Paper BB-11

Know Thy Data: Techniques for Data Exploration

Charu Shankar, SAS® Canada

ABSTRACT

Get to know the #1 rule for data specialists: Know thy data. What are the common keys for joins? Are there data type conflicts? How to locate changed variable names? How to reorder variables in the dataset without physically typing in all the names but instead using metadata to perform this action. In this session, you will learn to employ powerful PROC SQL's dictionary tables to easily explore aspects of your metadata.

SECTION 1: INTRODUCTION

Before 1900 the Pima Indians of Arizona were one of the world's healthiest ethnic groups. Diabetes was unheard of. Things exploded in the 1970s. At 38% and climbing in 2006, the Pima had the highest rate of diabetes of any population in the world. They also had staggering rates of obesity (~70%) and hypertension.

Investigators are interested in examining the occurrence of Type 2 diabetes in women of Pima Indian heritage who are at least 21 years old.

Using interesting and compelling data of the Pima Indians, learn multiple data exploration techniques in this session to get to know your data.

Display 1. The population of interest lives in Phoenix, Arizona

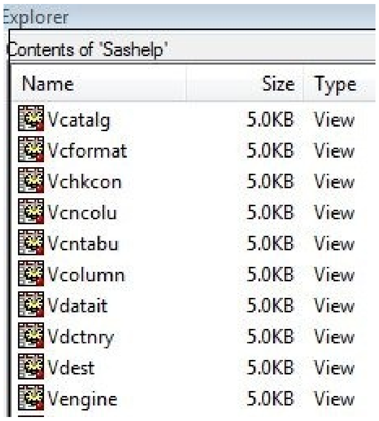
SECTION 2: PROC SQL DICTIONARY TABLES

Have you ever wished for more than what PROC CONTENTS can deliver on your metadata? Ever right clicked on a SAS dataset in the SAS explorer window to check properties and wanted more? Have you ever wanted to go behind the scenes and get information on all the titles, macros, datasets, etc.? Here is a great way to get to know your data through powerful dictionary tables.

Dictionary tables contain a wealth of information about your SAS session. They are special read-only PROC SQL tables or views. They are created upon SAS invocation, updated automatically by SAS and are available throughout a SAS session. They provide information about SAS libraries, SAS data sets, SAS system options, and external files that are associated with the current SAS session and much, much more.

2.1 Examine dictionary tables

Where do we begin to look up this amazing information? Open up the SASHELP library to view available dictionary tables:



Explorer

Contents of 'Sashelp'

Name	Size	Type
Vcatalg	5.0KB	View
Vcformat	5.0KB	View
Vchkcon	5.0KB	View
Vcncolu	5.0KB	View
Vcntabu	5.0KB	View
Vcolumn	5.0KB	View
Vdatait	5.0KB	View
Vdctnry	5.0KB	View
Vdest	5.0KB	View
Vengine	5.0KB	View

Display 2. SASHELP views

Alternatively submit the following code to display all supported dictionary tables and views.

```
proc sql ;
select distinct memname, memlabel
from dictionary.dictionaries;
quit;
```

Display 3. Examine metadata

The following table describes available DICTIONARY tables and associated SASHELP views.

	SASHELP View	Description
CATALOGS	VCATALG	Contains information about known SAS catalogs.
CHECK_CONSTRAINTS	VCHKCON	Contains information about known check constraints.
COLUMNS	VCOLUMN	Contains information about columns in all known tables.
CONSTRAINT_COLUMN_USAGE	VCNCOLU	Contains information about columns that are referred to by integrity constraints.
CONSTRAINT_TABLE_USAGE	VCNTABU	Contains information about tables that have integrity constraints defined on them.
DATAITEMS	VDATAIT	Contains information about known information map data items.

DESTINATIONS	VDEST	Contains information about known ODS destinations.
DICTIONARIES	VDCTNRY	Contains information about all DICTIONARY tables.
ENGINES	VENGINE	Contains information about SAS engines.
EXTFILES	VEXTFL	Contains information about known external files.
FILTERS	VFILTER	Contains information about known information map filters.
FORMATS	VFORMAT VCFORMAT	Contains information about currently accessible formats and informats.
FUNCTIONS	VFUNC	Contains information about currently accessible functions.
GOPTIONS	VGOPT VALLOPT	Contains information about currently defined graphics options (SAS/GRAPH software). SASHELP.VALLOPT includes SAS system options as well as graphics options.
INDEXES	VINDEX	Contains information about known indexes.
INFOMAPS	VINFOMP	Contains information about known information maps.
LIBNAMES	VLIBNAM	Contains information about currently defined SAS libraries.
MACROS	VMACRO	Contains information about currently defined macro variables.
MEMBERS	VMEMBER VSACCES VSCATLG VSLIB VSTABLE VSTABVW VSVIEW	Contains information about all objects that are in currently defined SAS libraries. SASHELP.VMEMBER contains information for all member types; the other SASHELP views are specific to particular member types (such as tables or views).
OPTIONS	VOPTION VALLOPT	Contains information about SAS system options. SASHELP.VALLOPT includes graphics options as well as SAS system options.
REFERENTIAL_CONSTRAINTS	VREFCON	Contains information about referential constraints.
REMEMBER	VREMEMB	Contains information about known remembers.
STYLES	VSTYLE	Contains information about known ODS styles.
TABLE_CONSTRAINTS	VTABCON	Contains information about integrity constraints in all known tables.
TABLES	VTABLE	Contains information about known tables.
TITLES	VTITLE	Contains information about currently

		defined titles and footnotes.
VIEWS	VVIEW	Contains information about known data views.

Display 4. Dictionary tables and associated SASHELP views

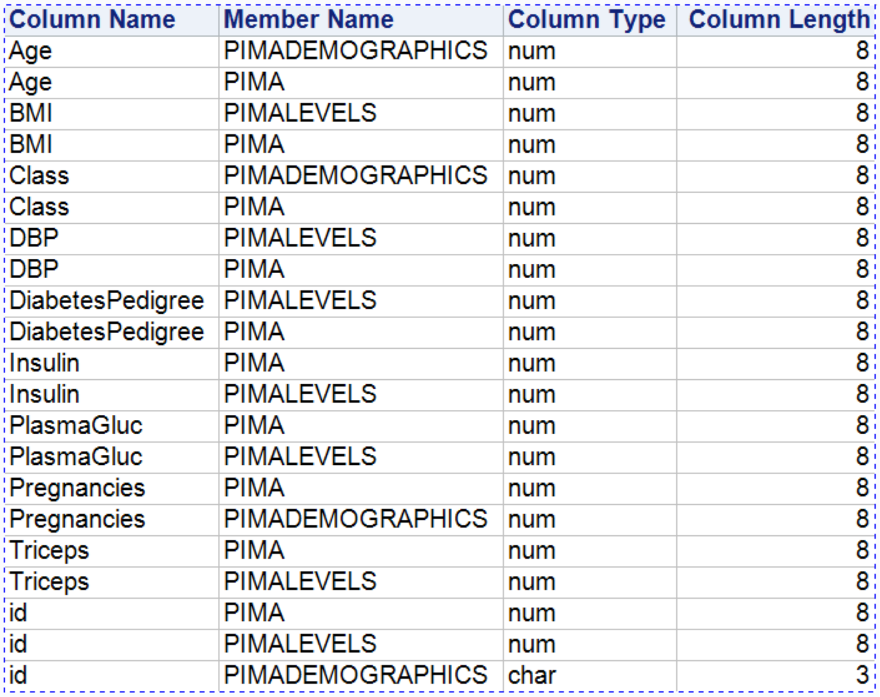
2.2 Investigate common columns for joins

Do you perform complex joins? And you don't know where to begin since you haven't got the information on common columns for joining data? Let Dictionary tables come to your help.

We will turn to the COLUMNS dictionary table to find common columns

```
proc sql;
   select name, memname, type, length
   from dictionary.columns
   where libname ='DIABETES'
   group by name
   having count(name) > 1
   order by name;
quit;
```

Display 5. Code to examine all common columns in the DIABETES library



Column Name	Member Name	Column Type	Column Length
Age	PIMADEMOGRAPHICS	num	8
Age	PIMA	num	8
BMI	PIMALEVELS	num	8
BMI	PIMA	num	8
Class	PIMADEMOGRAPHICS	num	8
Class	PIMA	num	8
DBP	PIMALEVELS	num	8
DBP	PIMA	num	8
DiabetesPedigree	PIMALEVELS	num	8
DiabetesPedigree	PIMA	num	8
Insulin	PIMA	num	8
Insulin	PIMALEVELS	num	8
PlasmaGluc	PIMA	num	8
PlasmaGluc	PIMALEVELS	num	8
Pregnancies	PIMA	num	8
Pregnancies	PIMADEMOGRAPHICS	num	8
Triceps	PIMA	num	8
Triceps	PIMALEVELS	num	8
id	PIMA	num	8
id	PIMALEVELS	num	8
id	PIMADEMOGRAPHICS	char	3

Display 6. Partial display SAS output -common columns in the DIABETES Library

2.3 Efficiency wise – SQL or other Procs?

Certainly dictionary tables can be accessed either through PROC SQL or SAS procedures/data step code.

```
proc sql;
   select libname, memname, name, type, length
   from dictionary.columns
   where libname ='DIABETES' and upcase(name) contains 'ID';
quit;
NOTE: PROCEDURE SQL used (Total process time):
  real time           0.45 seconds
  user cpu time       0.00 seconds
  system cpu time     0.06 seconds
  memory              5161.46k
  OS Memory           29176.00k
  Timestamp           03/24/2018 08:21:06 PM
```

Display 7. PROC SQL to locate all ID columns in the DIABETES library

If you prefer to use proc print to locate all ID columns in the DIABETES library in your SAS session, then take a look at the difference in resource usage.

```
proc print data=sashelp.vcolumn label noobs;
var libname memname name type length;
where libname ='DIABETES' and   upcase(name) contains 'ID';
run;
NOTE: There were 6 observations read from the data set SASHELP.VCOLUMN.
      WHERE (libname='DIABETES') and UPCASE(name) contains 'ID';
NOTE: PROCEDURE PRINT used (Total process time):
      real time           2.71 seconds
      user cpu time       0.90 seconds
      system cpu time     1.35 seconds
      memory              6727.18k
      OS Memory           29688.00k
      Timestamp           03/24/2018 08:23:37 PM
```

Display 8. Resource usage with PROC PRINT querying dictionary tables in SAS log

Here is why PROC PRINT uses more resources. While querying a DICTIONARY table, SAS launches a discovery process. Depending on the DICTIONARY table being queried, this discovery process can search libraries, open tables, and execute views. The PROC SQL step runs much faster than other SAS procedures and the DATA step because the WHERE clause is processed before the tables referenced by the SASHELP.VCOLUMN view are opened.

Therefore it’s more efficient to use PROC SQL instead of the DATA Step or SAS procedures to query dictionary tables.

2.4 Reorder variables in dataset

Data workers frequently request a change in the physical order of variables. Here are their reasons:

1. Display PROC PRINT output in alphabetic variable order. Use a variable list shortcut without explicitly having to type out variable names with the VAR statement.

2. Send SAS output to Excel to help the EXCEL user eliminate manual reordering.

Here’s what happens when you try to use a variable list shortcut. The log complains that variables are out of order.

```
proc print data=diabetes.pima;
var dbp--id;
ERROR: Starting variable after ending variable in data set.
212  run;
```

Display 9. Log note when Variable list shortcut for PROC PRINT fails

How do you get variables in order without doing any manual sorting of names & then typing? Let’s utilize a powerful synergy between proc sql and the macro language. We’ll store all the variables from the DIABETES.PIMA dataset in alphabetical order into a macro called newname. This technique uses the INTO clause to pass data values from the dataset into a macro.

```
proc contents data=diabetes.pima varnum;
run;

proc sql noprint;
select name into :newname separated by ","
from dictionary.columns
where libname ='DIABETES' and
upcase(memname) ='PIMA'
order by name;
quit;
```

Display 10. Code to put variables in alpha order using dictionary tables

Now read the alphabetically ordered variables just created into a dataset. Voila! No hardcoding required and you have what you asked for – all variables are stored in alphabetical order.

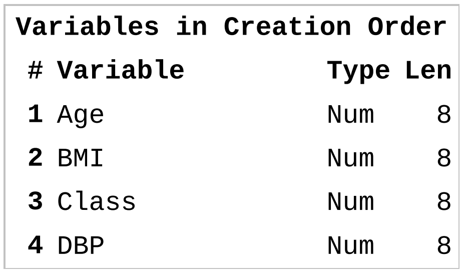
```
proc sql;
create table ordered as
select &newname
from diabetes.Pima;
quit;
```

Display 11. Create table with variables in alpha order

Submit a PROC CONTENTS to verify the order.

```
proc contents data=ordered varnum;
run;
```

Display 12. Proc contents to verify variables are in alpha order



```
Variables in Creation Order
 # Variable          Type Len
 1 Age               Num    8
 2 BMI               Num    8
 3 Class             Num    8
 4 DBP               Num    8
```

```
Variables in Creation Order
 # Variable          Type Len
 5 DiabetesPedigree  Num    8
 6 Insulin           Num    8
 7 PlasmaGluc        Num    8
 8 Pregnancies       Num    8
 9 Triceps           Num    8
10 id                Num    8
```

Display 13. Neat and tidy variables stored in alpha order, PROC CONTENTS output

2.5 Isolate variable type conflicts

How often have you been stumped with a variable type mismatch while trying to join tables on a common key? Wouldn't it be more effective to know your data before you start joining? This will help eliminate any surprises and more importantly conserve time when you have an important deadline to meet.

Gather information on type conflicts by the clever use of the Count function.

```
proc sql ;
select libname, memname, name, type, length
from dictionary.columns
where upcase(name) contains 'ID' and libname='DIABETES'
group by name
having count(distinct type) > 1
order by 1, 2
;
quit;
```

Display 14. Isolate variable conflicts using dictionary tables

```
Library Name Member Name       Column Name Column Type  Column Length
DIABETES     PIMA              id          num                      8
DIABETES     PIMADEMOGRAPHICS id          char                     3
DIABETES     PIMALEVELS        id          Num                      8
```

Display 15. Dataset Compare shows type conflicts for the id column

2.6 Identify working folder and a cool SAS option

What if you want to identify the working folder and pass this information to a SAS program to run from that location? You know you can easily lookup the working folder by going to lower right portion of your interactive SAS session. But what if you want to store the path programmatically to use and reuse. Additionally what if you are not working in Windows & don't have access to an interactive SAS session.

The "working folder", also known as the SASinitialFolder, is an important concept to know when reading and saving data sets, formats, macros or programs read from external files, or other objects written or read within the session. The location of this folder is the default path where SAS reads or stores files when a specific drive and pathname isn't given. For example, if you do not provide a drive and pathname within the statements such as LIBNAME, FILENAME, INFILE, %INCLUDE or other statements that refer to external files or directories, SAS looks for these files in the working folder.

Let’s turn to the extfiles dictionary table to grab the path, store it in a macro variable and use it to call a program stored in the working folder.

```
options symbolgen;
%macro CurrDir;
filename _temp '.';
%global Current;
proc sql ;
  select xpath into :Current TRIMMED
  from dictionary.extfiles
  where fileref = '_TEMP';
quit;
filename _temp clear;
%put _user_;
%mend;
```

Display 16. Picking working directory folder using dictionary tables

Note: when you create macro variables via SQL it preserves leading and trailing blanks. If you were to run a describe on the dictionary.extfiles table you would see that the XPATH column you are querying is 1024 bytes in length thus all the trailing blanks, which is really at the heart of the problem.

This is why we used a cool new option in SAS 9.3. Specify TRIMMED on the "INTO" clause to avoid having to post-process the macro variables with %let.

If running SAS 9.2 after QUIT add %LET CURRENT=&CURRENT which trims leading & trailing blanks

Now use the macro to call the alloptions.sas program stored in the working directory.

```
%include "&current\alloptions.sas”;
title "Notice no date which was the alloptions program being called
by the %include statement";
proc print data=diabetes.pima;
run;
```

Display 17. Confirming working directory stored in a macro works

```
Notice no date which was the alloptions program that was being called by the
%include statement
                     Plasma                                  Diabetes
Obs  id  Pregnancies  Gluc   DBP   Triceps   Insulin    BMI  Pedigree  Age Class

1    1      6         148      72       35       0       33.6  0.627    50      1
2    2      1          85      66       29       0       26.6  0.351    31      0
3    3      8         183      64        0       0       23.3  0.672    32      1
4    4      1          89      66       23      94       28.1  0.167    21      0
5    5      0         137      40       35     168       43.1  2.288    33      1
6    6      5         116      74        0       0       25.6  0.201    30      0
7    7      3          78      50       32      88       31.0  0.248    26      1
8    8     10         115       0        0       0       35.3  0.134    29      0
9    9      2         197      70       45     543       30.5  0.158    53      1
10  10      8         125      96        0       0        0.0  0.232    54      1
```

Display 18. PROC PRINT output picks up the alloptions program which set the options to nodate

CONCLUSION

“Know Thy Data” has to be the most important rule – perhaps the *only* rule – for Data developers. Too often, SAS users ask “We know we should ‘Know Our Data’ – but we don’t. Can SAS help?” The goal in this presentation was to share the many ways in which PROC SQL can help to get to know your data. Ensure data quality and readiness for analysis by embracing PROC SQL’s dictionary tables. So you can satisfy the #1 programmer’s rule- Know thy data.

REFERENCES

Droogendyk, Harry. “QCYour SAS ® and RDBMS Data Using Dictionary Tables”. 18th Annual SouthEast SAS Users Group (SESUG) Conference Savannah, GA, September 26 – 28, 2010. http://analytics.ncsu.edu/sesug/2010/BB04.Droogendyk.pdf

Eberhardt, Peter & Brill, Irene. “How Do I Look it Up If I Cannot Spell It: An Introduction to SAS® Dictionary Tables”. SAS® Users Group International SUGI 31 San Francisco Proceedings, March 26-29, 2006. http://www2.sas.com/proceedings/sugi31/259-31.pdf

Go, Imelda C. “Reordering Variables in a SAS® Data Set”. 10th Annual SouthEast SAS Users Group (SESUG) Conference, Savannah, GA, September 22 – 24, 2002. http://analytics.ncsu.edu/sesug/2002/PS12.pdf#navpanes=0

Lafler, Kirk. “Exploring DICTIONARY Tables and Views”. SAS® Users Group International SUGI 30, Philadelphia, PA, April 10-13, 2005. http://www2.sas.com/proceedings/sugi30/070-30.pdf

Libeg, Linda. ”The SAS® Magical Dictionary Tour”. 19th Annual SouthEast SAS Users Group (SESUG) Conference, Alexandria, VA, October 23–25, 2011. http://analytics.ncsu.edu/sesug/2011/BB09.Libeg.pdf

Website Support.sas.com. “How to view DICTIONARY tables”. Available at http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a002300185.htm

ACKNOWLEDGMENTS

Charu is grateful to Pharmasug for accepting her paper for presentation. She originally co-authored this paper with Andy Kuligowski as an invited paper at SAS Global forum 2013. Charu has modified this paper for Pharmasug and included up to date references. She appreciates her manager Stephen Keelan and SAS Canada for the support and encouragement to share her SAS® and SQL knowledge. She is grateful to her many wonderful customers and students whose ongoing questions provided the impetus to research & share dictionary table techniques.

CONTACT INFORMATION

The author welcomes correspondence about this work. You can contact her at:

Charu Shankar
Senior Technical Training Specialist
SAS® Institute Inc.
280 King Street East
Toronto, ON M5A 1K7
Charu.Shankar@sas.com