Paper 3261-2015

# Creating a Data Quality Scorecard

Tom Purvis, Qualex Consulting Services, Inc.

Clive Pearson, Qualex Consulting Services, Inc.

## ABSTRACT

The approach in this paper shows users how to develop a data quality scorecard based on rules, and judged against a set of standards set by the user.

Each rule (a/k/a metric) has a **standard** that determines whether it passes, fails, or needs review (a green, red, or blue score). A rule can be as simple as: Is the value for this column missing?, or is this column within a valid range? Further, it includes comparing a column to one or more other columns, or checking for specific invalid entries. It also includes rules that compare a column value to a lookup table to determine whether the value is in the lookup (a/k/a reference) table.

Users can create their own rules and each column can have any number of rules. For example, a rule can be created to compare a dollar column to a range of acceptable values. The user can determine that it is expected that up to two percent of the values are allowed to be out of range. If two to five percent of the values are out of range, then data should be reviewed. And, if over five percent of the values are out of range, the data is not acceptable.

The entire table has a color-coded scorecard showing each rule and its score. Summary reports show columns by score and distributions of key columns. The scorecard enables the user to quickly assess whether the SAS data set is acceptable, or whether specific columns need to be reviewed.

Drill-down reports enable the user to drill into the data to examine why the column scored as it did. Based on the scores, the data set can be accepted or rejected, and the user will know where and why the data set failed.

The approach can store each scorecard data in a data mart. This data mart enables the user to review the quality of their data over time. It can answer questions such as: is the quality of the data improving overall? Are there specific columns that are improving or declining over time? What can we do to improve the quality of our data?

## INTRODUCTION

Many users would like to check the quality of data after the data integration process has loaded the data into a data set or table. SAS has many products that automate the data integration and/or data quality process. However, if the user does not own these products, or the process does not warrant this level of automation, then a Base SAS solution might be more appropriate. This paper describes such an approach.

This scorecard is not intended to replace the quality control of the data integration or ETL process. It is a supplement to the ETL process. The programs are written using only Base SAS® and Output Delivery System (ODS), macro variables, and formats. This paper shows how to: (1) use ODS HTML; (2) color code cells with the use of formats; (3) use formats as lookup tables; (4) use INCLUDE statements to make use of template code snippets to simplify programming; and (5) use hyperlinks to launch stored processes from the scorecard.

## A FLOWCHART OF THE PROCESS

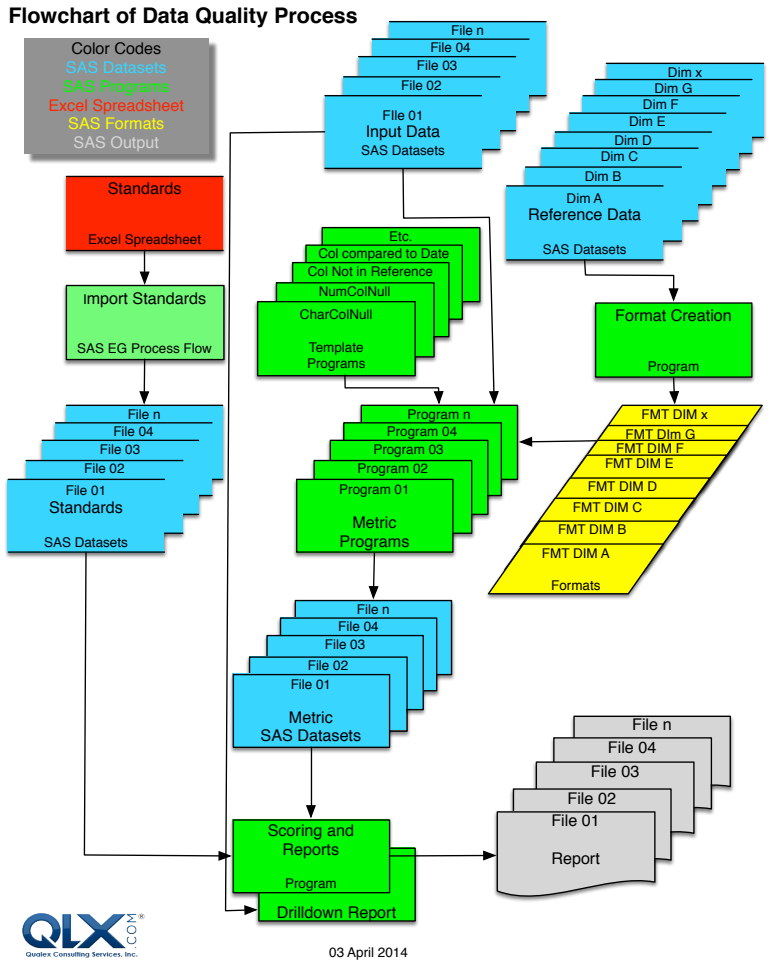



**Figure 1. Flowchart of the Scorecard Process**

## PROCESS DESCRIPTION

### RULES SPREADSHEET:

As described in the following pages, this data quality check process begins with a spreadsheet of rules (a tab is created for each file being reviewed – in the flowchart these files are shown as File01 .. File n). There is no limit to the number of file types included. Each rule represents one check performed on one column. Each rule is executed on each row of data.

Examples of common rules include:

- Is the column value blank?
- Is the date column value (e.g., birthdate) after today?
- Is the value in the column found in a reference table? (e.g., is postal code valid? Is this code in the dimension table?)
- Is the column value blank but other column is not blank? (e.g., a customer ID is present but customer name is blank)
- Is column A greater than Column B? (any column with any comparison to any other column)

Each rule is assigned a metric ID to uniquely identify the rule. (For this paper "metric" and "rule" are synonyms).

Each rule is also assigned a set of percentages to score the rule into three levels (categories): Green, Blue, and Red. (The user can change the colors).

For example, if rows for column A are blank for 0 to 10 percent of all records then this rule is scored as "Green". If the rule scores greater than 10 percent but less than 20 percent, then the rule is scored as "Blue". If the rule is scored over 20 percent with blank values then the rule is scored as "Red".

These thresholds are set by the user and can be changed at any time. Threshold levels can be set so that only two levels apply.

Within the spreadsheet each rule is defined and the ranges are set for each level (color). There is no limit to the total number of rules, and a column can have multiple rules, as long as each rule is a separate row in the spreadsheet.

Table 1 is a sample standards spreadsheet:

| Data_Field_Name | Include_In_Report | Metric_ID | Metric_Label | Green_Beg | Green_End | Blue_End | Red_end |
|---|---|---|---|---|---|---|---|
| Column_01 | Yes | Rule 001 | XXX ID is blank but XXX Name is not blank | 0.00 | 10.00 | 20.00 | 100.00 |
| Column_01 | Yes | Rule 002 | Blank Values in Column | 0.00 | 10.00 | 20.00 | 100.00 |
| Column_03 | Yes | Rule 003 | Blank Values in Column | 0.00 | 10.00 | 20.00 | 100.00 |
| Column_07 | Yes | Rule 004 | Blank Values in Column | 0.00 | 10.00 | 20.00 | 100.00 |
| Column_41 | Yes | Rule 005 | Blank Values in Column | 0.00 | 10.00 | 20.00 | 100.00 |
| Column_42 | Yes | Rule 006 | Other XXX ID is blank but Other XXX Name is not blank | 0.00 | 0.00 | 0.00 | 100.00 |
| Column_42 | Yes | Rule 007 | Blank Values in Column | 0.00 | 10.00 | 20.00 | 100.00 |
| Column_44 | Yes | Rule 008 | Blank Values in Column | 0.00 | 10.00 | 20.00 | 100.00 |
| Column_49 | Yes | Rule 009 | ZZZ_ID is blank but ZZZ_Name is not blank | 0.00 | 0.00 | 0.00 | 100.00 |
| Column_49 | Yes | Rule 010 | Blank Values in Column | 0.00 | 10.00 | 20.00 | 100.00 |
| Column_51 | Yes | Rule 011 | Blank Values in Column | 0.00 | 10.00 | 20.00 | 100.00 |
| Column_55 | Yes | Rule 012 | Blank Values in Column | 0.00 | 10.00 | 20.00 | 100.00 |
| Column_55 | Yes | Rule 013 | PTermination_Date GT Rundate | 0.00 | 0.00 | 0.00 | 100.00 |
| Column_45 | Yes | Rule 014 | Effective_date GT Rundate | 0.00 | 0.00 | 0.00 | 100.00 |
| Column_45 | Yes | Rule 015 | Blank Values in Column | 0.00 | 10.00 | 20.00 | 100.00 |
| Column_46 | Yes | Rule 016 | XXXX_Effective_Date is Prior to Termination date | 100.00 | 100.00 | 100.00 | 0.00 |

**Table 1. Standards Spreadsheet**

This spreadsheet defines each rule, its column, and its thresholds. One tab in the spreadsheet is used for each file type being checked.

The spreadsheet resides in any directory that the user chooses. A simple SAS Enterprise Guide (SAS EG) project imports each tab of the spreadsheet into a permanent SAS dataset placed in the data path. The path in the scoring program can be changed to point to any number of scoring datasets if the user wants to use different scores or rules for different sources.

## METRIC PROGRAMS:

Each input data file type has a metric program that consists of a series of rule (metric) templates. A template is small set of program lines that use SAS macro variables to make the template specific to a rule defined in the spreadsheet. One template represents one type of rule (e.g., does the column has blank values?). A template can be used over and over again for whatever rules apply. There is no limit to the number of templates.

Sample of Metric Program: Highlighted lines show the calls to the template programs.

```
data work.metrics  ;
length metric_id $ 16 rowfailed rows_processed 8 ;
set  sasin.&inputname. end=end ;
keep metric_id rowfailed rows_processed ;
rows_processed + 1 ;

/******** Rule XXXX  **********/
%let metric_id = RuleXXXXX ;
```

```
%let col       = Column_ZZZZ;
%include "&path.\template_name.sas" / source2
… A section of code for each rule.
```

For checks against reference tables, a SAS format is used for the specific case of a rule matching a column in the file to a reference table. Each reference table is read into a temporary SAS dataset and then this data is used to create a SAS format. A rule is created and a template is added to the metric program to check the value in the file to the format. If the value is not in the format then the row is counted as “failed”.

For example, a reference file might contain all valid postal codes. A SAS format is created for these valid postal codes. Each value for the column “postal code” in the file is compared to the SAS format and those rows that do not match the format are counted as “failed”. The percentage of failed rows to all rows in the file is compared to the thresholds set in the spreadsheet to score the rule as Green, Blue, or Red. There is no limit to the number of reference files used (but using a lot of formats may fill up memory).

Reference files are converted to SAS formats by a simple program that reads the data and creates a SAS format. Each reference file creates one unique format and the format is named to be a mnemonic of the reference file.

Sample Template file for calling a reference file via a SAS Format:

```
metric_id  = "&metric_id." ;
if strip(&col.) NE " " and
   strip(put(&col.,&fmttest.)) = strip(&col.) then rowfailed = 1 ;
else                                          rowfailed = 0 ;
output ;

%let metric_id = ; /* these lines reset the macro variables so they will not persist. */
%let col       = ;     /* these lines reset the macro variables so they will not persist. */
%let fmttest   = ;   /* these lines reset the macro variables so they will not persist. */
```

Below is sample code to create a reference format:

```
libname fmts  "/sasdata/data/" ;  /* This is the path where data is stored /*
options fmtsearch=(fmts) ; /* This option ensures the format library is searched by SAS */
%let type    = C  ;            /* This creates a Character Format */
%let destin  = fmtZIP ;        /* The name of the format is fmtZIP */
%let source  = zipcodes ;  /* The source of the reference table is zipcodes */
%let length  = 5 ;

data work.&source. ;
set  fmts.&source. ;

start = substr(strip(ZIP9),1,&length.) ; /* The Zip format uses the first 5 digits */
label = "Found" ;    /* In most reference files this would be the description
                        and therefore the line would be label = columnxxx where
                        columnxxx is the column of the description in the reference file
                        Since there is no description for zip codes the text “Found”
                        is used as a place holder.
                     */
run ;
proc sort data=work.&source. ;
by  start label   ;
run ;

data work.&source. ;
set  work.&source. ;
by  start label   ;
if last.start ;      /* Only one record per unique value allowed.   */
run ;
```

```
data fmts.&destin. ;  /* This writes the format data under the same name as the format */
set  work.&source.  ;
fmtname = "&destin." ;  /* The format name is the destination defined above  */
type = "&type." ;
end   = start ;
length = &length. ;
/*if length(strip(&start.)) NE &length.  then delete ;*/
keep start end label type length fmtname ;
run ;
proc format cntlin=fmts.&destin. library=fmts.formats; /* This line creates the format */
run ;
```

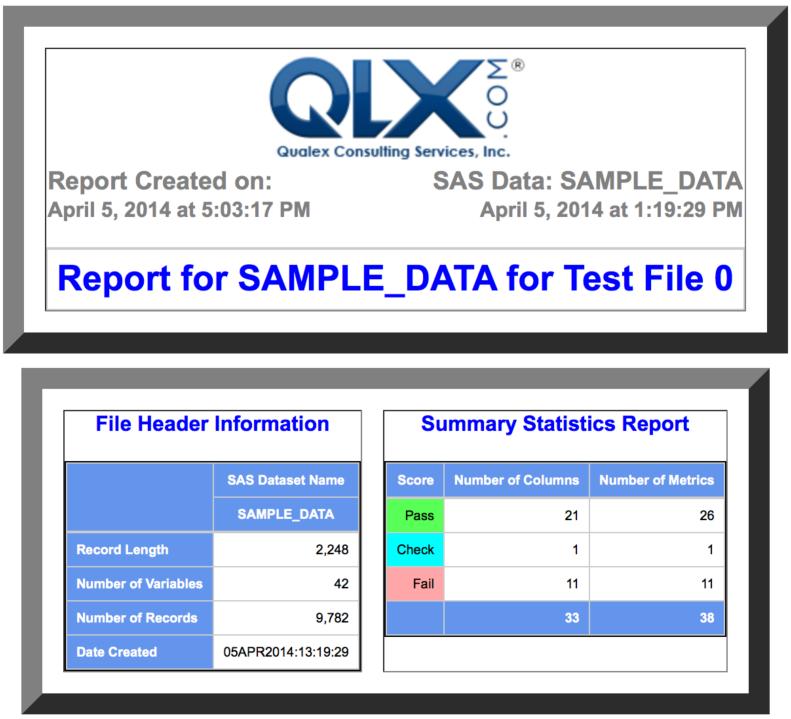
Many types of reference files can be adopted as rules. This is specifically useful when the file contains columns of codes and the database has dimension tables that describe the codes. Each dimension table can be converted into a SAS format and each value for a column in file can be checked to ensure it has a valid match in the dimension table. For example, if the file has a column for part number, and there is a dimension (or other table of part descriptors), a rule can be created to count the number of invalid part numbers. That is, rows in the files that contain a part number do not match any of the valid part descriptors in the reference table.

**Note that the formats are special tables loaded into memory. Too many formats used for reference files or formats that are very large may cause the system to run out of memory.**

## SCORECARD REPORT (ONE PER FILE)

### Section A: General Information

This section tells the user general information about the file, including number of columns and rows. A summary score section shows how many columns scored within each level (color) and how many metrics (rules) scored within each level. This summary gives the user a quick view of the data quality of file.



Qualex Consulting Services, Inc.

Report Created on:
April 5, 2014 at 5:03:17 PM

SAS Data: SAMPLE_DATA
April 5, 2014 at 1:19:29 PM

**Report for SAMPLE_DATA for Test File 0**

**File Header Information**

| | SAS Dataset Name |
|---|---|
| | SAMPLE_DATA |
| Record Length | 2,248 |
| Number of Variables | 42 |
| Number of Records | 9,782 |
| Date Created | 05APR2014:13:19:29 |

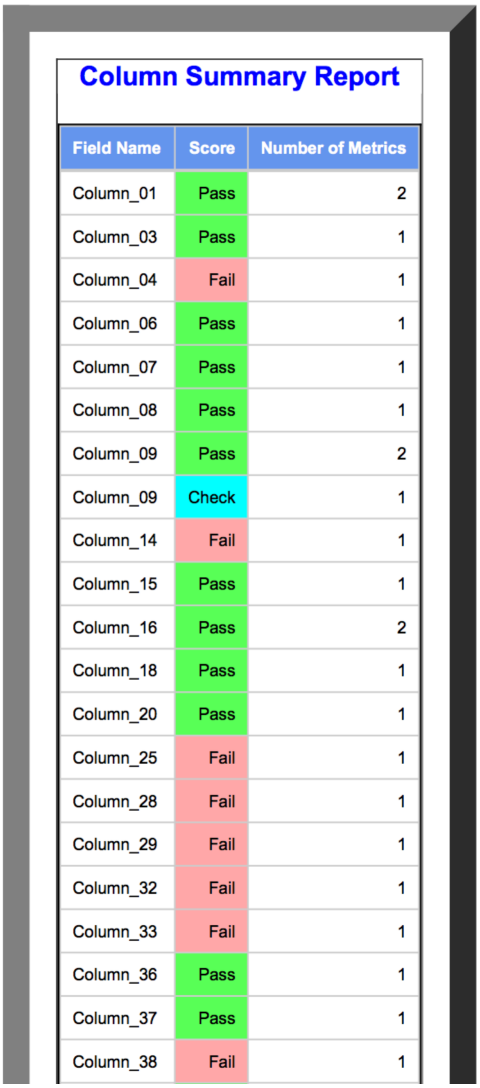
**Summary Statistics Report**

| Score | Number of Columns | Number of Metrics |
|---|---|---|
| Pass | 21 | 26 |
| Check | 1 | 1 |
| Fail | 11 | 11 |
| | 33 | 38 |

**Output 1. Section A of the Report (General Information)**

**Section B: The Column Level Report (Partial List)**

Section B of the report shows a detail report by column. This section shows each column and the number of rules by score (color). This is a first level of detail to understand where the issues with the file might reside.



**Column Summary Report**

| Field Name | Score | Number of Metrics |
|---|---|---|
| Column_01 | Pass | 2 |
| Column_03 | Pass | 1 |
| Column_04 | Fail | 1 |
| Column_06 | Pass | 1 |
| Column_07 | Pass | 1 |
| Column_08 | Pass | 1 |
| Column_09 | Pass | 2 |
| Column_09 | Check | 1 |
| Column_14 | Fail | 1 |
| Column_15 | Pass | 1 |
| Column_16 | Pass | 2 |
| Column_18 | Pass | 1 |
| Column_20 | Pass | 1 |
| Column_25 | Fail | 1 |
| Column_28 | Fail | 1 |
| Column_29 | Fail | 1 |
| Column_32 | Fail | 1 |
| Column_33 | Fail | 1 |
| Column_36 | Pass | 1 |
| Column_37 | Pass | 1 |
| Column_38 | Fail | 1 |

**Output 2. Section B of the Report (Column Summary)**

**Section C: The Column Details Report (Sample Partial List)**

Section C of the report is a detail list of each rule showing the column being evaluated, the number of rows that failed, the total number of rows processed, the percentage of rows that failed, and the score.

This part of the report also has a hyperlinked column for each rule that will launch any program the user directs to further evaluate the detail. Since this action is so flexible and dependent upon the source file, no example report is included in this paper.

## Column Details Report

| Column Name | Test / Metric | Rows Failed | Rows Processed | Percent Failed | Score |
|---|---|---|---|---|---|
| Column Name | Column Name: rows NULL | 8 | 592 | 1.4% | Fail |
| Column Name | Column NULL | 3 | 592 | 0.5% | Fail |
| Column Name | DOB: Date GT Run Date | 2 | 592 | 0.3% | Fail |
| Column Name | DOB: Date LT 1900 | 2 | 592 | 0.3% | Pass |
| Column Name | DOB: Column NULL | 6 | 592 | 1.0% | Fail |
| Column Name | Email: Column NULL | 25 | 592 | 4.2% | Fail |
| Column Name | Name: Column NULL | 9 | 592 | 1.5% | Fail |
| Column Name | Gender Column NULL | 9 | 592 | 1.5% | Fail |
| Column Name | Gender: evaluate distinct values | 13 | 592 | 2.2% | Fail |
| Column Name | Relationship: Column NULL | 3 | 592 | 0.5% | Fail |
| Column Name | Last Name: Column NULL | 4 | 592 | 0.7% | Fail |
| Column Name | Customer Id: Column NULL | 18 | 592 | 3.0% | Fail |
| Column Name | Customer Id: start w/0 | 2 | 592 | 0.3% | Fail |
| Column Name | Customer Phone: Column NULL | 6 | 592 | 1.0% | Check |

**Output 3. Section C of the Report (Column Detail)**

**Section D: Frequency Analysis (no output shown)**

Section 4 of the report is a frequency analysis of selected character columns (using PROC FREQ). This section of the report is used for analysis of columns that cannot be described by a simple rule. In some cases the user is interested in a frequency of values for a column. This information cannot be scored on a row-by-row basis. For this analysis the report can show the top X values for selected columns and the

top X values for the length of the selected columns. (X is a number that is user selected at the time the report is run).

For example, a user might wish to know the top 10 customer numbers to ensure that dummy numbers are not included (e.g., 111111111, 999999999). Often, dummy or test values are accidently included in production files. These dummy values are often seen much more frequently than valid values. Also, reviewing the lengths of the customer numbers might show that some percentage of values is too small to be valid customer numbers.

No section D report is included for numeric columns. However, a PROC MEANS or PROC UNIVARIATE could be added to create such a report.

## TAKING THE PROCESS FURTHER

### TIME SERIES REVIEW

Each time the user runs this application they create one summary SAS dataset for a file for a time period. By permanently storing these files the user can build a data mart of scores over time for every file type processed.

This data mart has three important uses. **First**, it can be used to establish data-driven standards for what can be expected from the input source. Standards can be adjusted if the reality of the data quality is different (higher or lower) than the expected. **Second**, the data mart can be used to trend data quality by file type and column over time. Data suppliers can be shown with confidence that their data quality is improving or deteriorating. **Third**, if the user has multiple suppliers of common data (e.g., vendors), the user can make inter-supplier comparisons of data quality to grade the suppliers against each other.

Building a data mart of quality scores has many advantages and provides the user more than a file-at-a-time report.

## CONCLUSION

The data quality scorecard solution is a simple but highly effective process that allows users to include the important data quality review process within a legacy ETL process. The process requires only Base SAS, is completely controlled by the user, and is highly flexible.

## CONTACT INFORMATION

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

Tom Purvis
Senior Consultant
Qualex Consulting Services, Inc.
email: tom.purvis@qlx.com
Skype: tompurvisjr
mobile: +1 904-233-3355

Clive Pearson
CEO
Qualex Consulting Services, Inc.
email: clive.pearson@qlx.com
www.qlx.com

## SAMPLE CODE

### SAMPLE TEMPLATE CODE

This code is a template used to determine if a character column is blank (null).

```
metric_id  = "&metric_id." ;
if strip(&col.) = "" then rowfailed = 1 ;
else                      rowfailed = 0 ;
output ;

%let metric_id = ; /* reset macro variables */
%let col       = ;
```

### SAMPLE METRIC CODE

This code is a sample stored process showing how the templates might be used to count rows failed for later scoring. Ultimately this code would include a section for each rule (metric) and there is one program for each file type. The code highlighted in yellow below shows how the sample template code above would be called.

```
*ProcessBody ;
/*filename _webout "<path>/htmlout.html" ;*/

%let inputname = <name of input file> ;
%let standard  = <name of accompanying standard>  ;

/* Parameters for Section D of report (character columns to review frequency) */

%let num = 3  ; /* how many columns to include in the table report must equal the list below */
%let col1 = <column name> ;
%let col2 = <column name> ;
%let col3 = <column name> ;
/* however many columns set in “let num =” should have a “let colN” statement */

libname sasin BASE "<path>"  ;

libname fmts  BASE "<path>" ;

%let path      = <path>/includes/ ;
%let pathscore = <path>/programs/ ;

%let _ODSDEST=TAGSETS.HTMLPANEL;
%let _odsoptions = rs=none;
%let _odsstyle = &style ;
%let _ODSSTYLESHEET= ;

%let embedded_titles = yes ;
%let panelborder = 16 ;

ODS ESCAPECHAR= '^' ;

options device=activex  fmtsearch=(fmts.formats) ; /* libname fmts must be set */

%STPBEGIN ;
ODS ESCAPECHAR= '^' ;
```

```
data work.metrics  ;
length metric_id $ 16 rowfailed rows_processed 8 ;
set  sasin.&inputname. end=end ;                  /* libname sasin must be set */
keep metric_id rowfailed rows_processed ;
rows_processed + 1 ;

/*First Rule    XXXXXXX001 (e.g. Check NULL/Blank Value) */

/******** XXXXXXXXX001 **********/

%let metric_id = XXXXXXXXX001  ;
%let col       = <numeric column name> ;
%include "&path.numcolnull.sas" / source2 ; /* this calls template for number column is null */



/*Second Rule      XXXXXXX002 (e.g. Check NULL/Blank Value for character column) */

/******** XXXXXXXXX002 **********/

%let metric_id = XXXXXXXXX002 ;
%let col       = <character column name> ;
%include "&path.charcolnull.sas" / source2 ; /* this calls template for character column is null */



/*Third  Rule        XXXXXXX003 (e.g. Check character column to ensure it is on reference list ) */
%let metric_id = XXXXXXX003 ;
%let col       = <character column>;
%let fmttest   = $<character format name>. ;  /* this format would have been previously created from reference files */
%include "&path.charformat.sas" /source2   ;  /* this calls the template to check that a character column is found
                                                 within a format */

....  Add as many rules here as are defined in the standards spreadsheet ......
run ;

/* this SQL summarizes the file above into one record per metric (rule)  */
proc sql ;
create table work.&inputname._metrics as                  /* this table may or may not be permanent */
select metric_id ,
      sum(rowfailed) as rows_failed format=comma15.0  ,
            max(rows_processed) as rows_processed format=comma15.0 ,
            sum(rowfailed) / max(rows_processed) as percent_failed format=6.4
from work.metrics
group by metric_id
;
quit ;

%include "&pathscore.scorereport.sas" /source2 ; /* this calls the score report */
%stpend ;
```

## SCORE REPORT CODE

```
%let metricfile = sasin.&inputname._metrics    ;

proc sort data=work.&inputname._metrics
   ;
by Metric_ID ;
run ;
```

```
proc sort data=sasin.&standard
          out=work.&standard
           ;
by Metric_ID  ;
where metric_id NE "" ;
run ;

data work.&inputname._scored   ;
merge work.&inputname._metrics (in=m)
      work.&standard.    (in=s)
      ;
by Metric_ID  ;
length Score 8 ;
if m ;

/* this creates the hyperlink to the stored process for detail analysis
   it is set up to pass two macro variables as parameters  ParamA and odsstyle
   but others can be added
*/
machine = "http://111.222.333.444:8080/" ;
engine  = "SASStoredProcess/do?" ;
progpath = "_program=SBIP://METASERVER%2F<folder>%2F<folder>%2F<folder>%2F" ;
program  = "<detail report name>" ;
paramA   = "colname="  ;
odsstyle = "&_odsstyle." ;
 ;

/* this creates the hyperlink */
htmlvar = "<a href=" ||
""" || /* beginning single quote for href */
trim(machine) ||
trim(engine) ||
trim(progpath) ||
trim(program) || '&' ||
trim(paramA)  ||  strip(column_name)   ||
'&_odsstyle='       || strip(odsstyle) ||
"'>" ;

Metric_ID_linked = htmlvar ||  strip(Metric_ID) || '</a>' ;

Based_On = "Percent" ;

if      Based_On = "Percent" then Measure = Percent_Failed * 100;
else if Based_On = "Rows"    then do ;
       Measure = Rows_Failed ;
                    if red_end = 999999999 then red_end = Rows_Processed ;
                    end ;
if green_beg <= red_end   then     /* Lower is Better  */
  do ;                                                                  /*          Lower Better        */
    if      Green_beg <=  Measure  <=  Green_end  then Score = 1 ;      /*        0  <=  x  <=  50      */
    else if  Green_end <   Measure  <=  Blue_end   then Score = 2  ;    /*       50  <   x  <=  90      */
    else if  Blue_end  <   Measure  <=  Red_end    then Score = 3   ;   /*       90  <   x  <= 100      */
    else                                  Score = 0 ;
  end ;
else                        /* Higher is Better  */
  do ;                                                                  /*         Higher Better       */
    if      Green_beg >=  Measure  >=  Green_end  then Score = 1  ;     /*       100 >= x >=  90     */
    else if  Green_end >   Measure  >=  Blue_end   then Score = 2 ;     /*        90 >  x >=  50      */
    else if  Blue_end  >   Measure  >=  Red_end    then Score = 3   ;   /*        50 >  x >=   0      */
    else                                  Score =  4 ;
  end ;
label Metric_ID_linked          = "Column Name"
```

```
     Metric_ID                    = "Metric_ID"
              Score               = "Score"
              Rows_Failed         = "Rows Failed"
              Rows_Processed      = "Rows Processed"
              Percent_Failed      = "Percent Failed"
              ;
run ;

proc sort data=work.&inputname._scored  ;
by Metric_id data_field_name  ;
run ;

proc format ;
 value score
   1 = "Pass"
   2 = "Check"
   3 = "Fail"
   ;
   value colorsc      /* This format assigns colors to cells */
   1 = 'Light Green'
   2 = 'Cyan'
   3 = 'Very Light Red'
   ;
run ;


Proc sql ;
create table work.filesummary as
select
     Score  ,
     put(score,score.) as ScoreLabel ,
     count(distinct Data_field_name) as Columns ,
     count(distinct Metric_ID) as Metrics
from work.&inputname._scored
group by 1 , 2
;

create table work.scoresummary as
select data_field_name ,
       Score ,
       count( distinct Metric_ID) as Metrics
from work.&inputname._scored
group by 1, 2
;
quit ;

proc contents data=sasin.&inputname. noprint out=work.contents ;
run ;

proc sql noprint ;
create table work.contents2 as
select memname ,
       sum(length) as record_length format=comma20.0,
       max(varnum) as number_variables format=comma15.0,
       max(nobs) as Number_records format=comma20.0 ,
       max(crdate) as date_created format=datetime25.
from work.contents
group by memname
;
select put(datepart(max(crdate)),worddate28.) into :filedate /* use of into with : creates a macro variable within SQL */
from work.contents
;
```

```
select put(timepart(max(crdate)),timeampm12.) into :filetime
from work.contents
;
select memname into :datasetname
from work.contents
quit ;

/* This creates section A of the Report */

%let panelcolumns = 1  ;   /* Tells ODS how many panels to use */

ods &_ODSDEST event=panel(start);

data test ;
length text $ 255 ;
text = "Report for %UPCASE(&inputname.) for &client." ;
output ;
run ;


title1   j=center '^S={preimage="http://<path to logo>"}' ;

title2    j=left height=14pt color=gray "Report Created on:"
         j=right height=14pt color=gray  "SAS Data: &datasetname." ;
title3
         j=left height=12pt color=gray "%TRIM(%QSYSFUNC(DATE(), worddate28.)) at %TRIM(%SYSFUNC(TIME(),
TIMEAMPM12.))"
         j=right height=12pt color=gray   "&filedate. at &filetime." ;

proc report data=test nowd noheader style(report)={rules=none frame=void }
    style(column)={font_weight=bold font_size=20pt just=center foreground=blue /*background=white*/ } ;
run;

ods &_ODSDEST event=panel(finish);

%let panelcolumns = 2  ;

ods &_ODSDEST event=panel(start);

title ;
title2 h=12pt color=blue
       "File Header Information" ;
footnote ;


PROC TABULATE
DATA=WORK.CONTENTS2           ;
            VAR record_length number_variables Number_records date_created;
            CLASS MEMNAME /         ORDER=UNFORMATTED MISSING ;
            TABLE
        record_length     = "Record Length"       *f=comma20.0
        number_variables  = "Number of Variables" *f=comma15.0
        Number_records    = "Number of Records"   *f=comma20.0
        date_created      = "Date Created"        *f=datetime25.
    ,
        Sum = "" *
        MEMNAME = "SAS Dataset Name"                    ;
             ;

RUN;
```

```
title2 h=12pt color=blue
       "Summary Statistics Report" ;
footnote ;
proc print data=work.filesummary label noobs ;
var score / style={background=colorsc.}  ; /* This applies the color format */
var Columns  metrics ;
sum columns metrics ;
format score score. ;
label Score = "Score"
     Columns = "Number of Columns"
               Metrics = "Number of Metrics"
               ;
format columns
      metrics   comma20.0
                ;
run ;


ods &_ODSDEST event=panel(finish);

/* This creates section B of the Report */

%let panelcolumns = 1  ;

ods &_ODSDEST event=panel(start);

title2 h=12pt color=blue
       "Column Summary Report" ;
footnote ;

proc print data=work.scoresummary label noobs ;
var data_field_name ;
var     score / style={background=colorsc.}  ; /* This applies the color format */
var   metrics ;
sum metrics ;
label Data_field_name = "Field Name"

               Metrics = "Number of Metrics"
               ;
format
      metrics   comma20.0
                score score.
                ;
run ;


ods &_ODSDEST event=panel(finish);


/* This creates section C of the Report */

%let panelcolumns = 1  ;
ods &_ODSDEST event=panel(start);

title2 h=12pt color=blue
       "Column and Metric Details Report" ;
footnote ;

proc print data= work.&inputname._scored label noobs ;
var   metric_id_linked data_field_name metric_label Rows_Failed  Rows_Processed Percent_Failed ;
var score / style={background=colorsc.}  ;
format score score.
```

```
      percent_failed percent6.1
                ;
label Data_field_name = "Column Name"
     metric_label    = "Metric Description"
;
run ;

ods &_ODSDEST event=panel(finish);


/* This creates section D of the Report */

%let panelcolumns = 2  ;
ods &_ODSDEST event=panel(start);

%macro Freq(num,top)  ;

data work.&inputname. ;
set  sasin.&inputname. ;
keep
  %do I= 1 %to &num.  ;
     &&col&I.. &&col&I.._len
  %end ;
 ;
  %do I = 1 %to &num. ;
     &&col&I.._len  = length(&&col&I..) ;
  %end ;
run ;

proc freq data=work.&inputname. noprint  order=freq ;

  %do I = 1 %to &num.  ;
     table &&col&I.. / missing out=work.F_&&col&I..  ;
              table &&col&I.._len / missing out=work.F_&&col&I.._len ;
  %end ;
run ;

%do I = 1 %to &num.  ;
    Title1 "Top &top. Frequency for Values of &&col&i.." ;
      proc print data=work.F_&&col&I.. (obs=&top.) ;
      run ;
    Title1 "Top &top. Frequency for Length of &&col&i.." ;
      proc print data=work.F_&&col&I.._len (obs=&top.) ;
      run ;
%end ;

%mend ;

%Freq(&num.,&top.)  ;

ods &_ODSDEST event=panel(finish);
```