SA118-2014

SAS ODS HTML + PROC Report = Fantastic Output

Girish K. Narayandas, OptumInsight, Eden Prairie, MN

ABSTRACT

ODS (Output Delivery System) is a wonderful feature in SAS to create consistent, presentable reports. Using this feature in combination with the powerful PROC Report can give you a lot of control over the report structure, design and overall appearance. You can set the column width, font style and size, add colors where desired, and include a picture or a logo, just to name a few. This paper contains tips and tricks for producing a presentation-worthy HTML output making use of some of the key features available in ODS HTML, PROC Report and SAS Macros.

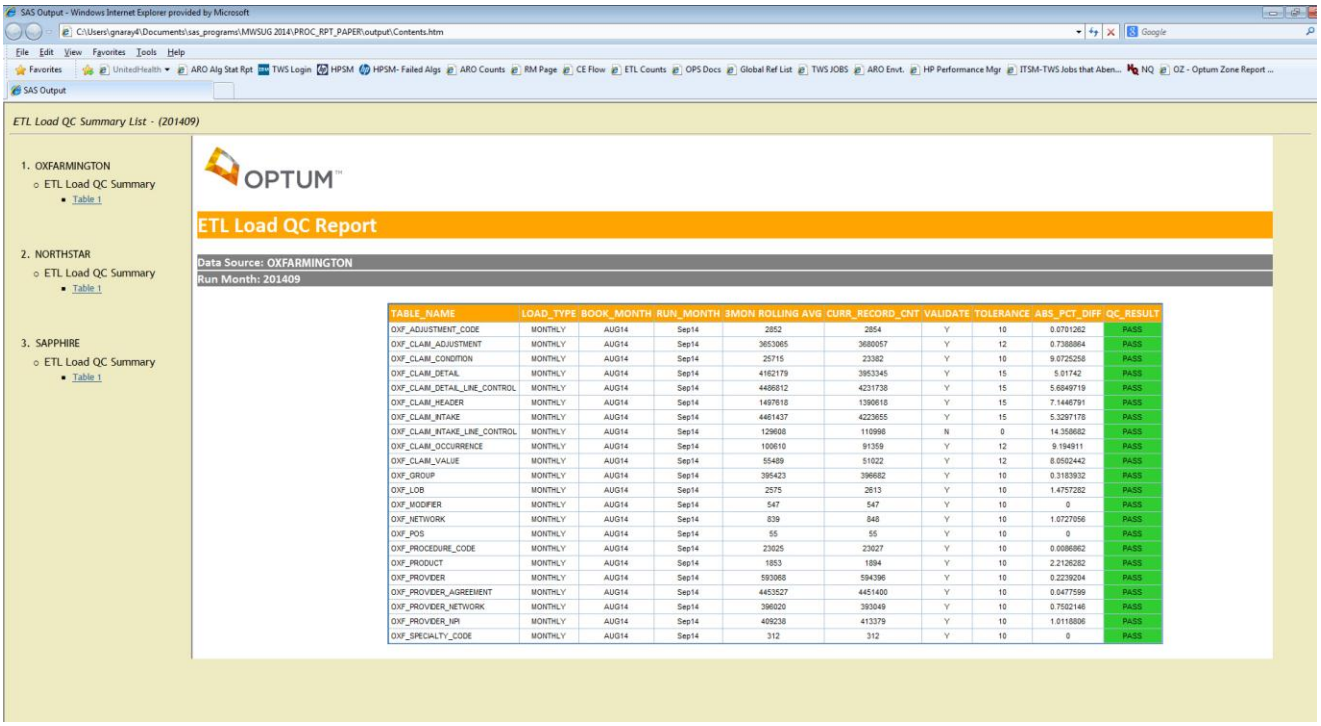
INTRODUCTION

Most companies strive to maintain uniformity and a certain level of consistency across the SAS reports they generate in order to align with corporate branding. Things such as the logo, colors, fonts, report structure etc., are some of the key details that define a corporate brand. In the data reporting world, it is common to see multiple developers/programmers creating reports separately. Sometimes these programmers are spread out across the country. In the case of global companies, they might even be located in different nations. The challenge is that management wants to see all of these individually developed reports rolled into one document. A unified document requires manual effort by developers or analysts to compile the individual outputs, and additional time and effort spent making each report look visually consistent with the corporate brand. This paper demonstrates how to automate the process by using pre-existing SAS tools and features. This paper also shows how to add additional functionality to a summary document such as point-and-click access into the granularity of each report, and a button to export data into Excel. Following this methodology, each time a new report is created it will append itself to the summary book and keep the compiled report current and consistent.

ODS HTML + PROC REPORT = FANTASTIC OUTPUT: AN EXAMPLE

Figure 1 is a screen shot of a data ETL (extraction transformation and loading process) quality check report. This report contains a content page and individual load quality control (QC) summaries. These summaries are available immediately after each data source loads. Because data comes from different sources at varying times throughout the month, we want to run a quality check each time a new data set is refreshed. Each individual report contains tables for the data source, the load type, whether table validation is required, and what the thresholds are for quality inspection. In this fictional example, a rolling three-month average of the data counts, including thresholds, is used to determine if new data is valid. New data's calculated validity is displayed with a value 'PASS' or 'FAIL' in the final column, QC_Result. For quick and easy readability, the 'compute' option in PROC REPORT is utilized to change the color of the cell to be green for pass and red for the tables that failed QC inspection.

Figure 1: ETL QC Report



TABLE_NAME	LOAD_TYPE	BOOK_MONTH	RUN_MONTH	3MON ROLLING AVG	CURR_RECORD_CNT	VALIDATE	TOLERANCE	ABS_PCT_DIFF	QC_RESULT
OXF_ADJUSTMENT_CODE	MONTHLY	AUG14	Sep14	2852	2854	Y	10	0.0701262	PASS
OXF_CLAIM_ADJUSTMENT	MONTHLY	AUG14	Sep14	3653065	3680057	Y	12	0.7388864	PASS
OXF_CLAIM_CONDITION	MONTHLY	AUG14	Sep14	25715	23382	Y	10	9.0725258	PASS
OXF_CLAIM_DETAIL	MONTHLY	AUG14	Sep14	4162179	3953345	Y	15	5.01742	PASS
OXF_CLAIM_DETAIL_LINE_CONTROL	MONTHLY	AUG14	Sep14	4486812	4231738	Y	15	5.6849719	PASS
OXF_CLAIM_HEADER	MONTHLY	AUG14	Sep14	1497618	1390618	Y	15	7.1446791	PASS
OXF_CLAIM_INTAKE	MONTHLY	AUG14	Sep14	4461437	4223655	Y	15	5.3297178	PASS
OXF_CLAIM_INTAKE_LINE_CONTROL	MONTHLY	AUG14	Sep14	129608	110998	N	0	14.358682	PASS
OXF_CLAIM_OCCURRENCE	MONTHLY	AUG14	Sep14	100610	91359	Y	12	9.194911	PASS
OXF_CLAIM_VALUE	MONTHLY	AUG14	Sep14	55489	51022	Y	12	8.0502442	PASS
OXF_GROUP	MONTHLY	AUG14	Sep14	395423	396682	Y	10	0.3183932	PASS
OXF_LOB	MONTHLY	AUG14	Sep14	2575	2613	Y	10	1.4757282	PASS
OXF_MODIFIER	MONTHLY	AUG14	Sep14	547	547	Y	10	0	PASS
OXF_NETWORK	MONTHLY	AUG14	Sep14	839	848	Y	10	1.0727056	PASS
OXF_POS	MONTHLY	AUG14	Sep14	55	55	Y	10	0	PASS
OXF_PROCEDURE_CODE	MONTHLY	AUG14	Sep14	23025	23027	Y	10	0.0086862	PASS
OXF_PRODUCT	MONTHLY	AUG14	Sep14	1853	1894	Y	10	2.2126282	PASS
OXF_PROVIDER	MONTHLY	AUG14	Sep14	593068	594396	Y	10	0.2239204	PASS
OXF_PROVIDER_AGREEMENT	MONTHLY	AUG14	Sep14	4453527	4451400	Y	10	0.0477599	PASS
OXF_PROVIDER_NETWORK	MONTHLY	AUG14	Sep14	396020	393049	Y	10	0.7502146	PASS
OXF_PROVIDER_NPI	MONTHLY	AUG14	Sep14	409238	413379	Y	10	1.0118806	PASS
OXF_SPECIALTY_CODE	MONTHLY	AUG14	Sep14	312	312	Y	10	0	PASS

STEPS INVOLVED

Here are the nine steps I used to create the report illustrated in Figure 1 above. Each step is explained in greater detail in the subsequent sections.

```
/*-------------------------------------------------------------
 STEP-1: Create Summary Macro
-------------------------------------------------------------*/
%macro generate_summary ;

/*-------------------------------------------------------------
 STEP-2: Check for necessary directories, If they don't exist,
 Create required directories, assign Library names etc.
-------------------------------------------------------------*/
%if %sysfunc (fileexist(load_qc_&run_month.)) ^= 1 %then...

/*-------------------------------------------------------------
 STEP-3: USE PROC TEMPLATE to Customize Content Title
-------------------------------------------------------------*/
Proc template;
...

/*-------------------------------------------------------------
 STEP-4: Initiate the HTML Report
-------------------------------------------------------------*/
ODS HTML ...;
```

```
/*-------------------------------------------------------------
STEP-5: DRIVE macro to list all the Final Datasets
ready for reporting
-------------------------------------------------------------*/
%MACRO DRIVE(dir);
    ...
/*-------------------------------------------------------------
 STEP-6: Create any additional report components that are
     needed, store them into Macro Variables
-------------------------------------------------------------*/
 PROC SQL noprint; select distinct INTO: ...

/*-------------------------------------------------------------
 STEP-7: Output HTML File for each dataset in the output dir
-------------------------------------------------------------*/
ODS HTML FILE =...

/*-------------------------------------------------------------
 STEP-8: PROC Report to stylize the output report
-------------------------------------------------------------*/
PROC REPORT DATA =...

/*-------------------------------------------------------------
 STEP-9: E-Mail Component
-------------------------------------------------------------*/
DATA _NULL_;
   FILE mymail; ...
```

1. **STEP-1: CREATE A SUMMARY REPORT MACRO**

   The generate_summary macro is central to the automation process. It consists of using a PROC TEMPLATE step, PROC REPORT step, ODS Statements, creating and executing other macros, email components etc. Depending on the situation, the macro can be utilized to meet the reporting needs of a single team with multiple users, or could be enhanced further to handle reports generated by multiple teams across a division or multiple projects in a company.

```
/*-------------------------------------------------------------
 STEP-1: Create Summary Macro
-------------------------------------------------------------*/
%MACRO generate_summary ;
...
%MEND generate_summary ;
```

2. STEP-2: CREATE NECESSARY DIRECTORIES AND LIBRARY NAMES

The purpose of this step is to organize the directories by each run month. We use the 'sysfunc(fileexist)' function to determine if the necessary project directory structure is already set up or if it needs to be created. Because multiple members across a team can execute the summary macro, and each teammate uses this macro in a new run month (say, 201410), all the required directories wouldn't be in place. So, we create the needed folders and assign library names as shown.This step is an optional step, customized for the multiple person scenario.

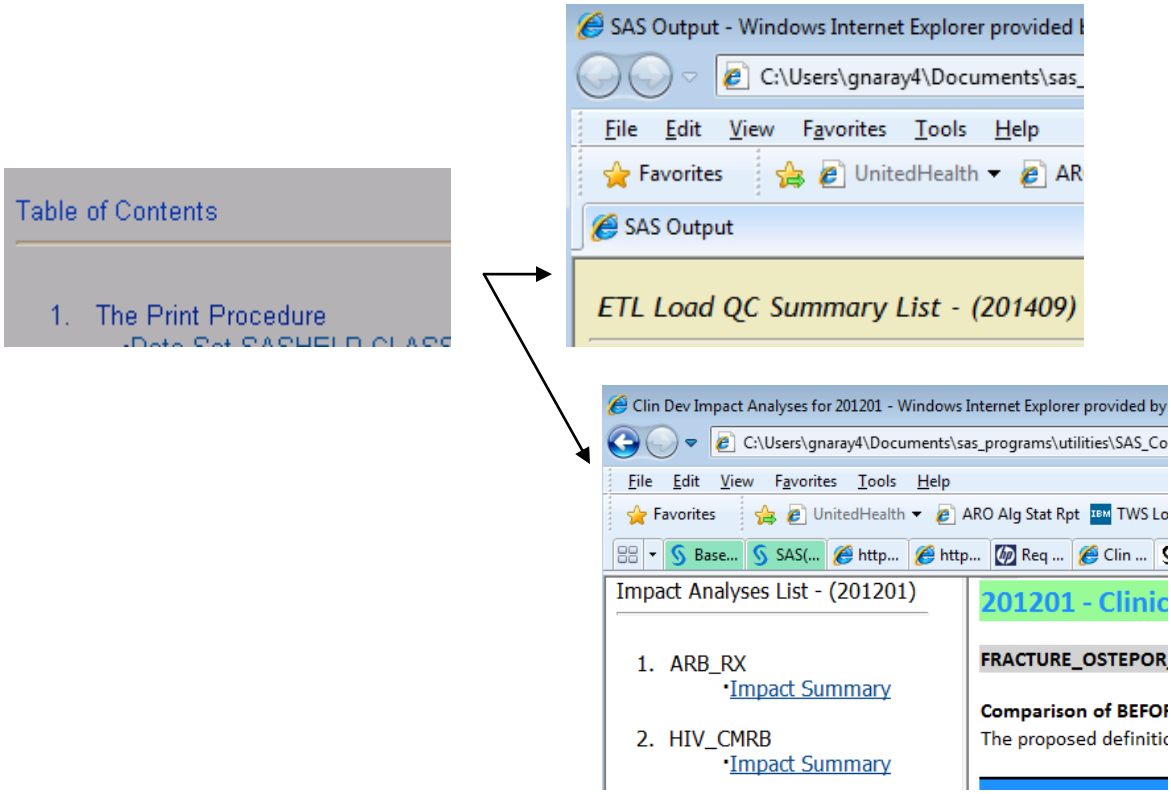
```
/*-------------------------------------------------------
 STEP-2: Check for necessary directories, If they don't exist,
 Create required directories, assign Library names etc.
-------------------------------------------------------*/
%if %sysfunc (fileexist(load_qc_&run_month.)) ^= 1 %then %do
  systask command "mkdir -p &loc./&run_month./qc_hub" wait;
  systask command "mkdir -p &loc./&run_month./qc_hub/data" wait;
  systask command "mkdir -p &loc./&run_month./qc_hub/out" wait;
%end;

libname in_data "&loc./&run_month./load_qc_hub/data";
libname output  "&loc./&run_month./load_qc_hub/out";
```

3. STEP-3: PROC TEMPLATE PROCEDURE

I used PROC TEMPLATE procedure to customize the "Table of Contents": text that is the default; the result is shown in Figure 2. There are several ready-made styles to choose from. I changed the default style to 'seaside' to suit my needs. However, a deeper dive into PROC TEMPLATE can provide you with an even greater level of control over the general style, font, text, border and table attributes. For more information on editing styles please refer to '*SAS 9.2 Output Delivery System User's Guide, SAS Documentation*' available at http://support.sas.com/documentation/cdl/en/odsug/61723/PDF/default/odsug.pdf.

```
/*-------------------------------------------------------
 STEP-3: USE PROC TEMPLATE to Customize Content Title
-------------------------------------------------------*/
PROC TEMPLATE;
  define style styles.test;
  parent=styles.seaside;
  style contenttitle from contenttitle /
  pretext = "ETL Load QC Summary List - (&run_month.)";
  end;
run;
```

Figure 2: Customization of “Table of Contents”



4. STEP-4: INITIATE ODS HTML

The **ODS HTML** statement opens or closes the HTML destination. The **PATH**= option specifies the location of the HTML files. **FRAME** identifies the file that integrates the table of contents, the page contents, and the body file. **‘CONTENTS’** opens the HTML destination and creates the file that contains a table of contents for the output. ODS HTML automatically creates an **ANCHOR** for every piece of output generated by the SAS procedures. An anchor specifies a particular location within an HTML file. In order for the links from the contents, page, or frame file to work, each piece of output in the body files must have a unique anchor to link to. For **STYLE**, we are choosing modified seaside style names as ‘test’ created in the PROC TEMPLATE step described above. ODS HTML syntax has improved. See ‘*ODS HTML Evolution: HTML that Scrolls, Panels, Floats, Reads, and Integrates*’ by Eric Gebhart (Reference #3) for greater details.

```
/*------------------------------------------------------------
 STEP-4: Initiate the HTML Report
------------------------------------------------------------*/
ODS HTML frame = "Frame_ETL_Load_QC.html"
         (title="ETL Load QC Summary Report for &run_month.")
  contents = "Contents.html"
  anchor = "data_source"
  path = "&loc./&run_month./summary_rpt/" (url=none)
  style = test;
```

5. STEP-5: MACRO TO PRINT ALL DATASETS OF A DIRECTORY

This macro is a ready-to-use portable macro available on the SAS support website, http://support.sas.com/kb/25/074.html. It enables you to print all files in a folder with a specified extension. This macro takes 2 parameters. The first parameter is the directory that contains the files. The second parameter is the file extension of the files you are looking for. I made a few modifications to this macro, included in steps 6, 7 and 8 described below to meet my report needs.

```
/*------------------------------------------------------------
STEP-5: DRIVE macro to list all the Final Datasets
ready for reporting
------------------------------------------------------------*/
%MACRO drive(dir,ext);
  %let filrf=mydir;

  /* Assigns the fileref of mydir to the directory and opens the
directory */
  %let rc=%sysfunc(filename(filrf,&dir));
  %let did=%sysfunc(dopen(&filrf));

 /* Returns the number of members in the directory */
  %let memcnt=%sysfunc(dnum(&did));

 /* Loops through entire directory */
   %do i = 1 %to &memcnt;

 /* Returns the extension from each file */
     %let name=%qscan(%qsysfunc(dread(&did,&i)),-1,.);

 /* Checks to see if file contains an extension */
     %if %qupcase(%qsysfunc(dread(&did,&i))) ne %qupcase(&name)
%then %do;

 /* Checks to see if the extension matches the parameter value */
 /* If condition is true prints the full name to the log       */
      %if (%superq(ext) ne and %qupcase(&name) = %qupcase(&ext))
or
          (%superq(ext) = and %superq(name) ne) %then %do;
         %put %qsysfunc(dread(&did,&i));
       %end;
     %end;
   %end;

 /* Closes the directory */
  %let rc=%sysfunc(dclose(&did));

%MEND drive;

/* First parm is the dir of where your files are stored.     */
/* Second parm is the extension you are looking for.         */
```

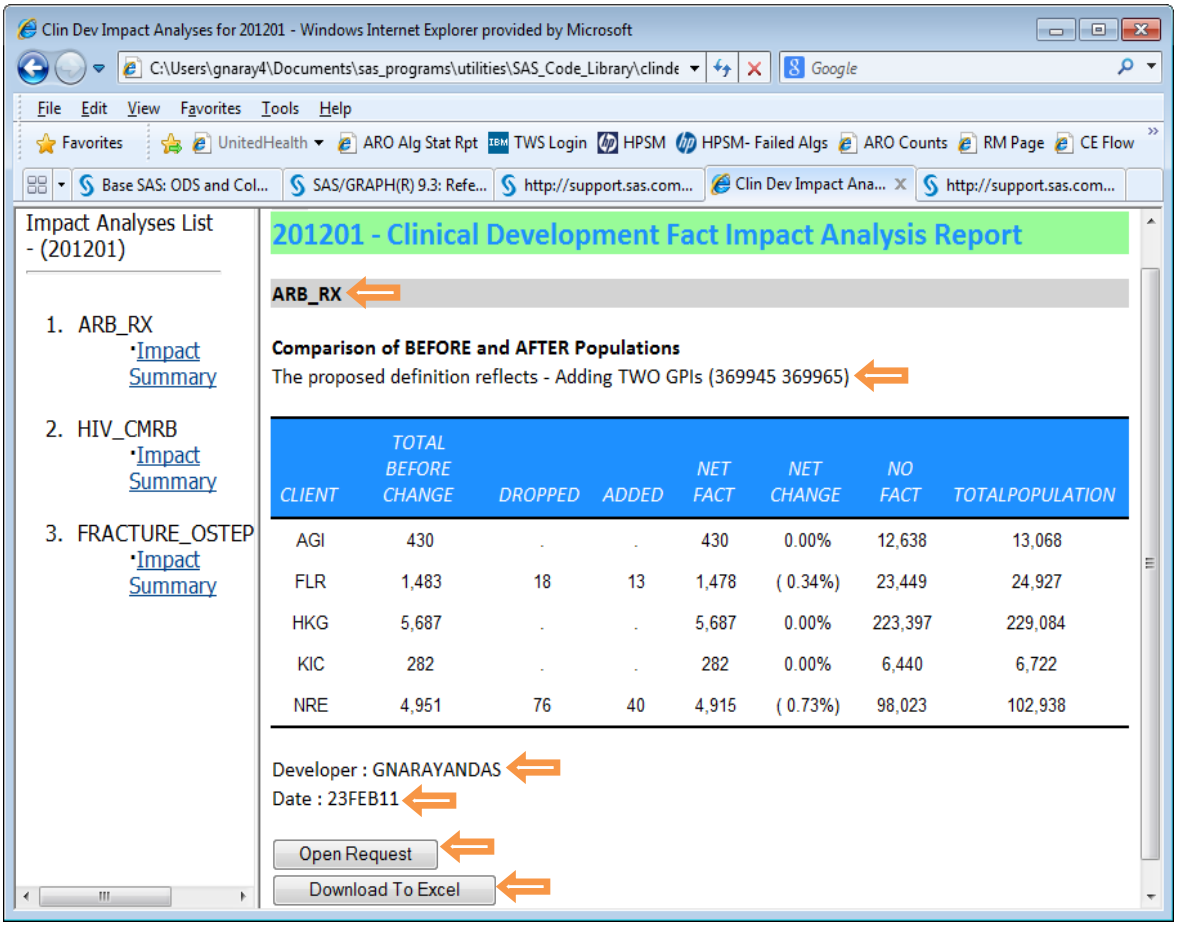
```
/* Leave 2nd parm blank if you want a list of all the files. */

%drive(c:\,sas)
```

6. STEP-6: BUILD ADDITIONAL REPORT COMPONENTS AS NEEDED

While the summary report is great, sometimes it is useful to have access to other details. For example you can look up who ran the report, or open the work request pertaining to the report. We can include such details by creating macro parameters when the jobs are executed and use them in the report as clickable buttons, shown in Figure 3 below.

```
/*------------------------------------------------------------
 STEP-6: Create any additional report components that are
     needed, store them into Macro Variables
------------------------------------------------------------*/
proc sql noprint;
      select distinct compress(fact, " ") into: factname
        from &librf..&fname. ;
      select distinct impact_definition into: impact_def
        from &librf..&fname. ;
      select distinct (developer) into: developer
        from &librf..&fname. ;
      select distinct (rundate) into: rundate
        from &librf..&fname. ;
      select distinct (fact_request) into: fact_request
        from &librf..&fname. ;

quit;

footnote4 J=l  "<div align=""left""><input type=""button""
value=""Open Request""
onclick=javascript:window.open('&fact_request')></div>";
footnote5 j=l '<input
onclick="document.execCommand(''SAVEAS'',true,
''c:\\temp\\test.xls'')" value= "Download To Excel"
type="button">';
```

Figure 3: Links on a Report



7. STEP-7: WRITE HTML OUTPUT FILE FOR EACH DATASET OR PROCEDURE

The **FILE**= option identifies the file containing the HTML version of the procedure output. Use the ODS PROCLABEL statement to create customized labels in the table of contents. Since **PROCLABEL** affects only the SAS procedure immediately following it, it works great in this situation, since we need different labels to represent different data sources each time PROC REPORT iterates.

```
/*-------------------------------------------------------------
 STEP-7: Output HTML File for each dataset in the output dir
-------------------------------------------------------------*/
ods html file =
"&loc./&run_month./summary_rpt/ETL_Load_QC_&data_source..html"
anchor = "data_source"  ;
ods proclabel "&data_source.";
```

8. STEP-8: PROC REPORT TO STYLIZE THE OUTPUT

There is a quite bit of material already available on PROC REPORT procedure if you want to understand all the intricate details, features and choices it has to offer. The intention of this paper is to illustrate just a few of these options, combined with ODS HTML and SAS utilities to develop a refined report. Below is the PROC REPORT statement that I used to make the output visually more appealing. I described the process in six parts, A-F, below.

```
/*-------------------------------------------------------------
 STEP-8: PROC Report to stylize the output report
-------------------------------------------------------------*/
proc report data = &librf..&fname.                                    A
            split='*'
            contents = "ETL Load QC Summary"
  style(header) =
    [just = left
     font face = calibri
     font_size = 3                                                    B
     foreground = white
     background = orange
     protectspecialchars=off]  ;

  column  Table_name  Load_Type Book_Month  Run_Month                  C
          AVG_Count_3MON CURR_Record_CNT VALIDATE TOLERANCE
          ABS_PCT_DIFF QC_RESULT ;

  define Table_name/display 'TABLE_NAME' style={just=l vjust =b };
  define Load_Type / display 'LOAD_TYPE' style={just=c vjust =b };
  define Book_Month/ display 'BOOK_MONTH'style={just=c vjust =b };
  define Run_Month/ display 'RUN_MONTH' style={just=c vjust =b };
  define AVG_Count_3MON / display '3MON ROLLING AVG'style={just=c
vjust =b };
  define CURR_Record_CNT/ display 'CURR_RECORD_CNT' style={just=c    D
vjust =b };
  define VALIDATE/ display 'VALIDATE'  style={just=c vjust =b };
  define TOLERANCE / display 'TOLERANCE' style={just=c vjust =b };
  define ABS_PCT_DIFF /  display 'ABS_PCT_DIFF'style = {just=c
vjust =b };
  define QC_RESULT / display 'QC_RESULT' style={just=c vjust =b };


  compute qc_result;
     if qc_result = 'PASS' then
       call define(_col_,"style","style={background=limegreen}");
      else if qc_result = 'FAIL' then                                 E
         call define(_col_,"style","style={background=crimson}");
     else call define(_col_,"style","style={background=crimson}");
  endcomp;
```

```
 TITLE1 j=l '<IMG SRC="/location/OPTUM.gif" HEIGHT=80 >';

 TITLE3 j=l c=white bcolor = orange font=calibri h=6 bold "ETL Load
QC Report";

 TITLE5 j=l c=White bcolor = grey font=calibri h=4 bold "Data
Source: &Data_source";

 TITLE6 j=l c=White bcolor = grey font=calibri h=4 bold "Run Month:
&run_month";

footnote J=l  "<div align=""left""><input type=""button""
value=""Open Request""

QUIT;
```

F

A. PROC REPORT has a regular DATA statement. SPLIT breaks a column heading when it reaches that character and continues the heading on the next line. The split character (*) itself is not part of the column heading although each occurrence of the split character counts toward the 256-character maximum for a label. 'CONTENTS' specifies text for the HTML or PDF table of contents entry for the output.

B. STYLE: option, or rather options, in the PROC REPORT statement can add a great amount of detail to the report. For example, we have listed six options in that statement. We are stating that the header be LEFT JUSTIFIED, font style = Calibri, font size =3, controlling background and foreground colors. When sending output to the HTML destination using PROC REPORT a number of the usual attributes are not available. The issue arises when you type something like '<MCA />' into your data then SAS will protect those characters and output will actually be converted into character entities that don't make sense. A simple fix for such situations is to turn off the 'protectspecialchars' attribute within style definitions, so SAS ignores special characters and output is exactly as entered.

C. COLUMN: The primary function of the COLUMN statement is to provide a list of variables for REPORT to print in the order from left to right that they are to appear on the report. Though not demonstrated in this code, the COLUMN statement can be used to attach a statistic to the variable. It can also add headers to group variables.

D. DEFINE: While COLUMN statement is used to declare the variables of interest, DEFINE statements provide the details of how to use those variables. In this particular report, we are using it for DISPLAY of the variable name. Additionally, other STYLE attributes such as horizontal (left or right or center) and vertical (top or bottom or center) justification could be chosen.

E. COMPUTE: A compute block can be associated with a report item or with a location (at the top/bottom of a report/page). A compute block is created with the COMPUTE window or with the COMPUTE statement. There is enormous flexibility with the

COMPUTE statement. In our example, we are simply using a 'compute' statement to change colors of the cell based on the QC_RESULT value of a data source table.

F. TITLE and FOOTNOTES: statements can be defined anywhere in the SAS program

9. Step-9: EMAIL COMPONENT:

This is a straight forward SAS utility that most SAS users may are already be aware of. Several email compositions could be written and used with appropriate conditional logic if a different email composition is needed to be sent to different team members based on the quality check result.

```
/*------------------------------------------------------------
 STEP-9: E-Mail Component
------------------------------------------------------------*/
filename mymail email to="userid@optum.com"
     subject="ETL Load QC Summary Report - &run_month." ;

data _null_;
  file mymail;
  put "Hi %upcase(&&sysuserid)," /;
  put "The ETL Load QC Summary Report for the month of
&run_month. Completed processing."/;
  put "Please review the Load QC Summary Results at the link
below"/;
  put "File://X:&run_month./output/Frame_ETL_QC_Summary.htm" /;
  put "Thank you!"/;
  put " ";
  run;
```

CONCLUSION

PROC REPORT is a very powerful report generating tool that is fairly easy to use once you get familiar with it. Initially, the key words and different syntax style may seem complex, but it becomes more manageable and even enjoyable as you become familiar with the flexibility and choices available to spruce up your reports. By combining PROC REPORT with ODS HTML, SAS MACROS and other SAS utilities, you can create impressive and versatile reports for your company.

REFERENCES

1. Carpenter, Art (2012). Carpenter's Guide to Innovative SAS Techniques textbook (Chapter 8.4, Using PROC REPORT to Better Advantage, pages 280 -290)
2. Carpenter, Arther, L. (2012). PROC REPORT Basics: Getting Started with the Primary Statements
3. Gebhart, Eric (2010). ODS HTML Evolution: HTML that Scrolls, SAS Institute Inc., Cary, NC

4. Gebhart, Eric. (2009). Inline Formatting with ODS Markup. Proceedings of the SAS Global Forum 2009 Conference. support.sas.com/resources/papers/proceedings09/222-2009.pdf.
5. Olinger, Chris (2000) ODS For Dummies, Proceedings of SUGI 25 (April 9-12, 2000)
6. Concepts for ODS - Colors Concepts, SAS Support Documentation Notes http://support.sas.com/rnd/base/ods/templateFAQ/Template_colors.html
7. SAS 9.2 Output Delivery System User's Guide, SAS Documentation' available at http://support.sas.com/documentation/cdl/en/odsug/61723/PDF/default/odsug.pdf.
8. Listing all files that are located in a specific directory, SAS Support Documentation, Sample 25074: http://support.sas.com/kb/25/074.html
9. PROC REPORT Statement – SAS Documentation, http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a002473620.htm
10. Wright, Wendi L. (2000), ODS Step by Step, Educational Testing Service, Princeton, NJ, proceedings of NorthEast SAS Users Group (NESUG)

TRADEMARK CITATIONS

CONTACT INFORMATION

Your comments and questions are valued and encouraged.
Contact the author at:
Girish Narayandas
Optum
13625 Technology Drive
Eden Prairie, MN 55344
Email: girish.narayandas@optum.com
www.optum.com