USING SASHELP TO VALIDATE ATTRIBUTES

Sadia Abdullah, Sr. SAS Programmer, PPD Inc., Austin TX

Abstract

SASHELP is a SAS library that is made available at the start of a session. It contains views or tables which provide information about the session. These views or tables contain what can be considered as the metadata of the entire SAS session.

This paper gives an introduction to SASHELP and how it can be used in various situations. It lists some important views from this library and mentions the application of all these views.

Introduction

Since what SASHELP contains is the metadata of all the data in the existing session, it can conveniently be used to crosscheck the attributes of the data generated in a SAS session with its specifications. In this paper we are comparing the attributes from a SAS dataset with its specifications existing in excel format.

With SASHELP selective comparison can take place with ease, for example checking to see if all the date variables in a libref have the same format, or if a certain variable has the same attributes across all datasets in a SAS session. In certain cases when the same data is generated from two different sources their informats may vary, in such cases PROC COMPARE would show mismatches, however using SASHELP views selective comparison can be done to check if the attributes that need to match are the same across both datasets.

Dictionary tables and SASHELP views:

All the information about a SAS session exists in the Dictionary tables, and can be accessed using PROC SQL. It is replicated in the SASHELP views and hence the dictionary tables and SASHELP views are very similar in their content.

DICTIONARY tables are read-only tables that contain information about the SAS data libraries, data sets, system options, and external files in the current SAS session. SAS assigns an automatic libref DICTIONARY for the dictionary tables. They can be accessed using PROC SQL only.

SASHELP views exist in the SASHELP library and they contain the same information as in the Dictionary tables. They are different from the DICTIONARY tables in that they can be accessed using both PROC SQL and data step programming. In an interactive SAS session you can also navigate to SASHELP library under the EXPLORER tab.

Another difference in the Dictionary tables and SASHELP views is the processing time when a WHERE clause is applied. The WHERE clause is processed at a faster speed when applied to a dictionary table in comparison to a view. This happens because all the rows need to be generated before a WHERE clause is applied to a SASHELP view,

where as only the pertinent rows are generated when it is applied to a Dictionary table making it faster in comparison.

Some of the most commonly used views are:

VCOLUMN:
It provides detailed information about each variable contained in all the datasets present in the SAS session.
Note: Since VCOLUMN provides information on variables, if a dataset does not contain any variables then that dataset will not show up under memname in VCOLUMN.

VTABLE:
This view provides detailed information about SAS data files. It lists the name of the dataset, libname to which the dataset belongs and number of observations and variables in the dataset.

VMEMBER:
This view is similar to VTABLE, however it does not provide as much information. One of the useful variables it provides which is not present in VTABLE is PATH, which specifies the physical path of the dataset.

VTITLE:
This view has an entry for each title and footnote line in effect in the SAS session.

VEXTFL:
This view lists all the external files being used in the current SAS session and their paths

VCATALG:
This view provides the catalog name in MEMNAME, the libref in which it exists and the names of the formats contained in the catalog in OBJNAME.

To see all the views in SASHELP library:

```
PROC SQL;
 CREATE TABLE VIEWS AS
 SELECT DISTINCT MEMNAME
 FROM SASHELP.VSVIEW
 WHERE LIBNAME='SASHELP'AND MEMNAME LIKE 'V%';
QUIT;
```

To see all the tables in libref DICTIONARY:

```
PROC SQL;
 CREATE TABLE TABLES AS
 SELECT * FROM DICTIONARY.DICTIONARIES;
QUIT;
```

Below are some examples of how SASHELP Views can be used. An advantage of using SASHELP views which is very obvious in the below examples is that each dataset does

not have to be queried individually to look for any condition, a collective query can be launched using SASHELP.

Example 1 – To concatenate variables from a dataset into a string and assign its value to a Macro variable

```
%MACRO VAR (LIB , DSET);
 PROC SQL;
  SELECT NAME INTO: VAR SEPARATED BY ","
  FROM SASHELP.VCOLUMN WHERE LIBNAME=&LIB AND MEMNAME=&DSET;
 QUIT;
%MEND;
%VAR("EXTRACT","DEMO");
```

Example 2 – To output a list of SAS data sets with number of observations greater than zero

```
%MACRO NOBS(LIB);
 PROC SQL;
  SELECT DISTINCT MEMNAME AS
  DTS FROM SASHELP.VTABLE
  WHERE LIBNAME=&LIB AND NOBS > 0;
 QUIT;
%MEND;

%NOBS(OCVTEST);
%NOBS(EXTRACT);
```

Example 3 – To query if any or what type of indexes are associated with variables

```
PROC SQL;
 SELECT LIBNAME, MEMNAME, NOBS, INDXTYPE
 FROM SASHELP.VTABLE
QUIT;
```

Example 4 – To check if there are any variables ending with “1O” which do not have the format DATETIME20.

```
DATA DT_CHK;
 SET SASHELP.VCOLUMN;
 X=LENGTH(NAME);
 IF X > 3 THEN DO;
  IF SUBSTR(NAME,X-1) = "1O" AND FORMAT NE "DATETIME20." THEN OUTPUT;
 END;
 KEEP LIBNAME MEMNAME NAME FORMAT;
RUN;
```

Example 5 – To validate the attributes of datasets with their specifications.

SASHELP can be used to programmatically validate the attributes of the variables in a dataset. As an example, I am validating the attributes of an analysis dataset, but the code

can be used to validate any type of dataset. The specification document for this example is in excel format with the specification of each dataset in a separate sheet.

I have coded a macro for this purpose and named it SPEC_IN.

```
%MACRO SPEC_IN;
/*PHYSICAL LOCATION OF THE SPEC*/
LIBNAME SPEC EXCEL "U:\SCSUG\SPEC.XLS";
LIBNAME DERIVED "U:\SCSUG\DERIVED";
LIBNAME ATTR "U:\SCSUG";

/*GETTING THE SHEETNAME FROM SASHELP.VSTABVW INTO A DATASET*/
DATA SHEETNAME;
 SET SASHELP.VSTABVW(WHERE=(LIBNAME="SPEC")
 RENAME=(MEMNAME=SHEETNAME));
 SHEETNAME=COMPRESS(SHEETNAME,'$');
 KEEP SHEETNAME;
RUN;

PROC SQL NOPRINT;
/*COUNTING THE NO. OF SHEETS IN THE EXCEL FILE*/
 SELECT COUNT(DISTINCT SHEETNAME) INTO :CNT_SHT
 FROM SHEETNAME;

/*CREATING MACRO VARIABLES FOR SHEET NUMBERS*/
 SELECT DISTINCT SHEETNAME INTO :SHEET1 - :SHEET%LEFT(&CNT_SHT)
 FROM SHEETNAME;
QUIT;
%DO I=1 %TO &CNT_SHT;
/*IMPORTING THE SHEETS TO SAS RENAMING THE VARIABLES TO MATCH THE VCOLUMN
VARIABLES*/
PROC IMPORT DATAFILE="U:\SCSUG\SPEC.XLS"
 OUT=attr.&&SHEET&I;
 SHEET="&&SHEET&I";
 GETNAMES=YES;
 MIXED=YES;
RUN;

/*GETTING THE ATTRIBUTES OF SDTM DATASETS FROM SASHELP.VCOLUMN*/
DATA &&SHEET&I;
 SET SASHELP.VCOLUMN (WHERE= (LIBNAME='DERIVED' AND MEMNAME="&&SHEET&I"));
 KEEP NAME TYPE LENGTH VARNUM LABEL FORMAT;
RUN;

/*COMPARING THE ATTRIBUTES*/
PROC COMPARE
 BASE=&&SHEET&I/*ATTRIBUTE FROM SDTM DATASET*/
 COMP=attr.&&SHEET&I/*ATTRIBUTES FROM SPEC*/
 LISTALL;
RUN;
%END;

%MEND SPEC_IN;
%SPEC_IN;
```

The steps for this macro are explained here:

1) Libnames are assigned for the macro. Libname SPEC is assigned for specifications document, libname DERIVE is assigned for the derived datasets whose attributes need to be validated and libname ATTR is assigned for datasets to be stored after being imported from the specifications document.
2) The names of the sheets for the specifications workbook are pulled in from SASHELP.VSTABVW.
3) The sheets in the workbook are counted and the count is stored in a macro variable CNT_SHT
4) A macro variable for each sheet is created
5) The next part of the code is a DO loop which is iterated for each sheet in the excel file.
6) The first step inside the DO loop is to import the excel file with a SHEET= option. The resulting dataset is stored under libref ATTR, this dataset contains specifications for the dataset.
7) Next inside the DO loop the attributes of the derived dataset are subsetted from SASHELP.VCOLUMN by specifying a WHERE clause for the LIBNAME and MEMNAME.
8) Just before closing the DO loop the attributes of the derived dataset from SASHELP.VCOLUMN is compared with the specifications dataset using a PROC COMPARE

Conclusion:

SASHELP is a very powerful source of information and there are numerous ways in which it can be used. This paper has hopefully given some insight into a few of the ways in which it can be used.

Acknowledgements:

I would like to thank the South Central SAS Users Group for accepting my abstract and paper as well as for conducting a very successful conference. I would also like to thank Jeanina Worden and Zhuo Chen for reviewing my work and providing feedback.

References:

Make the Invisible Visible: A Case Study of Importing Multiple Worksheet Files
By Using the SAS®9 LIBNAME Engine in Microsoft Excel
Zizhong Fan, Westat, Rockville, MD

Exploring DICTIONARY Tables and SASHELP Views
Kirk Paul Lafler, Software Intelligence Corporation

You Could Look It Up:
An Introduction to SASHELP Dictionary Views
Michael Davis, Bassett Consulting Services, North Haven, Connecticut

Dictionary Tables and SASHELP views
studysas.blogspot.com

Uma Sarath Annapareddy