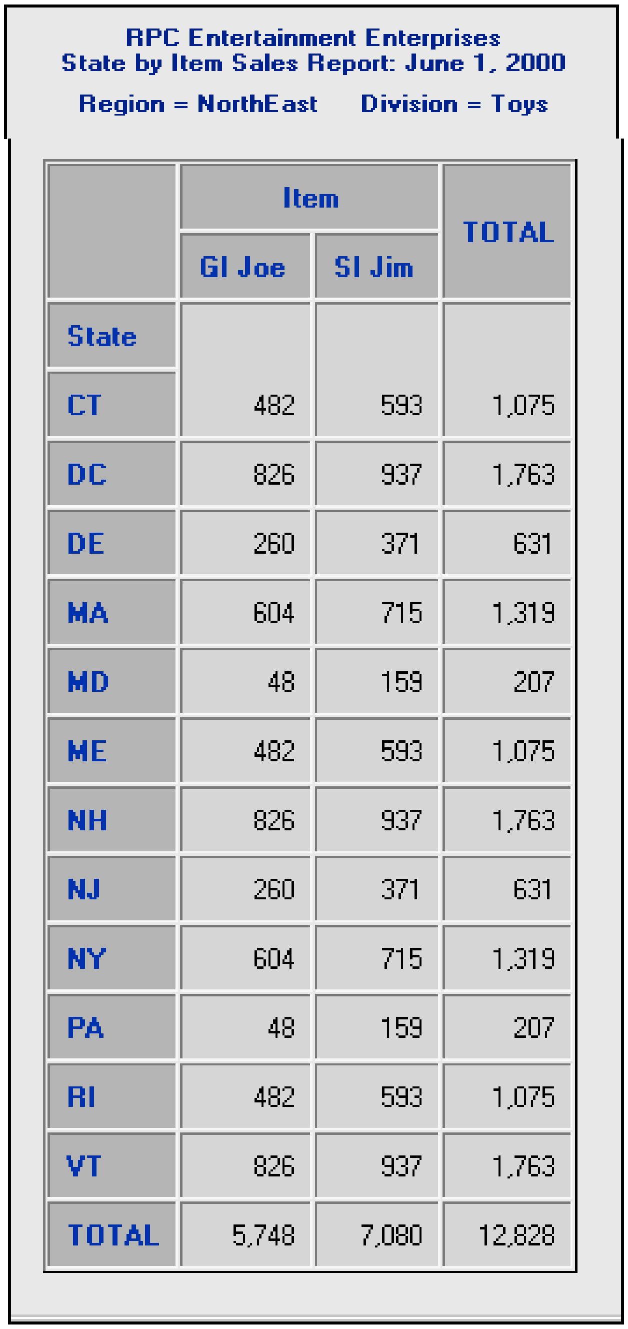
	Item		TOTAL
	GI Joe	SI Jim	
Region			
NC	4,740	4,850	9,590
NE	5,748	7,080	12,828
NW	2,480	3,146	5,626
SC	2,824	3,490	6,314
SE	4,440	5,550	9,990
SW	2,750	3,527	6,277
TOTAL	22,982	27,643	50,625

**Fig 7. Simple HTML Report
Region by Item (Division: Toys)**

actually coded as a two-byte character variable called **REG** with values of: NC, NE, NW, SE, SC, or SW. The non-HTML output uses a user-defined format, **$REGFMT.**, to display expanded names via a **FORMAT** statement in the TABULATE code. If a separate *State by Division* HTML table was created for each region, they could have file names of:

enhanced 2000-06-01-stxdiv-NC.htm
enhanced 2000-06-01-stxdiv-NE.htm
enhanced 2000-06-01-stxdiv-NW.htm
enhanced 2000-06-01-stxdiv-SC.htm
enhanced 2000-06-01-stxdiv-SE.htm
enhanced 2000-06-01-stxdiv-SW.htm

If we create an alternate variable **REG2** by surrounding the old values of **REG** with the above path names along with the appropriate HTML code, we could then produce HTML clickable links in our TABULATE output by using **REG2** in the TABULATE code instead of **REG**. We would also create a variable **REGEXT** as a **FORMAT** expanded version of **REG**. The code to create these variables would look something like:



RPC Entertainment Enterprises
State by Item Sales Report: June 1, 2000
Region = NorthEast Division = Toys

	Item		TOTAL
	GI Joe	SI Jim	
State			
CT	482	593	1,075
DC	826	937	1,763
DE	260	371	631
MA	604	715	1,319
MD	48	159	207
ME	482	593	1,075
NH	826	937	1,763
NJ	260	371	631
NY	604	715	1,319
PA	48	159	207
RI	482	593	1,075
VT	826	937	1,763
TOTAL	5,748	7,080	12,828

**Fig 8. Simple HTML Report
State by Item (Region: NE
Division: Toys)**

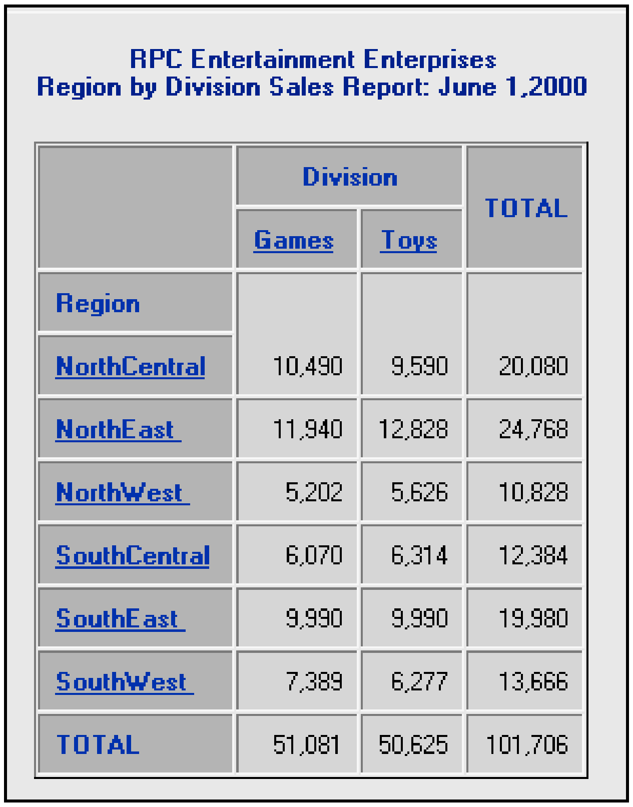
```
regext = put(reg, $regfmt.);

reg2 = "<A HREF='enhanced "
       || "2000-06-01-stxdiv-"
       || trim(reg)
       || ".htm'>"
       || regext
       || '</A>';
```

This would produce a converted value for NE of:

<A HREF='enhanced 2000-06-01-stxdiv-NE.htm'>NorthEast</A> .

Actually, there would have to be two alternate versions of **REG**, a **REG2** and a **REG3**, because clicking on a region name in the *Region by Division* table would have to link to a *State by Division* table for the clicked region, whereas clicking on the same region name in a *Region by Item* table would link to a *State by Item* table for the clicked region. Since the names of the two target files would be different, the values of **REG2** and **REG3**, both based on **REG**, would be different. Two alternate versions of **DIV** would be needed as well.

If the new versions of **REG** and **DIV** were used in the same TABULATE code that was used to produce the tables shown in Figs 5-8, ODS would cause the region and division names to be displayed as clickable HTML hot links. The resulting HTML output files as seen via an HTML browser are shown in Figs. 9-12.



RPC Entertainment Enterprises
Region by Division Sales Report: June 1,2000

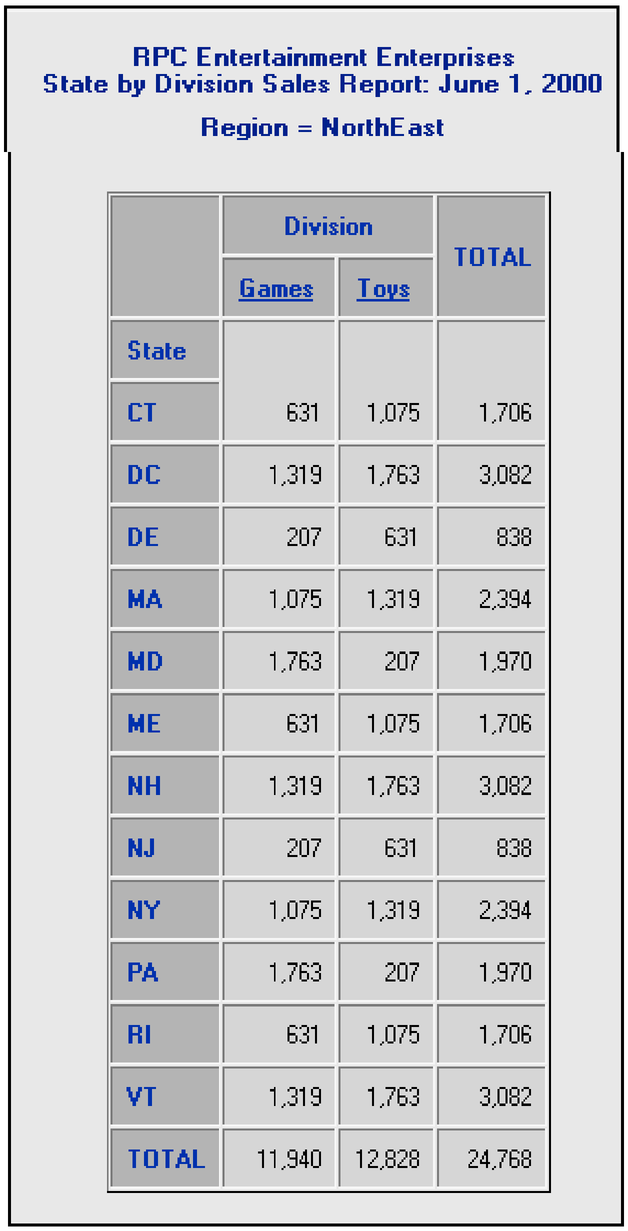
	Division		TOTAL
	Games	Toys	
Region			
NorthCentral	10,490	9,590	20,080
NorthEast	11,940	12,828	24,768
NorthWest	5,202	5,626	10,828
SouthCentral	6,070	6,314	12,384
SouthEast	9,990	9,990	19,980
SouthWest	7,389	6,277	13,666
TOTAL	51,081	50,625	101,706

Fig 9. Enhanced HTML Report
Region by Division

That's basically all there is to it. Just substitute enhanced data values for those values that you want to act as hot links and let ODS do the rest. There is more to the overall method to deal with, but that is the basic technique.

When this method is actually used on a periodic basis to create the reports, the **TODAY()** function is used for the dates, although its output format is modified with the following statement:

```
%let today1=%sysfunc(today(),yymmdd10.);
```



RPC Entertainment Enterprises
State by Division Sales Report: June 1, 2000

Region = NorthEast

	Division		TOTAL
	Games	Toys	
State			
CT	631	1,075	1,706
DC	1,319	1,763	3,082
DE	207	631	838
MA	1,075	1,319	2,394
MD	1,763	207	1,970
ME	631	1,075	1,706
NH	1,319	1,763	3,082
NJ	207	631	838
NY	1,075	1,319	2,394
PA	1,763	207	1,970
RI	631	1,075	1,706
VT	1,319	1,763	3,082
TOTAL	11,940	12,828	24,768

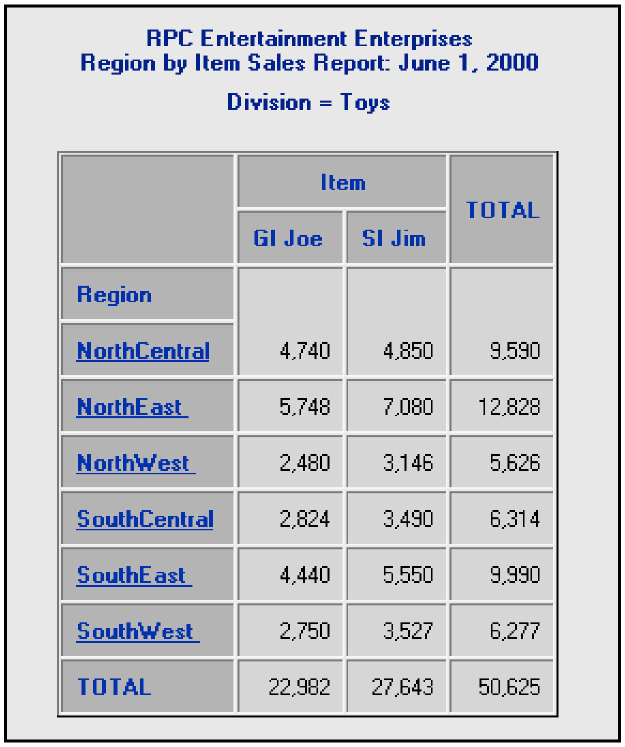
Fig 10. Enhanced HTML Report
State by Division (Region: NE)

This creates date values of the form 2000-06-01. The actual code for the **REG2** assignment statement is then:

```
reg2 = "<A HREF='enhanced "
       || "&today1.-stxdiv-"
       || trim(reg)
       || ".htm'>"
       || regext
       || '</A>';
```

Another date transformation is used to get the formatted date in the titles (each **TITLE** statement has **&TODAY2** in it):

```
%let today2=%sysfunc(today(),worddate12.);
```



RPC Entertainment Enterprises
Region by Item Sales Report: June 1, 2000

Division = Toys

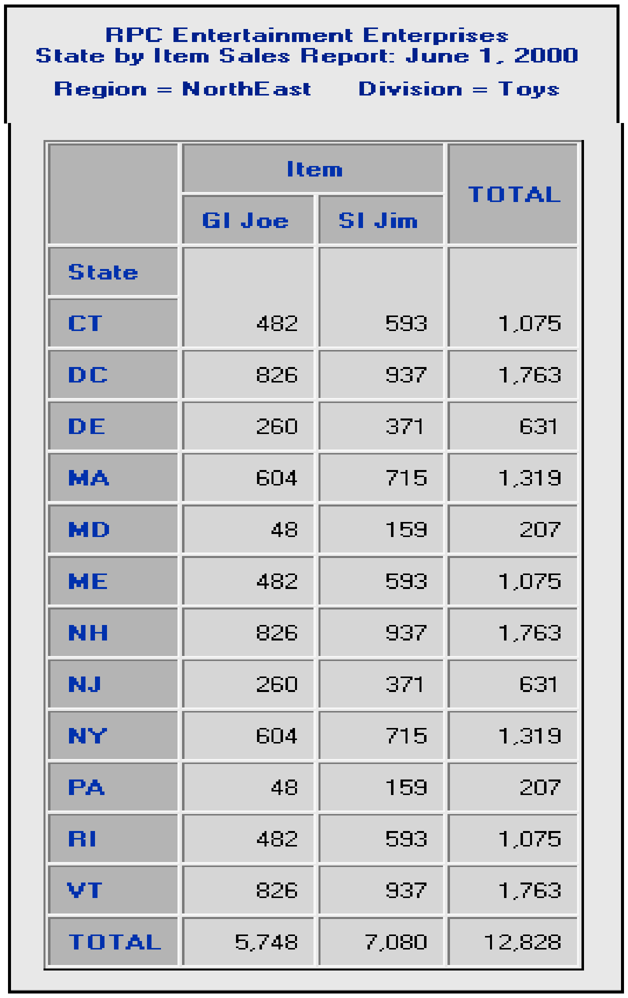
	Item		TOTAL
	GI Joe	SI Jim	
Region			
NorthCentral	4,740	4,850	9,590
NorthEast	5,748	7,080	12,828
NorthWest	2,480	3,146	5,626
SouthCentral	2,824	3,490	6,314
SouthEast	4,440	5,550	9,990
SouthWest	2,750	3,527	6,277
TOTAL	22,982	27,643	50,625

**Fig 11. Enhanced HTML Report
Region by Item (Division: Toys)**

ADDITIONAL ODS HTML OUTPUT

OK, now we know how to create hot links in the row and column headers in the ODS HTML output tables. But, there's more to it than that. We can also create a stand-alone HTML Table of Contents by passing PROC PRINT, PROC REPORT, or even PROC TABULATE output through ODS HTML processing. This notion of creating separate stand-alone ODS HTML Tables of Contents is covered more fully in another paper by the current author entitled I'll Have the Tabulates a la ODS Please, With a Table of Contents on the Side (found in the Proceedings of the NESUG 2000 and SESUG 2000 conferences). The following macro code creates the output *Table of Contents* page as shown in Fig. 13 (as viewed through an HTML browser.) The string values for **REP** and the **TITLE** statement are broken here to fit into the column requirements of this paper; in actual code they are each one continuous string.

```
%macro repstoc;
   data reps;
      length rep $ 77;
      label  rep="Reports for &today2";
      rep="<A HREF='&today1.-regxdiv.htm'>
      Region by Division Sales Report</A>";
      output;
      rep="<U>Report 2 (<I>non-operational
      </I>)</U>";
      output;
      rep="<U>Report 3 (<I>non-operational
      </I>)</U>";
      output;
run;
```

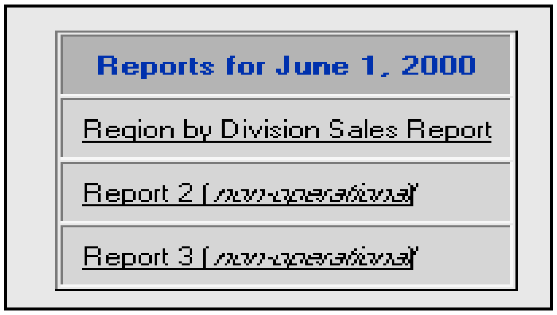


RPC Entertainment Enterprises
State by Item Sales Report: June 1, 2000
Region = NorthEast   Division = Toys

	Item		TOTAL
	GI Joe	SI Jim	
State			
CT	482	593	1,075
DC	826	937	1,763
DE	260	371	631
MA	604	715	1,319
MD	48	159	207
ME	482	593	1,075
NH	826	937	1,763
NJ	260	371	631
NY	604	715	1,319
PA	48	159	207
RI	482	593	1,075
VT	826	937	1,763
TOTAL	5,748	7,080	12,828

**Fig 12. Enhanced HTML Report
State by Item (Region: NE
Division: Toys)**

```
   *-----------------------------------------;
   ods html path = repout
            body = "&today1._reps_toc.htm";
   *-----------------------------------------;
   title1 "<H4><A  HREF='_cal.htm'>Calendar
           </A></H4>";
   run;
   *-----------------------------------------;
   proc print data=reps noobs label;
   run;
   *-----------------------------------------;
   ods html close;
%mend;
```



Reports for June 1, 2000
Region by Division Sales Report
Report 2 (*non-operational*)
Report 3 (*non-operational*)

Fig 13. Report Table of Contents

We can also use ODS HTML processing to create a *Calendar* page (as viewed via an HTML browser in Fig 14). New reports are created every two weeks in this example with the creation date used as part of the file name for each table (HTML file). The code used to create the calendar is presented below. The method also includes a start date for the first month to be displayed.

```
%macro calendar;
  data day;
     length date $ 38;
     date="<A HREF=&today1.-reps-toc.htm>"
          ||put(today(),day2.)||"</A>";
     datex = today();                              run;
  *-----------------------------------------------;
  proc datasets;
     append base=hot._days new=day;                run;
  *-----------------------------------------------;
  proc sort data=hot._days nodupkey;  by datex;    run;
  *-----------------------------------------------;
  proc format;
     value $miss (default=38) ' '='&#160;';
  *-----------------------------------------------;
  data alldays(drop=start d);
    length date $ 34;
    start = '01jun2000'd;
    d     = 0;
    do until (datex=intnx('month',today(),1)-1);
       datex = start + d;
       date  = put(datex,day2.);
       year  = year(datex);
       month = month(datex);
       output;
       d + 1;
    end;                                           run;
  *-----------------------------------------------;
  proc sort data=alldays;       by datex;          run;
  *-----------------------------------------------;
  data alldays;
     update alldays hot._days;  by datex;          run;
  *-----------------------------------------------;
  proc sort data=alldays;  by year month datex;    run;
  *-----------------------------------------------;
  data cal(keep=mon_yr sun mon tue wed thu fri sat);
    set alldays end=lastrec;
    by year month;
    length          sun mon tue wed thu fri sat $ 34;
    array days{7} $ sun  mon  tue  wed  thu  fri  sat;
    array daysx{7}  sunx monx tuex wedx thux frix satx;
    format          sun mon tue wed thu fri sat $miss.;
    retain days;
    mon_yr = intnx('month',datex, 0);

    if      weekday(datex)=1 then do;
            sunx=datex;  sun=date;  end;
    else if weekday(datex)=2 then do;
            monx=datex;  mon=date;  end;
    else if weekday(datex)=3 then do;
            tuex=datex;  tue=date;  end;
    else if weekday(datex)=4 then do;
            wedx=datex;  wed=date;  end;
    else if weekday(datex)=5 then do;
            thux=datex;  thu=date;  end;
    else if weekday(datex)=6 then do;
            frix=datex;  fri=date;  end;
    else if weekday(datex)=7 then do;
            satx=datex;  sat=date;  end;

    if last.month or lastrec then
    do i=(weekday(datex)+1) to 7;
       if daysx[i] lt datex then days[i] = ' ';
    end;

    if first.month or _n_=1 then
    do i=1 to (weekday(datex)-1);
       days[i] = ' ';
    end;

    if weekday(datex) = 7 or last.month or lastrec;
run;
```

```
   *-----------------------------------------------;
  ods html path = repout
           body = '_cal.htm';
   *-----------------------------------------------;
   proc report nowd data=cal;
      columns mon_yr sun mon tue wed thu fri sat;

      define mon_yr / order noprint;

      break after mon_yr / page;

      compute before _page_;
         line mon_yr monyy7.;
      endcomp;
   run;
   *-----------------------------------------------;
   ods html close
;%mend calendar;
```

HTML ENHANCED TITLES

The next parts of the process needed to create the fully navigational system are hot-link enhanced titles. These are easy to do because they come along for a free ride when ODS is used to create HTML. The trick is to make sure that the correct titles appear at the top of each page. Parts of the titles can be generally modularized as follows:

```
%let mcal = <A HREF=_cal.htm>Calendar</A>;
%let mrep = <A HREF=&today1.-reps-toc.htm>
               Reports</A>;
```

These would be used to produce the hot links to go to the *Calendar* page and a day-specific *Table of Contents* page, as in:

```
title1 "<H3>RPC Entertainment Enterprises
        <BR>Region by Division Sales
            Report: &today2</H3>";
title2 "<H4>&mcal &mrep</H4>";
```

Other titles would be constructed using the same general format.

DATA-DRIVEN MACRO AUTOMATION

To make the whole system automatic and data driven, it is all contained in a system of macros, which works as follows. **%CALENDAR** is used to recreate the HTML *Calendar* page each report day, and **%REPSTOC** is run to create a new date-specific HTML *Table of Contents* page. Each day that the program is run the source data set is re-created as a SAS data set called **HLDATA**. This data set contains variables **REG**, **DIV** and **SALES**. An alternate data set, **HLDATA2** is created with the converted variables **REGEXT**, **REG2**, **REG3**, **DIV2** and **DIV3**, along with **SALES**. Next, each type of individual TABULATE report is created as an HTML table and sent to a .htm file. The code for each different type of TABULATE (*Region by Division, State by Division, Region by Item, State by Item*) is contained in a table macro (**%REGXDIV**, **%STXDIV**, **%REGXITM**, **%STXITM**) along with its ODS code. The code for one of these table macros (**%STXDIV**) is as follows (the rest are similar):

```
%macro stxdiv;
   *-------------------------------------;
   ods html path   =repout
            body   ="&today1.-stxdiv-1.htm"
            newfile=bygroup;
   *-------------------------------------;
   proc sort data=hldata2;
      by regext;
   run;
   *-------------------------------------;
   title1 "<H3>RPC Entertainment
           Enterprises
          <BR>State by Division Sales
           Report: &today2
          <BR>Region=#byval(regext)</H3>";
   title2 "<H4>&mcal &t_sp &mrep &t_sp
           <A HREF=&today1.-regxdiv.htm>
            All Regions</A></H4>";
   run;
   *-------------------------------------;
   proc tabulate data=hldata2 missing;
      by regext;
      class  state div3;
      var    sales;
      table  state='State' all='TOTAL',
             (div3='Division' all='TOTAL')
             *sales=' '*sum=' '*f=comma32.;
   run;
   *-------------------------------------;
   ods html close;
%mend stxdiv;
```

A few notes about this macro (and its sister macros for other table types) are in order. The macro variable **&t_sp** is created early in the program as follows:

```
%let t_sp = %nrstr(&nbsp;&nbsp;&nbsp;);
```

and is used to insert visible spaces in the HTML output. Also, note that the dataset is pre-sorted by the categorical variable for which drill-down tables are being created (**REGEXT** in this case.) This sorting, in conjunction with the ODS **NEWFILE=BYGROUP** option (new in V 8.1) is what creates a separate HTML file for each byvar level. Note also that the current byvar value is inserted in the title by means of the **#BYVAL** notation (the **NOBYLINE** option is also in effect.)

Another major feature employed in this system is the renaming of the ODS produced files from a simple sequentially suffixed set (name1, name2, …) to a meaningfully suffixed set. This is accomplished by combining data-driven pre-processing of the data with post-processing file naming via operating system commands. The code for the first part of the process is contained in macro **%MVARS**. This macro is totally data-driven and contains information about the values of **REG** and **DIV** present in the source data.

```
%macro mvars;
   %global mrdcount mrcount mdcount regdivs
         mregs mdivs mrdnums mrnums mdnums;
   proc sql;
      create table regdivs as
      select   distinct
               reg,
               div
      from     hot._hldata
      order by reg,
               div;

      create table regs as
      select   distinct reg
      from     regdivs
      order by reg;

      create table divs as
      select   distinct div
      from     regdivs
      order by div;
   quit;
   *-------------------------------------;
   data regdivs(drop=reg div);
      set regdivs;
      regdiv = trim(reg)||'-'||trim(div);
      num    = _n_;
   run;
   *-------------------------------------;
   data regs;
      set regs;
      num    = _n_;
   run;
   *-------------------------------------;
   data divs;
      set divs;
      num    = _n_;
   run;
   *-------------------------------------;
   proc sql noprint;
      select count(*),
             regdiv,
             num
      into   :mrdcount,
             :mregdivs separated by '#',
             :mrdnums  separated by '#'
      from   regdivs;

      select count(*),
             reg,
             num
      into   :mrcount,
             :mregs  separated by '#',
             :mrnums separated by '#'
      from   regs;

      select count(*),
             div,
             num
      into   :mdcount,
             :mdivs  separated by '#',
             :mdnums separated by '#'
      from   divs;
   quit;
%mend mvars;
```

The results of the above macro execution are nine macro variables with the following values:

```
    mrdcount  - 12
    mregdivs  – NC-Games#NC-Toys#NE-Games# …
    mrdnums   - 1#2#3#4#5#6#7#8#9#10#11#12

    mrcount   - 6
    mregs     - NC#NE#NW#SC#SE#SW
    mrnums    - 1#2#3#4#5#6

    mdcount   - 2
    mdivs     - Games#Toys
    mdnums    - 1#2
```

The actual values of the macro variables are dependent on the actual data in the data set each day – this is the fully populated version.

The second part of the process uses operating system deletion and renaming commands to change the names of the ODS produced files (this code is PC-based and uses the DOS **DEL** and **REN** commands; in UNIX, you would use the **RM** and **MV** commands, etc.) In the above **%STXDIV** macro, the

```
  body    = "&today1.-stxdiv-1.htm
  newfile = bygroup
```

statements instruct ODS to create a new file for each new byvar level, and to name them (assuming the current date is June 1, 2000 and there are 6 regions)

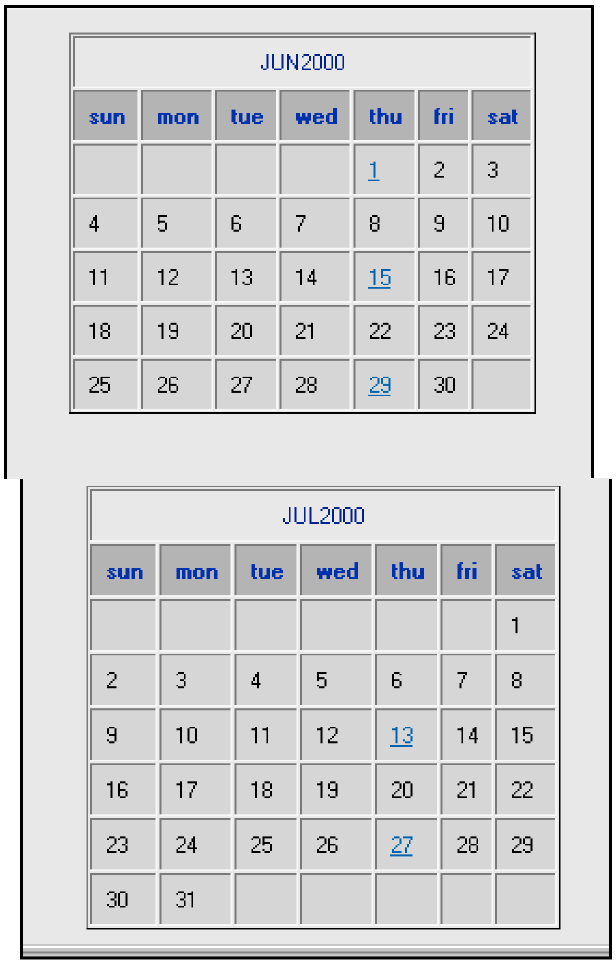
2000-06-01-stxdiv-1.htm through
2000-06-01-stxdiv-6.htm.

This is default ODS behavior, and is not very useful, although fully understandable. ODS has no way of knowing what to call the files it produces unless you tell it. Using the data-driven/created macros shown above, the following code renames all the ODS-produced .htm files meaningfully, based on the values of the appropriate byvars:

```
%macro names;
   %if %sysfunc(fileexist(
               &hldata.\&today1.-stxdiv-1.htm))
   %then %do m = 1 %to &mrcount;
      %let num = %scan(&mrnums,&m,#);
      %let reg = %scan(&mregs ,&m,#);

      %sysexec del
             &hldata.\&today1.-stxdiv-&reg..htm;
      %sysexec ren
             &hldata.\&today1.-stxdiv-&num..htm
                      &today1.-stxdiv-&reg..htm;
   %end;

   %if %sysfunc(fileexist(
               &hldata.\&today1.-regxitm-1.htm))
   %then %do m = 1 %to &mdcount;
      %let num = %scan(&mdnums,&m,#);
      %let div = %scan(&mdivs ,&m,#);

      %sysexec del
            &hldata.\&today1.-regxitm-&div..htm;
      %sysexec ren
            &hldata.\&today1.-regxitm-&num..htm
                      &today1.-regxitm-&div..htm;
   %end;

   %if %sysfunc(fileexist(
               &hldata.\&today1.-stxitm-1.htm))
   %then %do m=1 %to &mrdcount;
      %let num = %scan(&mrdnums ,&m,#);
      %let rd  = %scan(&mregdivs,&m,#);

      %sysexec del
             &hldata.\&today1.-stxitm-&rd..htm;
      %sysexec ren
             &hldata.\&today1.-stxitm-&num..htm
                      &today1.-stxitm-&rd..htm;
   %end;
%mend names;
```

For example, 2000-06-01-stxdiv-1.htm becomes 2000-06-01-stxdiv-NC.htm, 2000-06-01-stxdiv-2.htm becomes 2000-06-01-stxdiv-NE.htm, etc. These names now match the names built in to the drill-down HREF names in the TABULATE macros.

The final, fully HTML navigational system, comprised of the calendar, report table of contents and all component table reports, is shown in Figs 14-19, as viewed through an HTML browser.



JUN2000

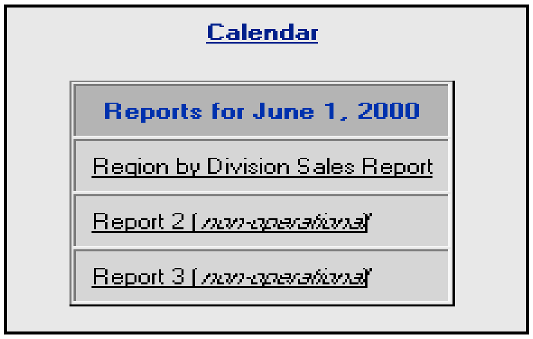
sun	mon	tue	wed	thu	fri	sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

JUL2000

sun	mon	tue	wed	thu	fri	sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Fig 14. Final Calendar

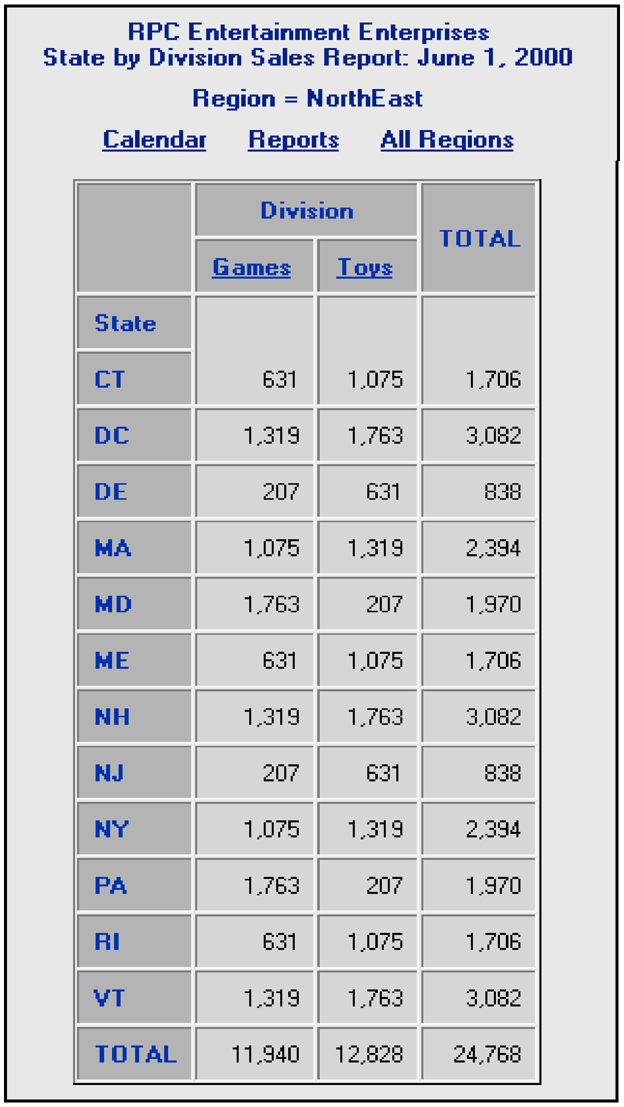
(Figures are placed out of sequential order on this page so as to avoid breaking them across columns.)



Calendar

Reports for June 1, 2000
Region by Division Sales Report
Report 2 (*non-operational*)
Report 3 (*non-operational*)

Fig 15. Final Report Table of Contents



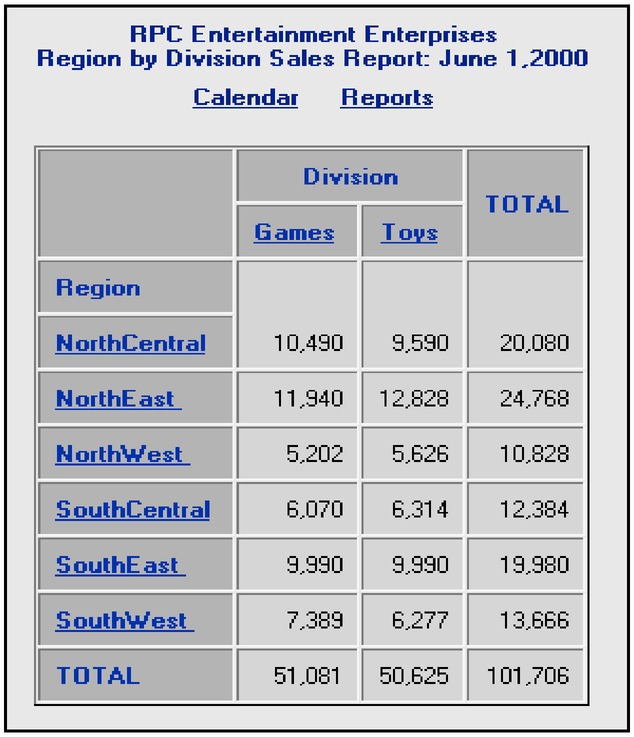
RPC Entertainment Enterprises
State by Division Sales Report: June 1, 2000

Region = NorthEast

Calendar   Reports   All Regions

	Division		TOTAL
	Games	Toys	
State			
CT	631	1,075	1,706
DC	1,319	1,763	3,082
DE	207	631	838
MA	1,075	1,319	2,394
MD	1,763	207	1,970
ME	631	1,075	1,706
NH	1,319	1,763	3,082
NJ	207	631	838
NY	1,075	1,319	2,394
PA	1,763	207	1,970
RI	631	1,075	1,706
VT	1,319	1,763	3,082
TOTAL	11,940	12,828	24,768

Fig 18. Final HTML Report
State by Division (Region: NE)

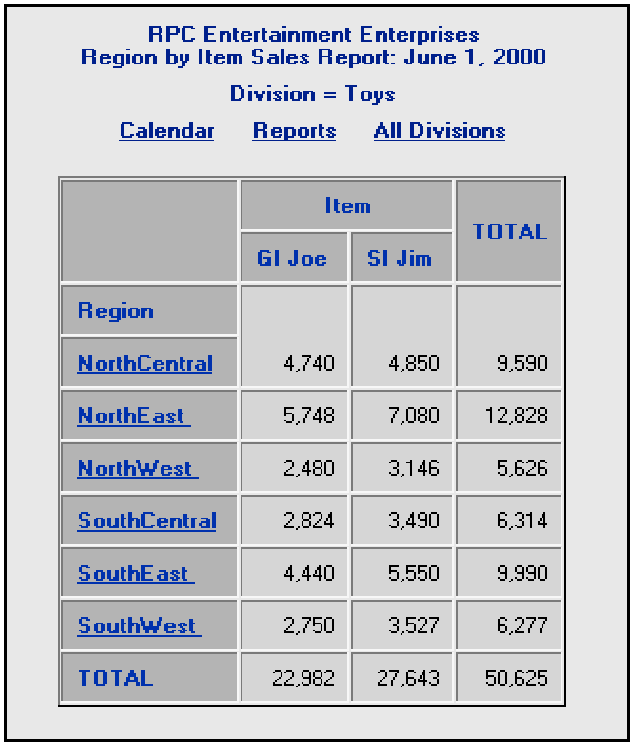


RPC Entertainment Enterprises
Region by Division Sales Report: June 1,2000

Calendar   Reports

	Division		TOTAL
	Games	Toys	
Region			
NorthCentral	10,490	9,590	20,080
NorthEast	11,940	12,828	24,768
NorthWest	5,202	5,626	10,828
SouthCentral	6,070	6,314	12,384
SouthEast	9,990	9,990	19,980
SouthWest	7,389	6,277	13,666
TOTAL	51,081	50,625	101,706

Fig 16. Final HTML Report
Region by Division



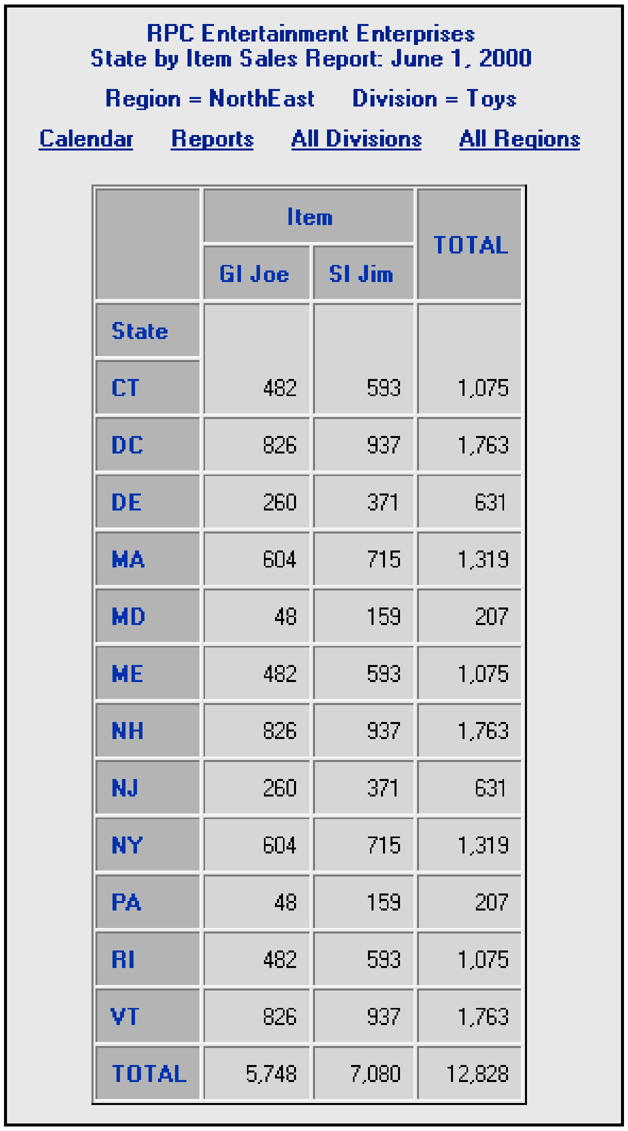
RPC Entertainment Enterprises
Region by Item Sales Report: June 1, 2000

Division = Toys

Calendar   Reports   All Divisions

	Item		TOTAL
	GI Joe	SI Jim	
Region			
NorthCentral	4,740	4,850	9,590
NorthEast	5,748	7,080	12,828
NorthWest	2,480	3,146	5,626
SouthCentral	2,824	3,490	6,314
SouthEast	4,440	5,550	9,990
SouthWest	2,750	3,527	6,277
TOTAL	22,982	27,643	50,625

Fig 17. Final HTML Report
Region by Item (Division: Toys


RPC Entertainment Enterprises
State by Item Sales Report: June 1, 2000

Region = NorthEast   Division = Toys

Calendar   Reports   All Divisions   All Regions

	Item		TOTAL
	GI Joe	SI Jim	
State			
CT	482	593	1,075
DC	826	937	1,763
DE	260	371	631
MA	604	715	1,319
MD	48	159	207
ME	482	593	1,075
NH	826	937	1,763
NJ	260	371	631
NY	604	715	1,319
PA	48	159	207
RI	482	593	1,075
VT	826	937	1,763
TOTAL	5,748	7,080	12,828

Fig 19. Enhanced HTML Report State by Item (Region: NE Division: Toys)

CONCLUSION

This paper has presented an introduction to an optional method of turning the static HTML documents created by the SAS Output Delivery System into a fully functional, HTML drill-down navigational, data-driven system of information display that can be implemented on a routine basis in an Internet or intranet environment. The basic paradigm is to enhance the data being used as input to ODS, so the values displayed can be rendered by an HTML browser as fully functional HTML hot links.

The examples presented herein deal with outputs from runs of TABULATE, PRINT and REPORT procedures, as well as enhanced TITLE statements. In addition, a method is displayed in which default ODS created sequentially suffixed HTML file names are renamed to content-meaningful names via totally automated and data-driven processing. This is an integral part of the entire system.

As far as the author is aware, with the exception of some minor use of PROC FORMAT related techniques, the methods presented in this paper are not documented in the SI collection of distributed literature.

The author has used these techniques successfully on a large nationwide intranet system with extremely productive results. In fact, the initial implementation has spawned numerous other similar systems on the same intranet. Now it's your turn to go out and create your own systems. The tools are there.

REFERENCES

SAS is a registered trademark of the SAS Institute Inc., Cary, NC, USA.

CONTACT INFORMATION

The author can be contacted at:

Ray Pass
Ray Pass Consulting
5 Sinclair Place
Hartsdale, NY 10530

(914) 693-5553
raypass@att.net