PhUSE US Connect 2019

Paper CT14

Code Generator to Verify Implementation of Cut-off Date in the SDTM Domains

Christine Teng, Merck Research Laboratories, Merck & Co., Inc., Rahway, NJ USA

ABSTRACT

Data cut-off is a very common practice for interim-analysis in clinical trials. Data management cleans data up to the cut-off date before database lock. Correct implementation of data cut-off greatly impacts the analysis result and interpretations. The focus of this paper is to illustrate a simple script generator to show if the data cut-off is done according to the expected cut-off date for all date variables associated with each of the SDTM domain. Script-building approach is used to create similar repetitive statements which will be executed automatically without needing to specify the variable and domain names. The report generated from the script will include all date variable, label attribute and the variable's maximum date. This report is very useful to verify the proper cut-off date for the projects you manage in house or from CRO.

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

INTRODUCTION

The Study Data Tabulation Model Document (SDTM) defines a standard structure for study data tabulations that are to be submitted as part of a product application to a regulatory authority such as the United States Food and Drug Administration (FDA). SDTM variables are classified into five major roles which describe the type of information conveyed by the variable about each distinct observation. Some variables have specific suffix to help identify their purpose, for example, timing variables such as –DTC and –STDTC; topic variables such as –TEST and –TESTCD. The standard naming pattern helps simplify the programming effort. This paper illustrates the benefit of the standard naming using SAS PROC SQL programming techniques.

PROC SQL AND MACRO VARIABLES

The following statements show the syntax of the SELECT statement that create user-defined macro variables using the "INTO" clause and range syntax '**-**'.

```
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.

The syntax of the INSERT INTO statement comes in three forms as depicted below. We will use the "inserting rows with a query" approach in our example.

Inserting Rows with the SET Clause

```
INSERT INTO table-name | sas/access-view | proc-sql-view
<(column-1<, column-2, ...>)>
```

SET column1=sql-expression-1 <, column-2=sql-expression-2, ...> <column1=sql-expression-1 <, column-2=sql-expression-2, ...> ...>;
Inserting Rows with the VALUES Clause **INSERT INTO** table-name | sas/access-view | proc-sql-view <(column-1 <, column-2, ...>)> **VALUES** (value-1 <, value-2, ...>) <VALUES (value-1 <, value-2, ...>) ...>;
Inserting Rows with a Query **INSERT INTO** table-name | sas/access-view | proc-sql-view <(column-1 <, column-2, ...>)> query-expression;

Another type of macro variable in PROC SQL is called 'automatic macro variable'; in the example, we will use the SQLOBS automatic macro variable. SQLOBS contains the number of rows or observations executed by a SQL statement in prior step.

PROGRAMMING ILLUSTRATIONS

We want to show report with all date variables (suffix --DTC) and their maximum value associated with each of the SDTM domains specified in a library path. We can glance through the created report to see if there is a date value and if the date value goes beyond the cut-off date. The code snippets below from a macro describes what is done for each 'Domain' dataset without hard coding. Neither the domain name nor the domain date variable names are pre-specified.

Script	***Comment***
create table datetest as select memname, name, label from dictionary.columns where (name like '%dtc' or name like '%DTC') and libname="DATADIR" order by memname, name;	➢ Read from libname which is the path has the SDTM domain datasets for the study. ➢ From the data dictionary table only look for date variables (which should end with DTC per standard naming convention) ➢ Dataset datetest is created with SDTM domain name, date variable name and associated variable label
select memname, name, label from datetest	➢ Display the content of dataset "datetest" (see OUTPUT section)
select "select " || quote(trim(left(memname)))|| ',' || quote(trim(left(name))) || ", MAX(" ||(trim(left(name))) || ") from datadir." || trim(left(memname)) || " where " || trim(left(name)) || " ne ' ' ;" into :select1- :select999 from datetest ;	➢ Building script into each macro variable name : **select1** to **selectn** ➢ An example of macro variable &select1 has: **select "AE", "AEENDTC",** **max(AEENDTC)** **from datadir.AE where AEENDTC ne '';**
%let cnt=&sqlobs;	➢ The SQLOBS macro variable contains the number of rows or observations executed by the above SELECT statement that builds the script. The INSERT INTO statement in the following script executes the number of times based on 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.
create table T1	➢ Create a table T1 to hold value for displaying purpose for report

(ds_name char(5), date_var char(10), maxdtc char(20)); %do i=1 %to &cnt; insert into T1 &&select&i; %end;	➢ Now, execute each macro variable created earlier from each **select1** to **selectn** and the resulting value will be inserted into the table created in previous step ➢ An example of macro variable &select1 has: **Insert into T1** **select "AE", "AEENDTC",** **max(AEENDTC)** **from datadir.AE where AEENDTC ne '';**
select ds_name label='Domain', date_var label='Date Variable', maxdtc label='Max Date' from T1;	➢ This step simply displays the report (see OUTPUT section)

OUTPTUT

Dataset "datetest" (partial display of some domains)

Member Name	Column Name	Column Label
AE	AEENDTC	End Date/Time of Adverse Event
AE	AESTDTC	Start Date/Time of Adverse Event
BS	BSDTC	Date/Time of Collection
CM	CMENDTC	End Date/Time of Medication
CM	CMSTDTC	Start Date/Time of Medication
CO	CODTC	Date/Time of Comment
DD	DDDTC	Date/Time of Collection
DM	BRTHDTC	Date/Time of Birth
DM	DISCNDTC	Date/Time of Discontinuance
DM	DTHDTC	Date/Time of Death
DM	RFENDTC	Subject Reference End Date/Time
DM	RFICDTC	Date/Time of Informed Consent
DM	RFPENDTC	Date/Time of End of Participation
DM	RFSTDTC	Subject Reference Start Date/Time
DM	RFXENDTC	Date/Time of Last Study Treatment
DM	RFXSTDTC	Date/Time of First Study Treatment
DS	DSDTC	Date/Time of Collection
DS	DSSTDTC	Start Date/Time of Disposition Event
EG	EGDTC	Date/Time of ECG

For the demo study, we have cut-off date of 3/26/2018. Notice that the cut-off date was not applied on the DM domain in this example and one of the DM variable does not have any date value. If this is not the intended result, this report can be communicated to the programmer for corrections.

Domain	Date Variable	Max Date
AE	AEENDTC	2018-03-24
AE	AESTDTC	2018-03-26
BS	BSDTC	2018-03-26
CM	CMENDTC	2018-03-26
CM	CMSTDTC	2018-03-26

CO	CODTC	2018-03-26T00:00:00
DD	DDDTC	2018-03-24
DM	BRTHDTC	1997-02
DM	DISCNDTC	2018-03-27
DM	DTHDTC	2018-04-12
DM	RFENDTC	2018-03-27
DM	RFICDTC	
DM	RFPENDTC	2018-06-13
DM	RFSTDTC	2017-05-29T10:25
DM	RFXENDTC	2018-03-27
DM	RFXSTDTC	2017-05-29T10:25
DS	DSDTC	2018-03-26
DS	DSSTDTC	2018-03-26
EG	EGDTC	2018-02-06T10:22

CONCLUSION

Script-building approach can be implemented on many types of data reporting to check for any potential issues. In the example here, since SDTM implementation guide specifies standard naming conventions, it is very easy to identify common pattern without hard coding or knowledge of variable names. PROC SQL in combination with SQL data-dictionary tables and macro variables can reduce the coding time for many data processing tasks. Using SQL to generate SQL scripts simplifies coding logic hence minimize the reviewing time when managing multiple projects.

REFFERENCES

SAS SQL Procedure User's Guide
https://documentation.sas.com/?docsetId=sqlproc&docsetTarget=titlepage.htm&docsetVersion=9.4&locale=en

Study Data Tabulation Model v1 4
https://www.cdisc.org/system/files/all/standard_category/application/pdf/study_data_tabulation_model_v1_4.pdf

ACKNOWLEGEMENTS

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

CONTACT INFORMATION

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

Christine Teng
Merck Research Laboratories, Merck & Co., Inc.
Rahway, NJ 07065
christine_teng@merck.com

sas® Certified Advanced Programmer