Paper 225-2010

# Flexible HTML Reporting Using Advanced Macro Programming and TableEditor Tagset

Smita Sumant, Wyndham Exchange & Rentals, Parsippany, NJ

## ABSTRACT

SAS ODS techniques provide the ability to create HTML outputs and help minimize manual work. However, if you still miss dynamic and interactive Excel functionalities, we have a solution! Now, you can filter, sort and conditional format HTML output just like Microsoft Excel. This paper demonstrates how advanced macro programming techniques and user-defined tagset TableEditor were used to create interactive and efficient HTML reports at RCI, the global leader in timeshare exchange and part of the Wyndham Worldwide family of companies. Advanced macro programming is applied to organize HTML pages and to create a dynamic menu page with hyperlinks. User-defined tagset TableEditor is utilized along with SAS ODS techniques to create Excel like data grids and to perform conditional formatting. The primary goal of this paper is to elaborate these programming techniques in such a way that the readers can immediately implement these methods with minimal effort. This paper is divided into three main sections:

1. Business Need section talks about initial requirements and scope of the project
2. Proposed Solution section discusses challenges faced and how they were overcome using SAS tools.
3. Technical Detail section demonstrates step by step approach towards designing the reporting system along with sample SAS code used in the project

## BUSINESS NEED

RCI launched a new pricing tool in 2009 which allows pricing analysts to change prices on a daily basis for their respective territory responsibilities. Every evening, pricing managers and analysts have to review the pricing changes before they are sent to the system. On average, each analyst makes more than 2,000 changes per day. Therefore, it is not practical to look at each and every update. Managers need a high level report to identify areas that require further review. Also, detailed views are needed for analysts and managers with the flexibility of sorting and filtering in order to easily identify errors. With these business needs, a multi-level, flexible and easy to use reporting system is required to efficiently monitor the pricing process.

## PROPOSED SOLUTION

Major challenges while designing the solution include (a) handling large amounts of data, (b) determining a common delivery point and delivery platform such as Excel, HTML for reports to support offices in USA, Mexico, and Europe and (c) facilitating automated delivery.

These challenges were resolved by using the SAS programming techniques, SAS information delivery portal and SAS BI platform. SAS has the capability to handle large amounts of data successfully (a). The SAS information delivery portal serves as a focal point for Global Revenue Management operations team to access information needed. Therefore, utilizing the SAS Portal as the delivery method was an obvious choice (b). Further, SAS BI platform acts together as self- service reporting system and does not require manual intervention (c).

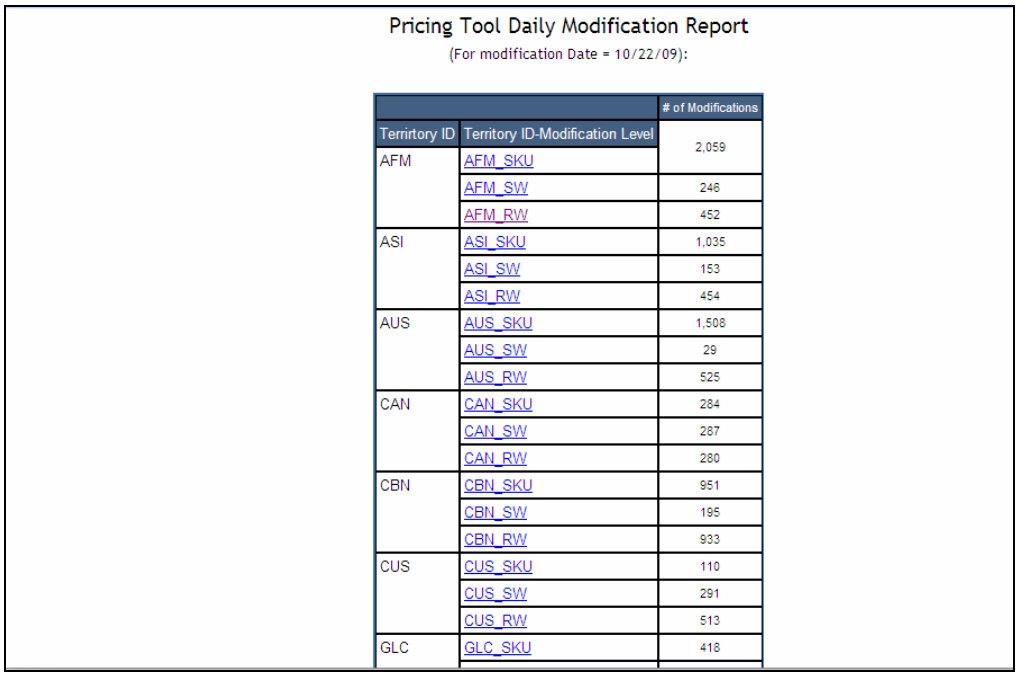
After resolving all the design challenges, next step was to develop the reporting system. A multi level HTML based reporting system was proposed which includes a menu page and individual detailed pages for each territory.

*Please note: All data shown are fake to ensure confidentiality.*

## DYNAMIC MENU PAGE

Figure 1 shows a sample of the menu page. The menu page would summarize information for each territory which would assist managers to identify possible problem areas. Further, the menu page would link to the detailed page to display changes done in the specific territory and details of the changes. This page acts as navigation channel throughout the reporting system. Each individual line representing Territory ID or Modification Level will have a hyperlink to the appropriate detailed page. Creating this page was challenging because the number of territories modified varies over time. Advanced macro programming and proc format is utilized to dynamically list the number of territories modified and create hyperlinks to the listed territories.



Pricing Tool Daily Modification Report

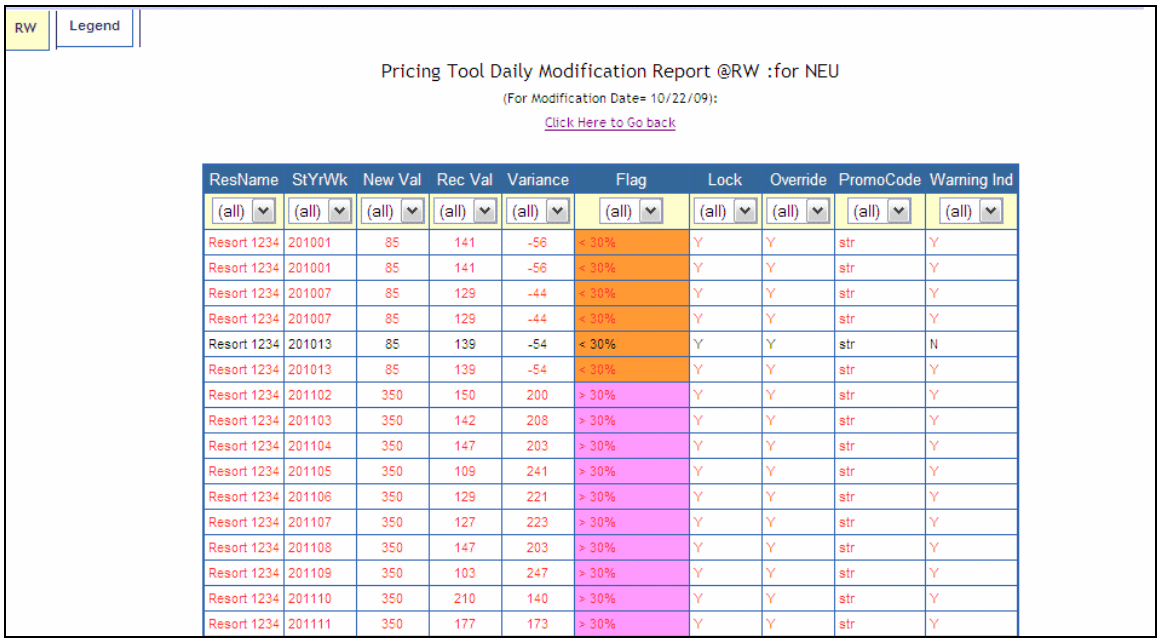
(For modification Date = 10/22/09):

| | | # of Modifications |
|---|---|---|
| Territory ID | Territory ID-Modification Level | |
| AFM | AFM_SKU | 2,059 |
| | AFM_SW | 246 |
| | AFM_RW | 452 |
| ASI | ASI_SKU | 1,035 |
| | ASI_SW | 153 |
| | ASI_RW | 454 |
| AUS | AUS_SKU | 1,508 |
| | AUS_SW | 29 |
| | AUS_RW | 525 |
| CAN | CAN_SKU | 284 |
| | CAN_SW | 287 |
| | CAN_RW | 280 |
| CBN | CBN_SKU | 951 |
| | CBN_SW | 195 |
| | CBN_RW | 933 |
| CUS | CUS_SKU | 110 |
| | CUS_SW | 291 |
| | CUS_RW | 513 |
| GLC | GLC_SKU | 418 |

**Figure 1: Menu Page**

## FLEXIBLE DETAILED PAGE

For the detailed views, one page was created for each territory. These pages have two tabs. The first tab is created to display detailed information such as resort name, start week, new value, recommended value, variance and other related information about selected territory as shown in Figure 2. These detailed pages have Excel like abilities which allow users to filter and sort data. Basic HTML Tagsets do not support excel like functionalities. Therefore, user defined tagset TableEditor is used to design these functionalities. Conditional formatting is used to draw user attention to certain data points. The second tab is created to display the Legend that is the business logic behind the conditional formatting as shown in Figure 3.



RW | Legend

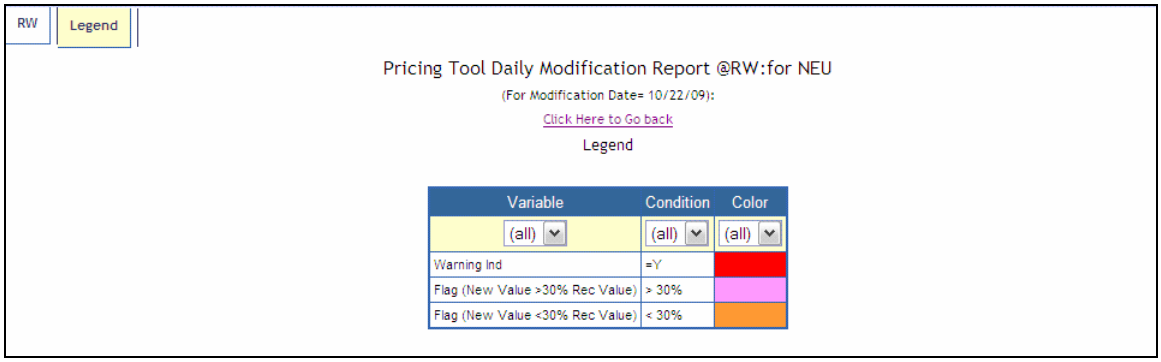
Pricing Tool Daily Modification Report @RW :for NEU

(For Modification Date= 10/22/09):

Click Here to Go back

| ResName | StYrWk | New Val | Rec Val | Variance | Flag | Lock | Override | PromoCode | Warning Ind |
|---|---|---|---|---|---|---|---|---|---|
| (all) | (all) | (all) | (all) | (all) | (all) | (all) | (all) | (all) | (all) |
| Resort 1234 | 201001 | 85 | 141 | -56 | < 30% | Y | Y | str | Y |
| Resort 1234 | 201001 | 85 | 141 | -56 | < 30% | Y | Y | str | Y |
| Resort 1234 | 201007 | 85 | 129 | -44 | < 30% | Y | Y | str | Y |
| Resort 1234 | 201007 | 85 | 129 | -44 | < 30% | Y | Y | str | Y |
| Resort 1234 | 201013 | 85 | 139 | -54 | < 30% | Y | Y | str | N |
| Resort 1234 | 201013 | 85 | 139 | -54 | < 30% | Y | Y | str | Y |
| Resort 1234 | 201102 | 350 | 150 | 200 | > 30% | Y | Y | str | Y |
| Resort 1234 | 201103 | 350 | 142 | 208 | > 30% | Y | Y | str | Y |
| Resort 1234 | 201104 | 350 | 147 | 203 | > 30% | Y | Y | str | Y |
| Resort 1234 | 201105 | 350 | 109 | 241 | > 30% | Y | Y | str | Y |
| Resort 1234 | 201106 | 350 | 129 | 221 | > 30% | Y | Y | str | Y |
| Resort 1234 | 201107 | 350 | 127 | 223 | > 30% | Y | Y | str | Y |
| Resort 1234 | 201108 | 350 | 147 | 203 | > 30% | Y | Y | str | Y |
| Resort 1234 | 201109 | 350 | 103 | 247 | > 30% | Y | Y | str | Y |
| Resort 1234 | 201110 | 350 | 210 | 140 | > 30% | Y | Y | str | Y |
| Resort 1234 | 201111 | 350 | 177 | 173 | > 30% | Y | Y | str | Y |

**Figure 2: Detailed Page**



RW | Legend

Pricing Tool Daily Modification Report @RW:for NEU

(For Modification Date= 10/22/09):

Click Here to Go back

Legend

| Variable | Condition | Color |
|---|---|---|
| (all) | (all) | (all) |
| Warning Ind | =Y | |
| Flag (New Value >30% Rec Value) | > 30% | |
| Flag (New Value <30% Rec Value) | < 30% | |

**Figure 3: Detailed Page (Legend)**

## TECHNICAL DETAILS

This section talks about technical details behind creating the Menu and Detail Pages. It also provides sample SAS code which can directly be utilized in similar conditions with little or no modifications.

### CREATING DYNAMIC MENU PAGE

As mentioned in the Proposed Solution section, because of the nature of this tool, static HTML reports with hardcoded links will not work. The menu page has to be dynamically created to handle changes in the number of territories and their names.

The description of a dynamic page design using proc format and advanced macro programming techniques is mentioned below:

Step 1: Create a SAS Data set destination1 which will hold all the information needed on the menu page.

```
proc sql;
    create table destination1 as
    select       Terr_id,
Terr_mod,
count(*) as no_mod
    from source1
    group by 1,2,3,4;
quit;
```

Step 2: Create another data set loop_link with link variable which will hold the links to individual territory pages. Create a new variable called linkfmt which will hold a user-defined data format for creating links to individual territory, year, and quarter combination.

```
%let webdavserver_addr = “SAS Web Server Address”;
data loop_link;
set destination;
link = "&webdavserver_addr _"||compress(Terr_mod)||".html";
linkfmt = COMPRESS(Terr_mod  ||'= '||'"'||link||'"');
run;
```

Step 3: Use Data Null statement to write .sas file on the SAS server. This code will have a macro (in this case it will be DURL) which will be basically creating a proc format. The content of this macro will vary each time the code is run based on the number of records in the source data.

```
Data _null_;
file "/user/ SASUSerID /sasuser.v91/DUrlfmt.sas" DROPOVER ;
set loop_link end=end;
if _n_ = 1 then put @1 '%Macro DURL;'/@1 'Proc Format;' /@2 'Value $urllink';
put @4 linkfmt;
if end then put @1 ';Run;' / @1 '%mend;';
run;
quit;
```

When the above code runs, there will be a SAS file created on the on SAS server named DurUrlfmt.sas with contents as follows:

```
%Macro DURL;
Proc Format;
Value $urllink
   ASI_SKU="&webdavserver_addr /daily_ASI_SKU.html"
   CUS_SKU="&webdavserver_addr /daily_CUS_SKU.html"
   MAY_SKU="&webdavserver_addr /daily_MAY_SKU.html"
   RWC_SKU="&webdavserver_addr /daily_RWC_SKU.html"
   SAM_SKU="&webdavserver_addr /daily_SAM_SKU.html"
   ASI_SW="&webdavserver_addr /daily_ASI_SW.html"
   CUS_SW="&webdavserver_addr /daily_CUS_SW.html"
   ASI_RW="&webdavserver_addr /daily_ASI_RW.html"
   CUS_RW="&webdavserver_addr /daily_CUS_RW.html"
   SAM_RW="&webdavserver_addr /daily_SAM_RW.html"
;Run;
%mend;
```

```
Step 4:  Write filename statement to load the DUrlfmt.sas macro and write data
_null_ statement to run the macro.

   filename programs "/user/ SASUSerID/sasuser.v91/ DURL.sas";
   %include programs;
   data _null_;
   %DURL;
   run;
```

Step 5: Proc tabulate and proc format are used to create the menu.html page and hyperlinks respectively. Additional information such as summarized data at destination level, date and time when the report was last updated is also added to the menu page.

```
   ods listing close;
   options mprint;
   %let _ODSOPTIONS=%STR(file="menu.html");
   %stpbegin;
   proc tabulate data =destination missing f=comma8. order = data
   style={ borderwidth =1 bordercolor = black cellheight=0mm background=white
   foreground=black just=c vjust=c font=('Arial',8pt)cellspacing=0 cellpadding=0
   };
   title1 h = 14pt "Pricing Tool Daily Modification Report " ;
   title2 h = 10pt " (For modification Date = &date.): " ;
   class terr_id Terr_mod  /style={borderwidth =1pt background=#436082
   foreground=white just=c  font=('Arial',10pt) cellspacing=0 cellpadding=0 };
   classlev terr_id;
   classlev Terr_mod    /style={ url = $urllink.};
   var mod;
   table terr_id = 'Terrirtory ID'* Terr_mod  = 'Territory ID-Modification Level
   ' ,
   mod ='# of Modifications'*sum = ''
   / misstext='0';
   run;
   quit; ods html close;
   %stpend;
```

Step 6: Last step in the process is sending an email to analysts informing them that the report ran successfully and they can check the report.

```
   filename outbox email 'reporting@rci.com';
   data _null_;
      file outbox
      to='reporting@rci.com'
      subject=" Pricing Tool - Current Modifications Report is Updated" ;
      put "Hi,";
      put /'From Job: rpt_daily.sas is complete.';
      put /'Pricing Tool - Modifications Report is Updated';
      put /'Thanks,' / 'Reporting Team';
   run;
```

## CREATING EXCEL LIKE DATA GRID

SAS ODS techniques are used for creating detailed pages and user defined tagset TableEditor is used to enhance functionality of the data grid. Tagset contains events such as system title header, data, System footer and others that are initiated in particular order based on a SAS procedure or data step. The TableEditor tagset inherits from HTML4 tagset that is used by HTML destination.

Though users can use this tagset for a variety of purposes; we used it to create a flexible data grid to enhance readability of reports. With the help of SAS help desk and after going through related research papers, few changes were made to the tagset to create the required reporting system and publish HTML files on SAS Portal.

At RCI, our biggest challenge in using this tagset was to publish HTML files on the WEB DAV server. Following is the regular code using stored process to publish an html packages to SAS WebDAV server:

```
%let _ODSSTYLE=seaside;
%let _PARENT_URL=;
%let _IF_EXISTS=updateany;
%let _HTTP_USER= userid;
%let _HTTP_PASSWORD= pwd;
%let _ODSDEST = HTML;
%let _GOPT_DEVICE = activex;
%let _ABSTRACT=Report Abstract;
%let _EXPIRATION_DATETIME=%sysfunc(inputn(20DEC2005:12:40:40.0,
datetime20.));
%let _RESULT=PACKAGE_TO_WEBDAV;
%let path = &webdavserver_addr/&destfolder;
%let _COLLECTION_URL= &path;
%let _ODSOPTIONS = %str(file = “filneme.html”);
%stpbegin;
          Proc report/proc print/ proc tabulate;
%stpend;
```

These are standard parameters which should be set up in order to publish HTML files on SAS WebDAV server. However, when you use TableEditor tagset, you have to assign parameters as follows:

Change `_ODSDEST = Tagsets.Tableeditor.`

Define tagset options in _ODSOPTIONS definition as follows:

```
%let _ODSOPTIONS=%STR(file="Filename.html"
options (
autofilter = "yes"
autofilter_width="4em"
filter_cols = “1,2,3,4,5”
load_msg="yes"
load_img="&path/ajax-loader.gif"
web_tabs='SW,Legend'));
```

## ADDING WEB TABS

The Web tab option in the tagset is used to produce different tabs, one for the ‘Report’ and one for the ‘Legend’ as shown in Figure 4. These web tabs look similar to worksheets in Microsoft Excel; these sheets are actually generated in Base SAS. Another use of the WEB Tabs option is to manage large reports just by splitting one report into different tabs. The following example illustrates how to create Web tabs with Web_TABS = option.

```
%let _ODSOPTIONS=%STR(file="Filename.html"
options (web_tabs='SW,Legend'));
 proc report data = test;
run;
proc report data = Legend;
run;
ods tagsets.tableeditor close;
```

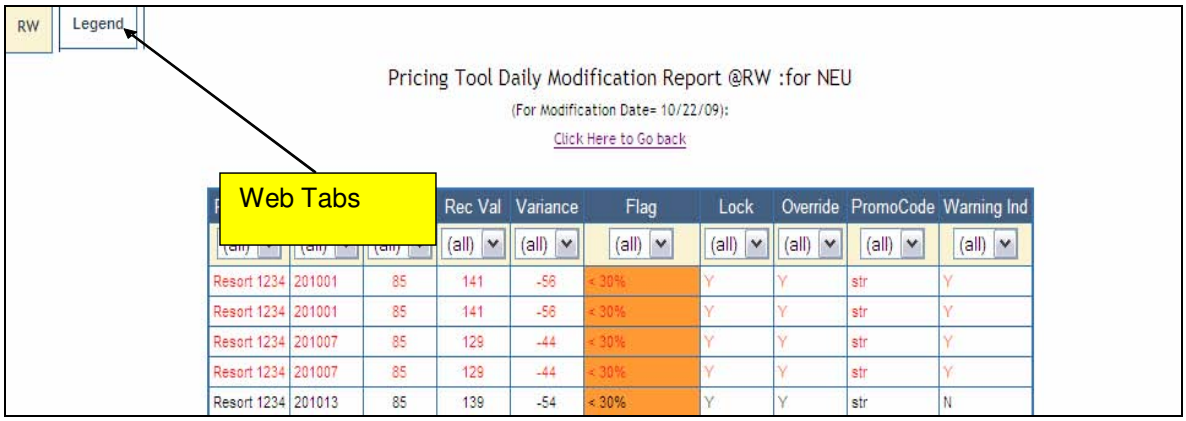



**Figure 4: Web Tabs**

## ADDING FILTERS

One of the biggest challenges in this project was the amount of data displayed on the one single HTML page. On average every page would have more than 1000 lines which will make the report lengthy and difficult to comprehend. Adding filters and conditional formatting made the reports easy to understand. The Autofilter option was used from the tagset to put in filters for selected columns as displayed in Figure 5. This allows users to drill down and look at a specific group of data. Now the user can drill down to a specific value or group of values without leaving the page. Here is an example of creating filtering effect:

AUTOFILTER = 'YES' this specifies that we want to apply autofilters.

AUTOFILTER_WIDTH = '4em' this option allows to add uniform width to the filters.

FILTER_COLS = "1,2,3,4,5,6" this options specifies where to apply autofilters.

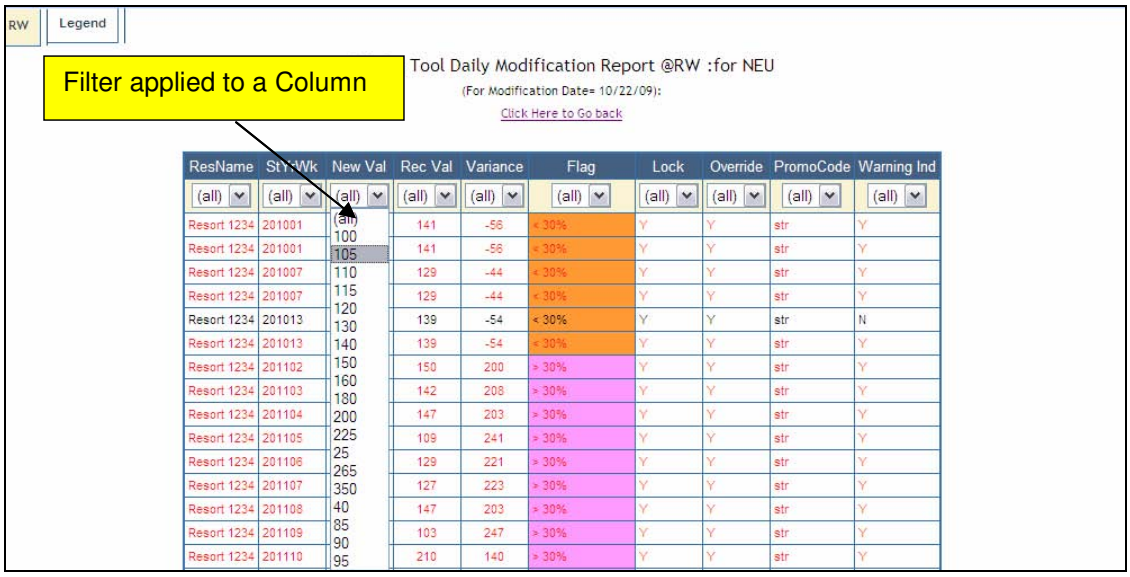



**Figure 5: Data Filters**

## ADDING SORT

Sorting capability is also added for selected columns as shown in Figure 6. The report can be dynamically sorted clicking on column headers. By specifying sort option several views of data can be obtained without leaving the page. Sort option can be applied to five different data types

1. String : used for character data

2. Number : used for integers and real numbers
3. NumberX: used for number that contain dollar sign or comma
4. Date: used with the Date9., MONNY and MMDDYY. Date formats.
5. None: prevents the column from being sorted.

Here is an example of creating sort option:

```
%let _ODSOPTIONS=%STR(file="Filename.html"
options Sort="yes"
Sort_Arrow_Color="brown"
Data_Type="String,String,Number,Number,Number,String,String,String,String,Str
ing");
 proc tabulate data = test;
run;
ods tagsets.tableeditor close;
```

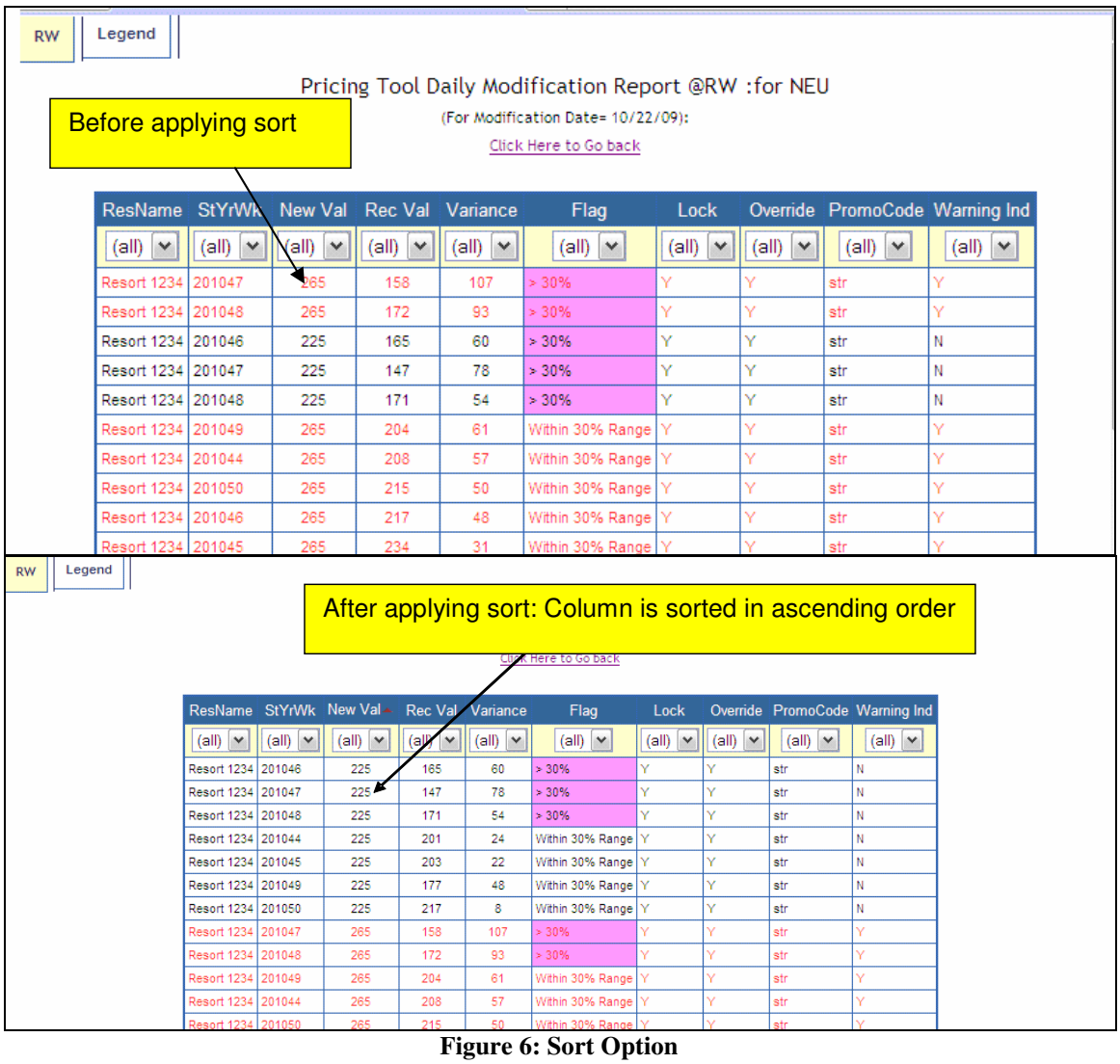



**Figure 6: Sort Option**

## ADDING CONDITIONAL FORMATTING

The use of conditional formatting allows the user to pin point problem areas at first glance of the report as shown in Figure 7. Below is the code that is used to create conditional formatting:

```
proc format;

    value $flg    '> 30%'= 'FF99FF'
                         '< 30%'= 'FF9933'
                         'Within 30% Range'= 'FFFFFF';
run;
```

After defining the formats in proc format apply the formats in proc report.

```
proc report data = DataSource  nowd colwidth = 20;
style (header)=  {background=#436082 foreground=white ont=('Arial',10pt)}
style (column)=  { background=white foreground=black  font=('Arial',8pt)}
style (summary)=  {background=#436082 foreground=white htmlclass="noFilter"
font=('Arial',8pt)}
box;
title1 h = 12pt "Pricing Tool Daily Modification Report @RW :for HPN" ;
title2 h = 8pt " (For Modification Date= &date.): " ;
title3 h = 8pt link = '&path/menu.html' "Click Here to Go back";
column RESORT_NAME Start_yrwk
          New_value Rec_value val_var val_flag
          lock OVERRIDE PROMO_CODE_IND WARNING_IND ;
define RESORT_NAME / 'ResName' left   ;
define lock / 'Lock' left   ;
define OVERRIDE / 'Override' left   ;
define PROMO_CODE_IND / 'PromoCode' left   ;
define WARNING_IND / 'Warning Ind' left   ;
define Effective_DATE / 'EffDate' left   ;
define Expiry_DATE / 'ExpDate' left   ;
define Analyst_first_name / 'Analyst' left   ;
define Start_yrwk  / 'StYrWk' left   ;
define New_value /'New Val' center analysis f =comma14.;
define Rec_value /'Rec Val' center analysis f =comma14.;
define val_var /'Variance' center analysis f =comma14.;
define val_flag /'flag' style(column)=[background=$flg.];
compute WARNING_IND ;
if _c10_ = 'Y' then
call define(_row_,'style','style={foreground=#FF1F1F}');
endcomp;
run;
```

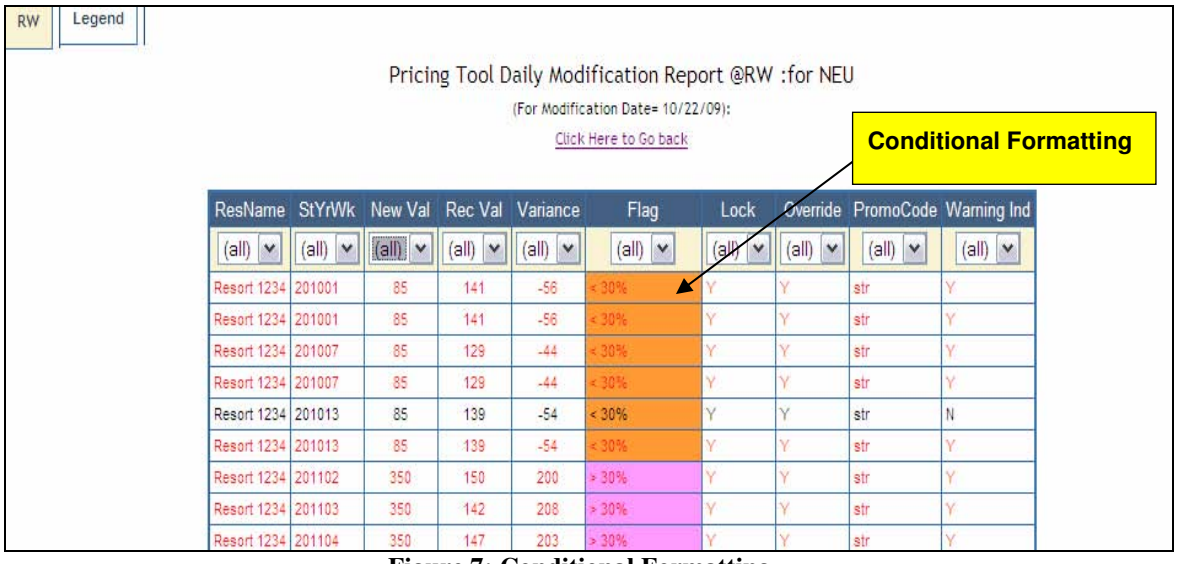



**Figure 7: Conditional Formatting**

### Limitations

We encountered some challenges with this method of creating the reporting tool. With our experience the file size for individual HTML page should be less than 2MB which comprises ~20,000 lines and ~ 22 columns in a report. If your file sizes are bigger than these it will take more than 2 minutes to load HTML page. These limitations are dependant on hardware specification of user's computer.

## CONCLUSION

The use of TableEditor tagset and SAS Macro programming techniques is a powerful way to create user friendly reports. We have come a long way in our reporting style with use of ODS techniques and user defined tagsets. Developing the Reporting System was well worth the effort because we can reuse these techniques to create more reports for various stake holders with less manual efforts.

## REFERENCES

http://www2.sas.com/proceedings/forum2008/258-2008.pdf

http://support.sas.com/rnd/papers/sgf07/sgf2007-report.pdf

http://dc-sug.org/how-report-style_new.pdf

## ACKNOWLEDGEMENTS

I would like to thank Chevell, Parker of SAS institute for creating this great tagset the TableEditor and helping me understand how to use it in different ways. I would also like to thanks Lily Duenas and Jeremy, Terbush from RCI for assisting me with this project and paper.

## CONTACT INFORMATION

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

Smita Sumant

Wyndham Exchange & Rentals

7 Sylvan Way

Parsippany, NJ 07054

Work Phone: (973) 753 - 6032

Email: Smita.sumant@rci.com

