Paper 061-2009

# Let SQL Write SQL Scripts for You – Counts Report

Christine Teng, Merck Research Labs, Merck & Co., Inc., Rahway, NJ

## ABSTRACT

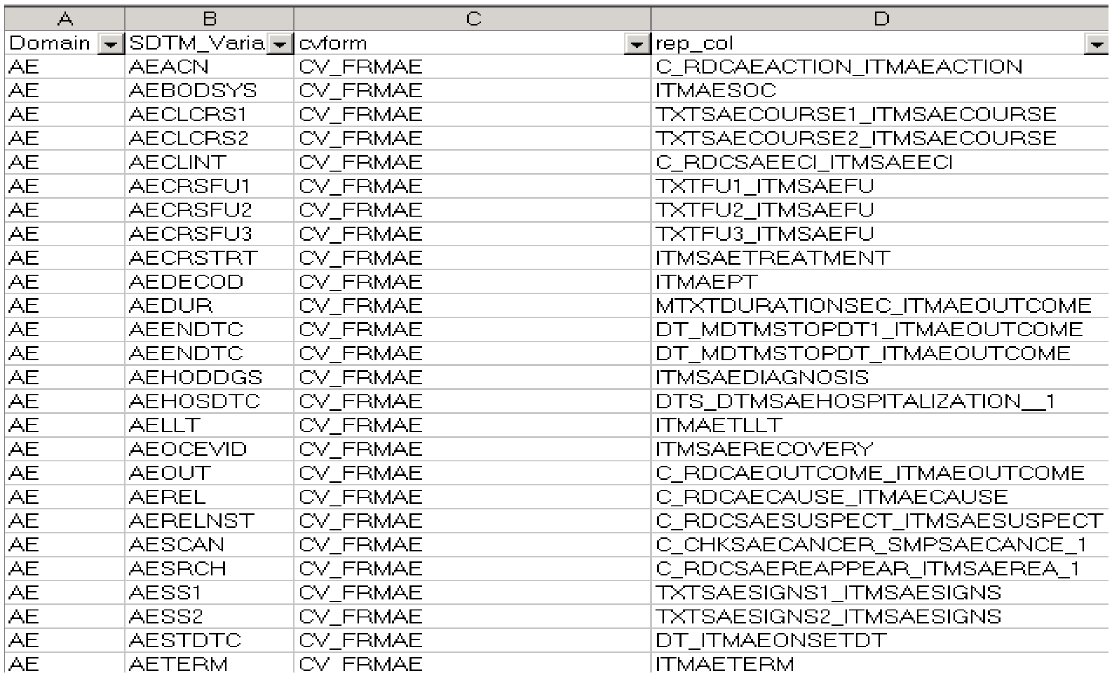
PROC SQL is a very powerful tool that can be used to easily build SQL scripts. Script-building can be used to produce useful detailed SAS dataset information. The output explained in this paper will produce a counts report which shows a list of variables with the total number of non-null values within its associated dataset between two data sources. The output consists of the following: dataset name, variable name, variable attributes and the count of non-null values.

SAS®9, Windows, Intermediate Level
Key Words: SAS DICTIONARY, PROC SQL, Macro

## APPLICATION

To illustrate the SQL script-building capability, a mapping application is used as an example (Table – 1). The main purpose of this application is to verify the mapping result between datasets. The details of the application will not be described here. The focus of this paper is to show how to utilize SQL to write SQL code so only details related to this topic will be described here. Table – 2, found below, is an Excel counts report produced from Table – 1 by comparing two datasets created from two different data sources. Column A represents the converted(target) dataset name and column B represents the variable name associated with the converted dataset; column C represents the source dataset name and column D represents the variable name associated with the source dataset.

**Table – 1**



| A | B | C | D |
|---|---|---|---|
| Domain | SDTM_Varia | cvform | rep_col |
| AE | AEACN | CV_FRMAE | C_RDCAEACTION_ITMAEACTION |
| AE | AEBODSYS | CV_FRMAE | ITMAESOC |
| AE | AECLCRS1 | CV_FRMAE | TXTSAECOURSE1_ITMSAECOURSE |
| AE | AECLCRS2 | CV_FRMAE | TXTSAECOURSE2_ITMSAECOURSE |
| AE | AECLINT | CV_FRMAE | C_RDCSAEECI_ITMSAEECI |
| AE | AECRSFU1 | CV_FRMAE | TXTFU1_ITMSAEFU |
| AE | AECRSFU2 | CV_FRMAE | TXTFU2_ITMSAEFU |
| AE | AECRSFU3 | CV_FRMAE | TXTFU3_ITMSAEFU |
| AE | AECRSTRT | CV_FRMAE | ITMSAETREATMENT |
| AE | AEDECOD | CV_FRMAE | ITMAEPT |
| AE | AEDUR | CV_FRMAE | MTXTDURATIONSEC_ITMAEOUTCOME |
| AE | AEENDTC | CV_FRMAE | DT_MDTMSTOPDT1_ITMAEOUTCOME |
| AE | AEENDTC | CV_FRMAE | DT_MDTMSTOPDT_ITMAEOUTCOME |
| AE | AEHODDGS | CV_FRMAE | ITMSAEDIAGNOSIS |
| AE | AEHOSDTC | CV_FRMAE | DTS_DTMSAEHOSPITALIZATION__1 |
| AE | AELLT | CV_FRMAE | ITMAETLLT |
| AE | AEOCEVID | CV_FRMAE | ITMSAERECOVERY |
| AE | AEOUT | CV_FRMAE | C_RDCAEOUTCOME_ITMAEOUTCOME |
| AE | AEREL | CV_FRMAE | C_RDCAECAUSE_ITMAECAUSE |
| AE | AERELNST | CV_FRMAE | C_RDCSAESUSPECT_ITMSAESUSPECT |
| AE | AESCAN | CV_FRMAE | C_CHKSAECANCER_SMPSAECANCE_1 |
| AE | AESRCH | CV_FRMAE | C_RDCSAEREAPPEAR_ITMSAEREA_1 |
| AE | AESS1 | CV_FRMAE | TXTSAESIGNS1_ITMSAESIGNS |
| AE | AESS2 | CV_FRMAE | TXTSAESIGNS2_ITMSAESIGNS |
| AE | AESTDTC | CV_FRMAE | DT_ITMAEONSETDT |
| AE | AETERM | CV_FRMAE | ITMAETERM |

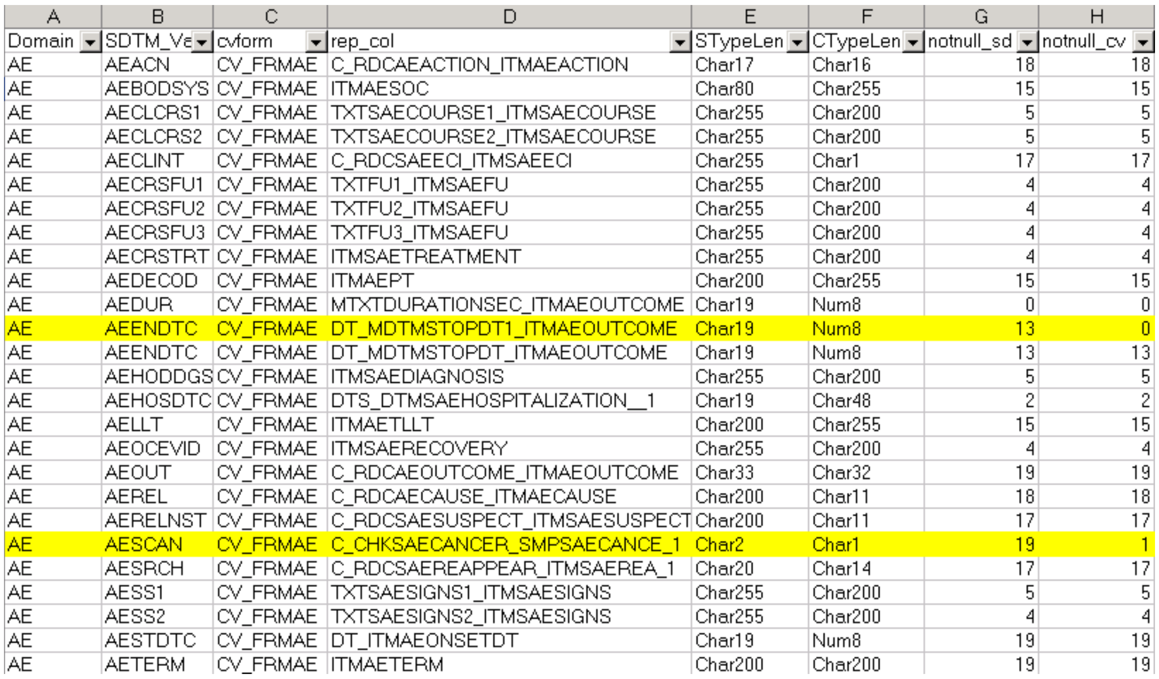
Column E, F, G and H in Table – 2 below are derived through SAS Macro programming. Columns E and F are derived from the SAS Dictionary.columns table. Column E represents the data attributes for the variables in column B and Column F represents the data attributes for the variables in column D.

Example code for formatting column E from a PROC SQL Select statement using Dictionary.columns table:

```
propcase(catx('',type,put(length, best5.))) as STypeLen
```

Further details for columns G and H are described in a later section below. Based on the variable attributes in the highlighted rows in Table – 2, there are count discrepancies that one would want to investigate further.

**Table – 2**



| A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|
| Domain | SDTM_Va | cvform | rep_col | STypeLen | CTypeLen | notnull_sd | notnull_cv |
| AE | AEACN | CV_FRMAE | C_RDCAEACTION_ITMAEACTION | Char17 | Char16 | 18 | 18 |
| AE | AEBODSYS | CV_FRMAE | ITMAESOC | Char80 | Char255 | 15 | 15 |
| AE | AECLCRS1 | CV_FRMAE | TXTSAECOURSE1_ITMSAECOURSE | Char255 | Char200 | 5 | 5 |
| AE | AECLCRS2 | CV_FRMAE | TXTSAECOURSE2_ITMSAECOURSE | Char255 | Char200 | 5 | 5 |
| AE | AECLINT | CV_FRMAE | C_RDCSAEECI_ITMSAEECI | Char255 | Char1 | 17 | 17 |
| AE | AECRSFU1 | CV_FRMAE | TXTFU1_ITMSAEFU | Char255 | Char200 | 4 | 4 |
| AE | AECRSFU2 | CV_FRMAE | TXTFU2_ITMSAEFU | Char255 | Char200 | 4 | 4 |
| AE | AECRSFU3 | CV_FRMAE | TXTFU3_ITMSAEFU | Char255 | Char200 | 4 | 4 |
| AE | AECRSTRT | CV_FRMAE | ITMSAETREATMENT | Char255 | Char200 | 4 | 4 |
| AE | AEDECOD | CV_FRMAE | ITMAEPT | Char200 | Char255 | 15 | 15 |
| AE | AEDUR | CV_FRMAE | MTXTDURATIONSEC_ITMAEOUTCOME | Char19 | Num8 | 0 | 0 |
| AE | AEENDTC | CV_FRMAE | DT_MDTMSTOPDT1_ITMAEOUTCOME | Char19 | Num8 | 13 | 0 |
| AE | AEENDTC | CV_FRMAE | DT_MDTMSTOPDT_ITMAEOUTCOME | Char19 | Num8 | 13 | 13 |
| AE | AEHODDGS | CV_FRMAE | ITMSAEDIAGNOSIS | Char255 | Char200 | 5 | 5 |
| AE | AEHOSDTC | CV_FRMAE | DTS_DTMSAEHOSPITALIZATION__1 | Char19 | Char48 | 2 | 2 |
| AE | AELLT | CV_FRMAE | ITMAETLLT | Char200 | Char255 | 15 | 15 |
| AE | AEOCEVID | CV_FRMAE | ITMSAERECOVERY | Char255 | Char200 | 4 | 4 |
| AE | AEOUT | CV_FRMAE | C_RDCAEOUTCOME_ITMAEOUTCOME | Char33 | Char32 | 19 | 19 |
| AE | AEREL | CV_FRMAE | C_RDCAECAUSE_ITMAECAUSE | Char200 | Char11 | 18 | 18 |
| AE | AERELNST | CV_FRMAE | C_RDCSAESUSPECT_ITMSAESUSPECT | Char200 | Char11 | 17 | 17 |
| AE | AESCAN | CV_FRMAE | C_CHKSAECANCER_SMPSAECANCE_1 | Char2 | Char1 | 19 | 1 |
| AE | AESRCH | CV_FRMAE | C_RDCSAEREAPPEAR_ITMSAEREA_1 | Char20 | Char14 | 17 | 17 |
| AE | AESS1 | CV_FRMAE | TXTSAESIGNS1_ITMSAESIGNS | Char255 | Char200 | 5 | 5 |
| AE | AESS2 | CV_FRMAE | TXTSAESIGNS2_ITMSAESIGNS | Char255 | Char200 | 4 | 4 |
| AE | AESTDTC | CV_FRMAE | DT_ITMAEONSETDT | Char19 | Num8 | 19 | 19 |
| AE | AETERM | CV_FRMAE | ITMAETERM | Char200 | Char200 | 19 | 19 |

## BASIC SQL CONCEPT USING MACRO VARIABLES

There are many features in PROC SQL. In this paper, only those features used in the example are addressed . The macro program described in this paper uses PROC SQL user-defined macro variables created by the "INTO" clause.

The syntax of the SELECT statement to create user-defined macro variables using the "INTO" clause and range '**-**' is depicted as follows:

**SELECT <**column name in a table**>**
**INTO** **:<**Macro Variable name1**>** **-**
**:<**Macro Variable name999**>**
**FROM <** table**>**

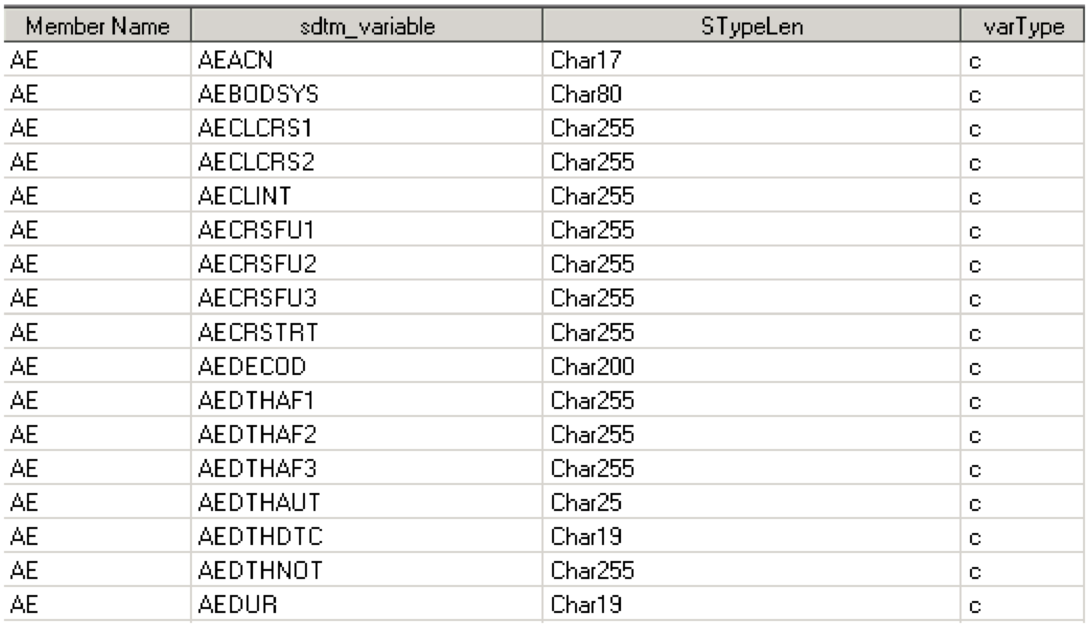
The SELECT statement above stores row values in a list of user-defined macro variables. Only the required number of macro variables will be created. A number large enough to hold the number of observations returned from the SELECT statement must be specified.

Another type of macro variable in PROC SQL is called an automatic macro variable; SQLOBS is used in the example. SQLOBS contains the number of rows or observations executed by a SQL statement.

## THE COUNTS REPORTS

SAS code from the count macro is found below. The dataset, sdtmtbl (Table – 3), contains information from the 'Domain' dataset where the attributes (Column E in Table – 2) have been obtained from the SAS dictionary table in a step not shown in this example. A separate table was created to hold the same information (Column F in Table – 2) for 'cvform' datasets. The code below describes what was done for the 'Domain' dataset. Similar code was completed using the same logic for the 'cvform' dataset which is not shown here.

**Table – 3**



| Member Name | sdtm_variable | STypeLen | varType |
|---|---|---|---|
| AE | AEACN | Char17 | c |
| AE | AEBODSYS | Char80 | c |
| AE | AECLCRS1 | Char255 | c |
| AE | AECLCRS2 | Char255 | c |
| AE | AECLINT | Char255 | c |
| AE | AECRSFU1 | Char255 | c |
| AE | AECRSFU2 | Char255 | c |
| AE | AECRSFU3 | Char255 | c |
| AE | AECRSTRT | Char255 | c |
| AE | AEDECOD | Char200 | c |
| AE | AEDTHAF1 | Char255 | c |
| AE | AEDTHAF2 | Char255 | c |
| AE | AEDTHAF3 | Char255 | c |
| AE | AEDTHAUT | Char25 | c |
| AE | AEDTHDTC | Char19 | c |
| AE | AEDTHNOT | Char255 | c |
| AE | AEDUR | Char19 | c |

```
/*--    Generate scripts to obtain counts for each variable within the domain dataset    -- */

proc sql noprint;
    create table TSD
    (ds_name char(30), sdtm_variable char(40), notnull_sd  num);     → A

    select  "select " || quote(trim(left(memname)))|| ',' || quote(trim(left(sdtm_variable))) || ", count(*)    → B
              from DATADIR." || trim(left(memname)) || " where " || trim(left(sdtm_variable)) || " ne ' ' ; " as ab
    into :sdsel1- :sdsel999
    from sdtmtbl
quit;

%let cntsc=&sqlobs;    → C

proc sql;
     %do i=1 %to &cntsc;
         insert into TSD         → D
             &&sdsel&i;
     %end;
quit;
```

**A.** The CREATE TABLE statement is used to create a table called TSD which holds the result of the variable count value for the Domain datasets. (Table – 4 below)

**B.** The SELECT statement reads from the SDTMTBL dataset (Table – 3) and builds SQL scripts that select the variable Column B from its corresponding domain. Each script created is stored in the user-defined macro variables :sdsel1- :sdsel999. For example, one of the macro variables will have a value as follows:

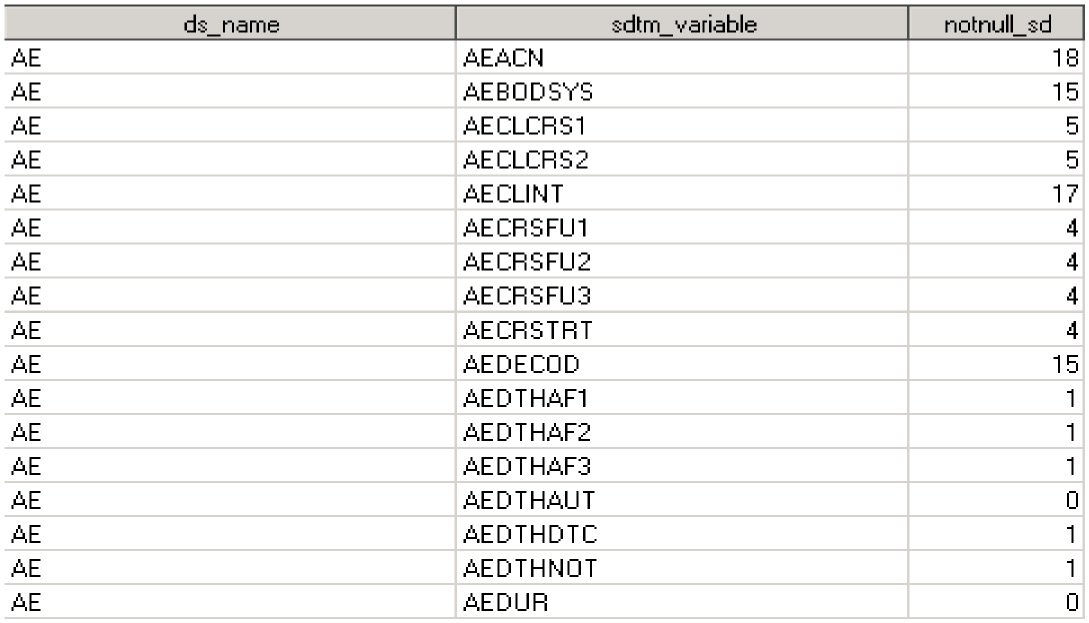
**select "AE","AEACN", count(*)  from DATADIR.AE where AEACN ne ' ';**

**C.** The SQLOBS macro variable contains the number of rows or observations executed by the SELECT statement that builds the script. The INSERT statement executes the number of times it is collected in SQLOBS. Please keep in mind that this automatic macro variable changes for each SQL run. It should be re-assigned to another user-defined macro variable to avoid getting a wrong value.

**D.** The INSERT statement is used to execute SELECT statements that are stored in the macro variables in Step B above. One iteration of the script executes as follows:

**insert into TSD**
**select "AE","AEACN", count(*)  from DATADIR.AE where AEACN ne ' ';**

**Table – 4**



| ds_name | sdtm_variable | notnull_sd |
|---|---|---|
| AE | AEACN | 18 |
| AE | AEBODSYS | 15 |
| AE | AECLCRS1 | 5 |
| AE | AECLCRS2 | 5 |
| AE | AECLINT | 17 |
| AE | AECRSFU1 | 4 |
| AE | AECRSFU2 | 4 |
| AE | AECRSFU3 | 4 |
| AE | AECRSTRT | 4 |
| AE | AEDECOD | 15 |
| AE | AEDTHAF1 | 1 |
| AE | AEDTHAF2 | 1 |
| AE | AEDTHAF3 | 1 |
| AE | AEDTHAUT | 0 |
| AE | AEDTHDTC | 1 |
| AE | AEDTHNOT | 1 |
| AE | AEDUR | 0 |

A similar process will be done to collect information from the cvform datasets. Once counts for variables from both data sources are collected, they are merged together with the mapping specification(Table – 1) to produce the output for Table – 2. The example here is to demonstrate how to construct SQL scripts in a PROC SQL statement. A simple way to get non-missing value can be replaced by N function .

## CONCLUSION

Using PROC SQL with macro variables can reduce the coding time for many data processing tasks. Using SQL to build SQL simplifies coding and minimizes coding errors.

## REFFERENCES

SAS SQL Procedure User's Guide

## ACKNOWLEGEMENTS

The author would like to thank her management team for their encouragement and review of this paper.

## TRADEMARKS

## CONTACT INFORMATION

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

Christine Teng
Merck & Co., Inc.
RY34-A320
P.O. Box 2000
Rahway, NJ 07065

christine_teng@merck.com

sas® Certified Advanced Programmer