Patient Profiles ‘On the Cheap’:
Quickly Capitalizing on PROC SQL and PROC REPORT to Efficiently and Cost Effectively Create Patient Profiles for Sponsor Source Data Verification, Audit, and Other Types of Clinical Data Verifications and Reviews
Mark Rothe, Roche, Nutley, NJ

ABSTRACT
As part of its compliance efforts, the FDA’s Office of Scientific Investigations (OSI) may request patient profiles to be produced for patients from select investigator sites whose data is part of an NDA (New Drug Application). These patient profiles are then used to verify data integrity against source documents during on-site inspections. If patient profiles are not already producible from the sponsor’s clinical database (especially likely if database management is outsourced and costs/resources are prohibitive or if the database were inherited without such capability), this paper illustrates some key BASE SAS® Version 8+ functionalities—available in PROC SQL and PROC REPORT on any operating system—that allow patient profiles to be quickly created to meet OSI’s needs. This same profile-generating methodology can be applied whenever clinical verification or review of patient data cannot be readily facilitated by other means.

INTRODUCTION
Electronic data submissions to the FDA have long supplanted paper patient profiles. However, the adjudication of the merits of drugs, biologics, and devices still involves data in the printed or static form. As recently as 2011, OSI requested patient profiles to check against clinical trial source documents during site inspections.

Generating true profiles from databases is no simple task and differs from printing completed eCRFs (electronic Case Report Forms): For every patient, data from each domain (e.g. medical history, concomitant medications, adverse events, etc.) needs to be temporally linked and clearly mapped onto a series of 8½ X 11 pages. Temporal linking across domains is possible but can be resource intensive—especially when implementing time windows for visit independent data. Clearly mapping data can be facilitated via DATA _NULL_—but precisely defining where it is located on the page is painstaking, especially with the sheer variety of patient data. The quicker alternative would be to display data by domain and to automatically map data onto the page. SAS provides three tools to create these patient profiles “on the cheap”:

1. PROC SQL DICTIONARY Tables
2. MACRO Variables & MACRO Functionalities
3. PROC REPORT NAMED and WRAP Options

PROC SQL DICTIONARY TABLES
SAS provides real time access to metadata in a variety of ways (e.g. PROC CONTENTS, PROC DATASETS, PROC SQL, etc.), and each method may lend itself better to solving different problems. For example, PROC CONTENTS is the tool of choice when one wants to quickly describe the variables contained in a data set. However, the read-only DICTIONARY tables accessible via PROC SQL provide instant data about a vast array of SAS constructs: catalogs, indexes, macros, titles, views, libraries, data sets, variables, external files, options, etc. Retrieving and storing the data set names and labels to be used for patient profiles can be accomplished via PROC SQL DICTIONARY TABLES:

PROC SQL;
    CREATE TABLE WORK.OSI_DATAINFO1 AS
    SELECT MEMNAME, MEMLABEL
    FROM DICTIONARY.TABLES
    WHERE (LIBNAME='NDA')
    ORDER BY MEMNAME;
QUIT;
Execution of this PROC SQL statement will create the SAS data set OSI_DATAINFO1 in the WORK library—storing the names of each of the SAS data sets in the NDA library in the variable MEMNAME and the corresponding data set labels in the variable MEMLABEL, sorted by MEMNAME in ascending order.

Figure 1.0 depicts example data sets to illustrate some of the metadata available in DICTIONARY.TABLES:

```
DATA WORK.DEMO
(LABEL='DEMOGRAPHY');
LENGTH PID $10.;
PID = '10001-101';
AGE = 34;
OUTPUT;
PID = '10001-102';
AGE = 57;
OUTPUT;
LABEL PID = 'PATIENT ID'
AGE = 'AGE';
RUN;

DATA WORK.AE
(LABEL='ADVERSE EVENTS');
LENGTH PID $10. AETXT $30.;
PID = '10001-101';
AETXT = 'HEADACHE';
OUTPUT;
PID = '10001-102';
AETXT = 'ABDOMINAL RASH';
OUTPUT;
LABEL PID = 'PATIENT ID'
AETXT = 'AE LITERAL';
RUN;
```

PROC SQL;
CREATE TABLE WORK.TABLE_METADATA AS
SELECT *
FROM DICTIONARY.TABLES
WHERE (LIBNAME='WORK')
ORDER BY MEMNAME;
QUIT;

Below are select columns/variables from WORK.TABLE_METADATA:

<table>
<thead>
<tr>
<th>Library Name</th>
<th>Member Name</th>
<th>Member Type</th>
<th>Date Set Label</th>
<th>Date Created</th>
<th>Number of Physical Observations</th>
<th>Observation Length</th>
<th>Number of Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK</td>
<td>AE</td>
<td>DATA</td>
<td>ADVERSE EVENTS</td>
<td>08AUG12:11:55:00</td>
<td>2</td>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>WORK</td>
<td>DEMO</td>
<td>DATA</td>
<td>DEMOGRAPHY</td>
<td>08AUG12:11:55:00</td>
<td>2</td>
<td>24</td>
<td>2</td>
</tr>
</tbody>
</table>

Some helpful columns/variables available in WORK.TABLE_METADATA are shown below via PROC CONTENTS:

```
Alphabetic List of Variables and Attributes

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Format</th>
<th>Informat</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>crdate</td>
<td>Num</td>
<td>8</td>
<td>DATETIME.</td>
<td>DATETIME.</td>
<td>Date Created</td>
</tr>
<tr>
<td>16</td>
<td>filesize</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
<td>Size of File</td>
</tr>
<tr>
<td>1</td>
<td>libname</td>
<td>Char</td>
<td>8</td>
<td></td>
<td></td>
<td>Library Name</td>
</tr>
<tr>
<td>23</td>
<td>maxlabel</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
<td>Longest label</td>
</tr>
<tr>
<td>22</td>
<td>maxvar</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
<td>Longest variable name</td>
</tr>
<tr>
<td>5</td>
<td>memlabel</td>
<td>Char</td>
<td>256</td>
<td>Data Set Label</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>memname</td>
<td>Char</td>
<td>32</td>
<td></td>
<td></td>
<td>Member Name</td>
</tr>
<tr>
<td>3</td>
<td>memtype</td>
<td>Char</td>
<td>8</td>
<td></td>
<td></td>
<td>Member Type</td>
</tr>
<tr>
<td>8</td>
<td>modate</td>
<td>Num</td>
<td>8</td>
<td>DATETIME.</td>
<td>DATETIME.</td>
<td>Date Modified</td>
</tr>
<tr>
<td>9</td>
<td>nobs</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
<td>Number of Physical Observations</td>
</tr>
<tr>
<td>40</td>
<td>num_character</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
<td>Number of Character Variables</td>
</tr>
<tr>
<td>41</td>
<td>num_numeric</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
<td>Number of Numeric Variables</td>
</tr>
<tr>
<td>11</td>
<td>nvar</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
<td>Number of Variables</td>
</tr>
<tr>
<td>10</td>
<td>obslen</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
<td>Observation Length</td>
</tr>
</tbody>
</table>
```
MACRO VARIABLES & MACRO FUNCTIONALITIES

Being able to capture metadata allows one to take advantage of MACRO variables and MACRO functionalities:

```sql
PROC SQL NOPRINT;

SELECT N(MEMNAME) INTO :N_OSI
FROM WORK.OSI_DATAINFO1;

SELECT MEMNAME INTO :OSI_D1--OSI_D%SYSEVALF(&N_OSI,INTEGER)
FROM WORK.OSI_DATAINFO1;

SELECT COMPBL(MEMLABEL) INTO :OSI_L1--OSI_L%SYSEVALF(&N_OSI,INTEGER)
FROM WORK.OSI_DATAINFO1;

QUIT;
```

- The 1st SELECT statement counts the number of MEMNAME values found in WORK.OSI_DATAINFO1 and places the total into the MACRO variable &N_OSI.
- The 2nd SELECT statement places each MEMNAME value into a different enumerated MACRO variable: &OSI_D1, &OSI_D2, ..., &OSI_D%SYSEVALF(&N_OSI,INTEGER). The special syntax for the last MACRO variable will resolve to the total number of SAS data sets in the NDA library without having to hardcode the total. This desired outcome is facilitated by %SYSEVALF, because it removes the inherent leading blanks that result when resolving &N_OSI (Prairie 2005, pp. 402-404). For example, if 17 data sets are in the NDA library, the last MACRO variable would be &OSI_D17.
- The 3rd SELECT statement does the same as the 2nd but for the data set labels.

Without %SYSEVALF (i.e. :OSI_D1--:OSI_D&N_OSI, :OSI_L1--:OSI_L&N_OSI), the PROC SQL code would generate the ERRORS as shown in Figure 2.0 (Prairie 2005, pp. 402-404).

---

**Fig. 2.0**

```
155   PROC SQL NOPRINT;
156
157   SELECT N(MEMNAME) INTO :N_OSI
158       FROM WORK.OSI_DATAINFO1;
159
160   SELECT MEMNAME INTO :OSI_D1--:OSI_D&N_OSI
161       FROM WORK.OSI_DATAINFO1;
162
163   SELECT COMPBL(MEMLABEL) INTO :OSI_L1--:OSI_L&N_OSI
164       FROM WORK.OSI_DATAINFO1;
```

---

Leading blanks when SAS resolves &N_OSI.

**ERROR 22-322**: Syntax error, expecting one of the following: ',', FROM, NOTRIM.

**ERROR 76-322**: Syntax error, statement will be ignored.
The MACRO variables created via PROC SQL can now be used to iterate through the SAS data sets and prepare their contents for mapping onto the page—selecting the relevant variables, choosing the order of the variables and records for display, etc. via storing the choices in MACRO variables—as illustrated in Figure 3.0.

Fig. 3.0

The order of the chosen variables in the OSI_SEL&I assignment statement will dictate their display order, because OSI_SEL&I will resolve as the SELECT statement in PROC SQL. The order of variables in SELECT statements determines the variable order in the resulting SAS data set.

Two ordering MACRO variables are created: The 1st for PROC SQL; the 2nd for PROC REPORT. The code would be more efficient if only one %LET statement were used with a MACRO employed to dynamically remove the commas when using OSI_ORD&I outside of PROC SQL, e.g. ...

PROC REPORT NAMED AND WRAP OPTIONS

Once mapping preparations are complete, the patient data from the sites slated for inspection can be transferred onto the page. Traditional PROC REPORT syntax would require SPLIT, COLUMN, DEFINE, DISPLAY, ORDER, WIDTH, FLOW, etc. to be properly defined for a successful summary table or data listing. However, by applying the NAMED and WRAP options, the PROC REPORT syntax is very succinct and straightforward—as can be seen in Figure 4.0.

Fig. 4.0

PROC REPORT DATA=WORK.AE NAMED NOWD MISSING WRAP;
RUN;

Produces the SAS output …

PATIENT ID=10001-101 AE VERBATIM TERM=HEADACHE
PATIENT ID=10001-102 AE VERBATIM TERM=ABDOMINAL RASH
• The NAMED option prints each variable name, an equal sign, and the variable's value. When the variables have labels, they are used in place of the variable names. This is very beneficial for ease of understanding the patient data—especially for the OSI inspector. Also, the NOHEADER option does not need to be specified, because PROC REPORT turns this option on by default with the NAMED option. [Use of NOHEADER prevents the display of column or variable headers.] (SAS Technical Report P-258 1993, p. 183-184)

• The NOWD or NOWINDOWS option prevents the REPORT window from opening up. The real time modifications afforded by the REPORT window would tremendously hinder patient profile generation due to the numerous patient and data set iterations. Furthermore, profile layout details are pre-specified and do not require additional changes. (SAS Technical Report P-258 1993, p. 187)

• The MISSING option is critical with GROUP, ORDER, or ACROSS to ensure missing values are considered valid values. Even though these are not used here, turning on the MISSING option is good programming practice. (SAS Technical Report P-258 1993, p. 183)

• Turning on the WRAP option is very important, because it forces lengthy row data to flow to the next line of the page—preventing it from spilling onto subsequent pages. Figure 5.0 demonstrates the WRAP option. (SAS Technical Report P-258 1993, pp. 183, 187, 225, 226)

```sql
DATA WORK.AE(LABEL='ADVERSE EVENTS');
LENGTH PID $10.
    AETXT AEPT AEBODSYS $30;
PID      = '10001-101';
AETXT    = 'HEADACHE';
AEPT     = 'HEADACHE';
AEBODSYS = 'CENTRAL NERVOUS SYSTEM';
AEBEGDT  = INPUT('05OCT2007',DATE9.);
AEENDDT  = INPUT('07OCT2007',DATE9.);
OUTPUT;

PID      = '10001-101';
AETXT    = 'UPSET STOMACH';
AEPT     = 'NAUSEA';
AEBODSYS = 'GASTROINTESTINAL';
AEBEGDT  = INPUT('11DEC2010',DATE9.);
AEENDDT  = .U;
OUTPUT;

FORMAT AEBEGDT AEENDDT DATE9.;
LABEL PID      = 'PATIENT ID'
    AETXT    = 'AE VERBATIM TERM'
    AEPT     = 'AE PREFERRED TERM'
    AEBODSYS = 'AE BODY SYSTEM'
    AEBEGDT  = 'AE BEGIN DATE'
    AEENDDT  = 'AE END DATE';
RUN;

PROC SQL;
CREATE TABLE WORK.AE_REPORT AS
    SELECT PID, AETXT, AEPT, AEBODSYS, AEBEGDT, AEENDDT
    FROM WORK.AE
    ORDER BY AEBEGDT, AEENDDT;
QUIT;
OPTIONS NOCENTER NODATE MISSING='';
TITLE1 'AE PROFILE';
PROC REPORT DATA=WORK.AE_REPORT
    NAMED NOWD MISSING WRAP;
    BY AEBEGDT;
RUN;
```

**Illustration of the Use of the PROC REPORT NAMED and WRAP Options**

A ➔ Specifies the order of AE variables in the profile.

B ➔ Left justifies the PROC REPORT BY line and removes the "- - - - - - - -" from the BY line.

C ➔ Prevents the OSI inspector from misunderstanding SAS numeric missing values.

D ➔ A new report is produced for each value of the BY variable. (SAS Technical Report P-258 1993, pp. 191-192) In this example, data associated with each AE begin date has a separate report (i.e. printed on a separate page). If there are too many BY variables, and/or the BY variables’ values are too lengthy, then the BY line is truncated—regardless of the WRAP option.
### Fig. 5.0 (Continued)

Results in the SAS output …

<table>
<thead>
<tr>
<th>AE PROFILE</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE BEGIN DATE=05OCT2007</td>
<td></td>
</tr>
<tr>
<td>PATIENT ID=10001-101 AE VERBATIM TERM=HEADACHE</td>
<td></td>
</tr>
<tr>
<td>AE PREFERRED TERM=HEADACHE AE BODY SYSTEM=CENTRAL NERVOUS SYSTEM</td>
<td></td>
</tr>
<tr>
<td>AE BEGIN DATE=05OCT2007 AE END DATE=07OCT2007</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AE PROFILE</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE BEGIN DATE=11DEC2010</td>
<td></td>
</tr>
<tr>
<td>PATIENT ID=10001-101 AE VERBATIM TERM=UPSET STOMACH</td>
<td></td>
</tr>
<tr>
<td>AE PREFERRED TERM=NAUSEA AE BODY SYSTEM=GASTROINTESTINAL</td>
<td></td>
</tr>
<tr>
<td>AE BEGIN DATE=11DEC2010 AE END DATE=</td>
<td></td>
</tr>
</tbody>
</table>

- The long row data flows to the next line as a result of using the WRAP option—instead of going onto the next page. Without this wrapping, the profile would be unnecessarily lengthened and difficult to use.

- If one were to use CENTER (the default option), the BY line would appear with “- - - - - - - - -”. The NOCENTER option allows more BY variables to be displayed without truncation—especially helpful for some data sets. BY variables are repeated in the profile body (e.g. AEBEGDT).

| PATIENT ID=10001-101 AE VERBATIM TERM=HEADACHE |
| AE PREFERRED TERM=HEADACHE AE BODY SYSTEM=CENTRAL NERVOUS SYSTEM |
| AE BEGIN DATE=05OCT2007 AE END DATE=07OCT2007 |

The extra space between some variables can neither be removed by TRIM and LEFT (e.g. AEPT = TRIM(LEFT(AEPT)) ) nor by applying a format such as $VARYING30. in the PROC SQL statement.

### PUTTING IT ALL TOGETHER

Given how PROC REPORT works with the NAMED and WRAP options, the programming necessary to complete the generation of patient profiles “on the cheap” is detailed in Figure 6.0.
%MACRO MREPORT;

/* SELECTING THE PATIENTS FROM THE SITES TO BE INSPECTED. */
PROC SQL NOPRINT;

CREATE TABLE WORK.OSI_PID AS
SELECT TRIM(LEFT(COMPRESS(PUT(CRTN,BEST.) || '-' || PUT(PT,BEST.)))) AS PID
LENGTH = 50,
TRIM(LEFT(COMPRESS(PUT(CRTN,BEST.) || '_' || PUT(PT,BEST.)))) AS FL_PID
LENGTH = 50
FROM NDA.DEMO
WHERE (CRTN IN (10001, 10002, 10003))
ORDER BY PID;

SELECT N(PID) INTO :N_PID
FROM WORK.OSI_PID;

SELECT PID INTO :PID1 - PID%SYSEVALF(&NPID, INTEGER)
FROM WORK.OSI_PID;

SELECT FL_PID INTO :FLPID1 - FLPID%SYSEVALF(&NPID, INTEGER)
FROM WORK.OSI_PID;

QUIT;

%DO J = 1 %TO &N_OSI;

/* SUBSETTING THE RELEVANT PATIENT DATA FROM EACH DATA SET. */
PROC SQL;
CREATE TABLE WORK.REP_&&OSI_D&J AS
SELECT TRIM(LEFT(COMPRESS(PUT(CRTN,BEST.) || '-' || PUT(PT,BEST.))))
AS PID
LENGTH = 50,
&&OSI_SEL&J
FROM WORK.&&OSI_D&J
ORDER BY &&OSI_ORD&J;
QUIT;

%END;

/* ITERATING THROUGH THE SUBSET PATIENTS. */
%DO X = 1 %TO &N_PID;

OPTIONS PAGENO=1;
PROC PRINTTO PRINT="<Subdirectory Path>/OSI_PRFL_&FLPID&X...TXT" NEW;
RUN;

/* ITERATING THROUGH THE DATA SETS. */
%DO Y = 1 %TO &N_OSI;

title1 "Sponsor: Drug Company XYZ";
title2 "Study No.: 12345";
title3 "IND Application Number: 54,321 / NDA Application Number: 6789";
title4 "Center (Site) Number: %LEFT(%TRIM(%COMPRES(%SCAN(&&PID&X,1,-))))";
title6 "&&OSI_L&Y Domain";
title8 "Patient Profile for &&PID&X [Center Number - Patient Number]";

...
CONCLUSIONS

BASE SAS Version 8+ functionalities exist that allow patient profiles to be generated “on the cheap”:

1. PROC SQL DICTIONARY Tables
2. MACRO Variables & MACRO Functionalities
3. PROC REPORT NAMED and WRAP Options

The resulting profiles can be used to not only satisfy OSI’s requirements for site inspections but can be applied whenever clinical verification or review of patient data cannot be readily accomplished by existing tools. Furthermore, if data domains are standardized (e.g. SDTM—Study Data Tabulation Model), then these three BASE SAS functionalities could be used to create a single profile-generating tool to be applied across studies.

Some of these same functionalities can also be adapted to solve a variety of other problems such as iteratively comparing data sets across different SAS libraries, storing denominators in MACRO variables for calculating summary statistics, generating user-defined, data dependent SAS LOG messages, etc.

Fig. 6.0 (Continued)

PROC REPORT DATA=WORK.REP_&&OSI_D&Y NAMED NOWD MISSING WRAP;
   WHERE (PID = "&&PID&X");
   BY &&OSI_BY&Y;
RUN;

/* PRINTING NULL REPORT INFORMATION FOR PATIENTS WITHOUT DATA. */
PROC SQL;
   CREATE TABLE WORK.__NO_DATA__ AS
      SELECT *
      FROM WORK.REP_&&OSI_D&Y
      WHERE (PID = "&&PID&X");
   QUIT;

DATA _NULL_; 
   FILE PRINT TITLES;
   IF (INPUT(SYMGET('SQLOBS'),BEST.) = 0) THEN DO;
      PUT ">>> NO <-- DATA FROM THE %UPCASE(&OSI_L&Y) DOMAIN EXISTS FOR PATIENT &PID&X";
   END;
   RUN;

PROC DATASETS LIBRARY=WORK MEMTYPE=DATA NOLIST;
   DELETE __NO_DATA__;
   QUIT;
%END;
PROC PRINTTO PRINT=PRINT;
   RUN;
%END;
%MEND MREPORT;

Use of SYMGET to capture the value of the automatically generated MACRO variable &SQLOBS determines whether or not 0 rows were processed by PROC SQL. (Prairie 2005, pp. 404-407)
REFERENCES


ACKNOWLEDGMENTS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

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

Mark Rothe
Pharma Development Biometrics—Statistical Programming and Analysis
Hoffmann-La Roche, Inc.
340 Kingsland Street—Building 1—Study 4D29
Nutley, New Jersey, USA 07110
Work Phone: +1.973.235.6362
Email: mark.rothe@roche.com