PhUSE 2007

Paper OC01

Having your cake and eating it too: extracting data and variable attributes from Oracle Clinical

John Hendrickx, Clinquest Europe, Oss, The Netherlands

ABSTRACT

Extracting data from Oracle Clinical into SAS is quite straightforward but getting the variable attributes – labels, formats, etc. – is not. This paper discusses two SAS macros for extracting the data together with the variable attributes.

INTRODUCTION

SAS/Access software lets SAS users extract data from a wide range of databases including Oracle Clinical. But the procedure is plain vanilla, variable attributes such as the label and (date, time, user-defined) format are not included, just the raw data. Oracle Clinical database does contain data on labels, formats, buried here and there in obscure meta-tables. This paper shows how to query Oracle Clinical meta-tables to obtain information on variable attributes, and assign these attributes while extracting the data. This requires a two-step procedure. Macro `%extract_fmt` creates a format library for the study. Once this is done, macro `%ocdata` can be used to extract the data and assign variable attributes.

EXTRACTING DATA FROM ORACLE CLINICAL

Extracting data from Oracle Clinical is a piece of cake, provided you're not too demanding. Here's a simple example:

```
proc sql;
       connect to oracle (user='xxxxxx' orapw='xxxxxx' path='xxxxxx') ;

       create table demo as
       select *
       from connection to ORACLE
       (select *
       from CLQ001$current.DEMO);

       disconnect from oracle ;
quit;
```

First, a connection to the Oracle database is defined. SQL commands using Oracle syntax is "passed through" to the oracle database in the parenthesized section. The results of this passed through syntax is selected vrom the Oracle connection using SAS PROC SQL syntax.

After running this example, a temporary dataset called "DEMO" will be created with the contents of the Oracle Clinical table DEMO. Not too difficult. But the variables in this dataset won't have labels, no user-defined formats will be assigned or made available. It will all be character strings and numbers.

%EXTRACT_FMT AND %OCDATA

For the purpose of this paper, let's assume that the SAS users *are* demanding. They want their data, but they want their labels too, and they certainly want their user-defined formats! They want to be able to extract them quickly and easily, either as SAS datasets or as SAS views. If they misspecify something, they want to be told so *nicely*, by an intelligible error message. `%extract_fmt` and `%ocdata` to the rescue!

USER DEFINED FORMATS

The macros for this paper are designed to extract the data from Oracle Clinical together with variable attributes such as labels and user defined formats. The first step is to extract the user-defined formats from Oracle Clinical and store them in a format catalog. These formats are in there, somewhere in the Oracle Clinical database, tucked away in meta-tables like DISCRETE_VALUES and DISCRETE_VALUE_GROUPS. Let's wheedle them out.

%extract_fmt does this using the PROC SQL 'pass-through' facility to query Oracle meta-tables and extract information on formats stored in Oracle for the study in question. %extract_fmt creates a SAS dataset with format information which is then transformed into a format catalog using PROC FORMAT with the CNTLIN option. The SAS output prints the formats in the library specified using PROC FORMAT with the FMTLIB option.

%extract_fmt creates both numeric and character formats (the character formats have the same name with a $ prefix). %extract_fmt also creates a numeric informat with the same name as the numeric format. This informat is used by %ocdata to transform character variables into numeric variables when extracting datasets. This is the main reason why %extract_fmt must be run prior to %ocdata.

%EXTRACT_FMT OPTIONS

Option	Purpose	Status
study	This must be the valid name of a study entered in the Oracle Clinical database. The parameter is not case sensitive, quotes are ignored.	Mandatory
library	A valid libname for the format catalog. Use library=work to create a temporary format catalog. Use library=library to create a permanent format catalog	Default library
show	Use show=no to prevent the contents of the format catalog from being printed	Default YES
rename	Rename invalid OC formats. Syntax: rename=old_fmt_name1=new_fmt_name1 old_fmt_name2=new_fmt_name2 old_fmt_name*n*=new_fmt_name*n*	Optional
v8_fmts	Use v8_fmts=yes to create a format catalog that is compatible with SAS version 8. Format names will be limited to 7 characters and a check will take place to determine whether some formats have the same name after truncation. In that case, the rename option should be used.	Default NO
user	The OC user name	Mandatory
orapw	The OC password	Mandatory
path	The OC path name	Mandatory
debug	Use debug=yes to print additional information and prevent temporary datasets from being deleted when macro terminates	Default NO

EXTRACTING THE DATA

Once the formats and informats have been stored in a format catalog, the data can be read and attributes can be assigned to variables. %ocdata uses the PROC SQL 'pass-through' facility to query Oracle meta-tables and extract information on SAS datasets, variables, labels and formats. %ocdata then reads the datasets and assigns the attributes to the variables.

Character variables for which a user-defined format is available are transformed into numeric variables and the format is assigned. This is done with the input function, using an informat created by %extract_fmt which maps character values to numeric ones. A character version of the variable with the suffix _orig is created as well.

DATE and TIME variables are also transformed during extraction. DATE variables are transformed from string to SAS date variables using the ANYDTDTE. informat and are assigned the date9. format. For TIME variables, the ANYDTTME. informat is used the hhmm. format is assigned. Here too, character versions of the variables are created with the suffix _orig.

%OCDATA OPTIONS

Option	Purpose	Status
study	This must be the valid name of a study entered in the Oracle Clinical database. The parameter is not case sensitive, quotes are ignored.	Mandatory
view_type	Name of the Oracle Clinical view type, e.g. current, test	Mandatory
libref	Name of the libref for the SAS datasets, e.g. work, input	Mandatory
out_type	Either TABLE or VIEW	Default TABLE
rename	This parameter must have the same value used in %extract_fmt	Optional
v8_fmts	This parameter must have the same value used in %extract_fmt	Default NO
convert	If convert=NO, then date and time variables and variables with a user-defined format are not converted	Default YES

Option	Purpose	Status
sas_data_names	If sas_data_names=YES, then the SAS_NAME specified in data_extract_views will used for the datasets	Default NO
sas_var_names	If sas_var_names=YES, then the SAS_NAME in template_columns for variable names	Default NO
user	The OC user name	Mandatory
orapw	The OC password	Mandatory
path	The OC path name	Mandatory
debug	Use debug=yes to print additional information and prevent temporary datasets from being deleted when macro terminates	Default NO

EXAMPLE

The %extract_fmt macro must always be used first in order to create the formats and informats that %ocdata depends on. Here's a simple example of its use. The %extract_fmt macro requires a valid study name for the library parameter and a valid libref that points to the location where the format catalog is to be stored. The Oracle Clinical login parameters are always required of course.

```
%extract_fmt(study=CLQ001,library=library,user=xxxx,orapw=xxxx,path=xxxx);
```

This will extract the formats from Oracle Clinical and store them in the format catalog 'library'. %extract_fmt will also print a list of the formats and informats it created.

The %extract_fmt macro will check whether any of the format names end with a number. In addition, if v8_fmt=yes was specified, %extract_fmt will also check whether truncating format names causes two or more formats to have the same name. In either case, an error message is printed to the log and %extract_fmt should be run again using the rename option. Here is an example of the usage of the rename option:

```
%extract_fmt(study=CLQ001,library=library,
      rename=YESNO_CDISC31=YNCDSC YESNO_CLQ=YNCLQ,
      user=xxxx,orapw=xxxx,path=xxxx);
```

Once a format catalog has been created for the study, we are ready to extract the data itself. The %ocdata macro requires a valid study name, a valid Oracle Clinical view type (e.g. test, current), and a valid libref for the location of the SAS datasets. For example:

```
%ocdata(study=CLQ001,view_type=CURRENT,libref=data,
      rename=YESNO_CDISC31=YNCDSC YESNO_CLQ=YNCLQ,
      user=xxxx,orapw=xxxx,path=xxxx);
```

This statement will extract the datasets for the CLQ001 study to the location defined by the libref 'data'. By default, the variables will use the SAS names defined in Oracle Clinical, which are limited to 8 characters. This can be overridden using the option sas_var_names=no. Likewise, the default is to use SAS dataset names defined in Oracle Clinical but this can be overridden using sas_data_names=no. If format names have been renamed in %extract_fmt, the same rename parameter must be defined in %ocdata. The default is to create SAS datasets but it also possible to create SAS views by specifying out_type=view. In the event you simply want to extract the data without assigning variable attributes, specify convert=no.

The macro starts by deriving the names of the datasets in the study from OC table data_extract_views. For each dataset, %ocdata queries OC meta tables to determine the attributes of each variable in the dataset. These attributes are then stored in a series of macro variables.

%ocdata then generates an expanded version of the PROC SQL pass-through statement at the beginning of this paper. Each variables is now specified in the select statement and variable attributes are assigned using SAS PROC SQL syntax. Variables with a date, time, or user-defined format are stored as character version with an '_orig' suffix and as a converted variable with an appropriate format.

Here's an example of the PROC SQL syntax that %ocdata would generate to create a dataset:

```
create TABLE DATA.DEMO as select
      AGE as AGE label="Age in AGEU at Reference Date/Time" format=2.0 ,
      BRTHDTC as BRTHDTC_ORIG label="Date/Time of Birth" format=$8. ,
      case
            when compress(BRTHDTC,'. ') = '' then .
            else input(BRTHDTC,ANYDTDTE.)
      end as BRTHDTC label="Date/Time of Birth" format=date9. ,
      DSSTAT as DSSTAT_ORIG label="Disposition Event Status"
                  format=$YNCDISC. length=15 ,
      case
```

```
                when compress(DSSTAT,'. ') = '' then .
                else input(DSSTAT,YNCDISC.)
        end as DSSTAT label="Disposition Event Status" format=YNCDISC. ,
        HEIGHT as HEIGHT label="Height" format=3.0 ,
        WEIGHT as WEIGHT label="Weight" format=5.1 ,
        PT as PT label="Patient" format=$10. ,
        STUDY as STUDY label="Clinical Study" format=$15. ,
        USUBJID as USUBJID label="Unique Subject Identifier" format=$26. ,
        VISIT_NUMBER as VISIT label="Visit" format=10.
from connection to ORACLE (
        select *
        from CLQ001$CURRENT.DEMO
);
```

The second 'select' statement in parentheses after 'connection to ORACLE' is passed through to the Oracle database just as in the first example. It selects all data from the Oracle Clinical table DEMO. The first select statement uses SAS PROC SQL syntax to select the data for the connection to Oracle while adding labels and other attributes.

BRTHDTC is an example of a date variable. First, a character version is saved as BRTHDTC_ORIG. Then a date variable BRTHDTC is derived and associated with format date9. DSSTAT is an example of a variable with a user-defined format. A formatted character version DSSTAT_ORIG is created, followed by a formatted numeric version under the name DSSTAT. Of course, this requires that format $YNCDISC. and informat YNCDISC. have been created in a format catalog by `%extract_fmt`.

KNOWN ISSUES

The `%extract_fmt` and `%ocdata` macros work wonderfully for the most part. One important limitation though is that `%ocdata` can't deal with partial dates or time values. Partial date/time values will become missing values, which is one of the reasons why it's important that the original character values are extracted as well.

Another minor shortcoming is that variables are extracted in alphabetical order rather than the order in which they were defined in Oracle Clinical. Actually, the variables are extraced in alphabetical order within the dataset variables and the key variables (pt, study, usubjid and visit_number) in the example above.

Of course, it's quite conceivable that there are many *unkown* issues, e.g. because the macros are dependent on certain conventions at our department of data management. It's important to validate the macros to verify that all datasets for the study have been extracted with all cases and variables accounted for and that transformed variables do correspond with their original counterparts.

CONCLUSION

These macros make it easy to extract data from an Oracle Clinical database with variable attributes properly assigned. They are fairly user friendly, parameters are not case sensitive and the macros will terminate with an error message for common cases of misspecification. My experience is that the macros work well, with the caveats mentioned in the above paragraph.

REFERENCES

Caroline Bahler. *Optimizing Data Extraction from Oracle® Tables*. Paper presented at PharmaSUG 2000.

Kyle McBride. *Automating the Documentation of Oracle Clinical Database Specifications.* Paper presented at PharmaSUG 2004.

Angela S. Ringelberg. *The behind the scenes of extracting SAS data sets from Oracle Clinical*. Paper presented at PharmaSUG 2003.

Oracle/PLSQL Topics – http://www.techonthenet.com/oracle/index.php

ACKNOWLEDGMENTS

These macros build forth on the PharmaSUG papers by Kyle McBride and that by Angela Ringelberg. Part of the code of Kyle McBride's `%oc_extract_defs` macro is used in `%ocdata`. `%ocdata` also uses certain utility functions by Roland Rashleigh-Berry which are available at http://www.datasavantconsulting.com/roland/sasautos.html.

CONTACT INFORMATION

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

John Hendrickx
Clnquest Europe
Rossinistraat 46
NL-5344AK Oss,
The Netherlands

Email: John.Hendrickx@clinquest.nl

APPENDIX A: THE %EXTRACT_FMT MACRO

```
/*
************************************************************************
Macro accompanying the PhUSE 2007 paper
Having your cake and eating it too: extracting data and variable attributes
from Oracle Clinical

By
John Hendrickx
Clnquest Europe
Rossinistraat 46
NL-5344AK Oss,
The Netherlands

Email: John.Hendrickx@clinquest.nl

For information on the use of this macro, please refer to the paper.

This macro is intended for use in conjunction with the %ocdata macro.

This code may be modified and used free of charge provided a reference to the
original author is maintained.

The %WORDS macros called by this macro was written
by Roland Rashleigh-Berry (rolandberry@hotmail.com)
These macros and further inforamtion on their use are available at
http://www.datasavantconsulting.com/roland/sasautos.html
************************************************************************
*/

%macro extract_fmt(study=,library=,rename=,v8_fmts=NO,
                user=ops$stat,orapw=,path=,show=Y,debug=NO);
        %local noprint allfmts allfmts2 elems rnms i;

        %if %length(&study) eq 0 %then %do;
                %put ERROR: No study specified;
                %goto exit_macro;
        %end;
        %if %length(&library) eq 0 %then %do;
                %put ERROR: No library specified;
                %goto exit_macro;
        %end;
        %let library=%upcase(%sysfunc(dequote(&library)));
        %if %sysfunc(libref(&library)) %then %do;
                %put ERROR: library &library not defined;
                %goto exit_macro;
        %end;

        %if %length(&rename) ne 0 %then %do;
                %let rename=%qupcase(&rename);
                %let rnms=%words(&rename,delim=%str( =));
                %if %eval(&rnms - (&rnms/2)*2) %then %do;
                        %put ERROR: rename statement was misspecified (&rename);
                        %goto exit_macro;
                %end;
        %end;

        %let show=%substr(%upcase(&show),1,1);
        %let debug=%substr(%upcase(&debug),1,1);
        %let v8_fmts=%substr(%upcase(&v8_fmts),1,1);

        %if &debug eq N %then %let noprint=noprint;

        proc sql &noprint;
                connect to oracle (user="%sysfunc(dequote(&user))"
                orapw="%sysfunc(dequote(&orapw))"
                path="%sysfunc(dequote(&path"))) ;
```

```
/* get clinical_study_id for study=CTOxxx */
select clinical_study_id
into :id
from connection to ORACLE
(select clinical_study_id,
    study
from CLINICAL_STUDIES)
where study="&study";

/* ids of formats used in study */
Create table __ex_dcmq as
(select * from connection to ORACLE
(select distinct discrete_val_grp_id as id,
    discrete_val_grp_subset_nm as subset,
    clinical_study_id as study
from DCM_questions)
where study = &id and
      id ^= . and
      subset ^= .) ;

/* format codes and labels read in 3 times:
   for numeric formats, character formats
   and numeric informats */
create table __ex_dv as
(select * from connection to ORACLE
(select distinct discrete_value_dvg_id as id,
    discrete_value_dvg_subset_nm as subset,
    to_char(display_sn) as strt,
    long_label_description as label,
              'N' as type
from discrete_values
union
select distinct discrete_value_dvg_id as id,
    discrete_value_dvg_subset_nm as subset,
              discrete_value_value as strt,
    long_label_description as label,
              'C' as type
from discrete_values
union
select distinct discrete_value_dvg_id as id,
    discrete_value_dvg_subset_nm as subset,
              discrete_value_value as strt,
    to_char(display_sn) as label,
              'I' as type
from discrete_values));

/* format names (fmtname) */
create table __ex_dvg as
(select * from connection to ORACLE
(select distinct discrete_value_grp_id as id,
    discrete_val_grp_subset_num as subset,
    name as fmtname
from DISCRETE_VALUE_GROUPS));

/* merge the three tables using id and subset as keys */
/* 'blank' as hlo to prevent character value "other"
   from being interpeted as "other values" */
create table __ex_fmt as
select  type,
              fmtname,
              dv.subset as subset,
              strt as start,
              trim(label) as label,
              ' ' as hlo
from __ex_dv dv, __ex_dvg dvg, __ex_dcmq dcmq
where dv.id = dvg.id and dvg.id = dcmq.id and
         dv.subset=dvg.subset and dvg.subset=dcmq.subset
order by type, fmtname, start, subset desc;

/* Get the number of format names before renaming and truncating */
select distinct fmtname
```

```
                into :allfmts separated by ' '
                from __ex_fmt;

                %let nfmts=%words(&allfmts);
                create table __ex_fmt2 as
                select type,
                case
                        %if %length(&rename) ne 0 %then %do i=1 %to &rnms %by 2;
                                when fmtname="%scan(&rename,&i,%str( =))" then
"%scan(&rename,%eval(&i+1),%str( =))"
                        %end;
                        %if &v8_fmts eq Y %then %do;
                                %*  redundant when statement added for when rename is not used;
                                when type='N' then substr(fmtname,1,7)
                                else substr(fmtname,1,7)
                        %end;
                        %else %do;
                                when type='N' then  fmtname
                                else fmtname
                        %end;
                end as fmtname,
                subset, start, label, hlo
                from __ex_fmt;

                select distinct fmtname
                into :allfmts2 separated by ' '
                from __ex_fmt2;

                %if %words(&allfmts) ne %words(&allfmts2) %then %do;
                        %put ERROR: Truncating/renaming format names resulted in an error;
                        %put ERROR: Before: &allfmts;
                        %put ERROR: After : &allfmts2;
                        quit;
                        %goto exit_macro;
                %end;

                /* check for invalid format names */
                %put format names before: &allfmts;
                %put format names after : &allfmts2;
                %let elems=%prxmatch(&allfmts2,/\d+$/);
                %if %length(&elems) ne 0 %then %do;
                        %put ERROR: format names ending with one or more digits:;
                        %put ERROR: &elems;
                        quit;
                        %goto exit_macro;
                %end;

                disconnect from oracle ;
        quit;

        /* view created by extract_fmt macro is sorted
        by type, fmtname, start, subset desc
        sort using nodupkey option to select only the highest
        subset value */
        proc sort data=__ex_fmt2 nodupkey out=__ex_fmtset;
        by type fmtname start;
        run;

        /* create the catalog */
        proc format library=&library cntlin=__ex_fmtset;
        run;

        %if &show eq Y %then %do;
                proc format library=&library fmtlib;
                run;
        %end;

        %if &debug EQ N %then %do;
                proc datasets library=work memtype=data nolist;
                        delete __ex:;
                quit;
```

```
        %end;

        %exit_macro:
%mend;

/*
Based on %rxmatch by Roland Rashleigh-Berry (rolandberry@hotmail.com)
*/
%macro prxmatch(list,prxpattern);
        %local prx i;
        %let prx=%qsysfunc(prxparse(&prxpattern));
        %do i=1 %to %words(&list);
                %if %sysfunc(prxmatch(&prx,%scan(&list,&i,%str( )))) %then %scan(&list,&i,%str(
));
        %end;
        %syscall prxfree(prx);
%mend;

/*
Macro WORDS
by Roland Rashleigh-Berry (rolandberry@hotmail.com)
These macros and further inforamtion on their use are available at
http://www.datasavantconsulting.com/roland/sasautos.html
*/

%macro words(str,delim=%str( ));
        %local i;
        %let i=1;
        %do %while(%length(%qscan(&str,&i,&delim)) GT 0);
                %let i=%eval(&i + 1);
        %end;
        %eval(&i - 1)
%mend;
```

APPENDIX B: THE %OCDATA MACRO

```
/*
********************************************************************************
Macro accompanying the PhUSE 2007 paper
Having your cake and eating it too: extracting data and variable attributes
from Oracle Clinical

By
John Hendrickx
Clnquest Europe
Rossinistraat 46
NL-5344AK Oss,
The Netherlands

Email: John.Hendrickx@clinquest.nl

For information on the use of this macro, please refer to the paper.

This macro is intended for use in conjunction with the %extract_fmt macro.

This code may be modified and used free of charge provided a reference to the
original author is maintained.

Note: Parts of this macro were taken from the %OC_EXTRACT_DEFS macro in
Kyle McBride's paper "Automating the Documentation of Oracle Clinical Database
Specifications", presented at PharmaSUG 2004

The %WORDS, %MATCH %NODUP and %VARLIST macros called by this macro were written
by Roland Rashleigh-Berry (rolandberry@hotmail.com)
These macros and further inforamtion on their use are available at
http://www.datasavantconsulting.com/roland/sasautos.html
********************************************************************************
*/

%macro ocdata(study=,view_type=,libref=,out_type=table,rename=,convert=YES,
              sas_data_names=YES, sas_var_names=YES,v8_fmts=NO,
              user=,orapw=,path=,debug=NO);
       %local noprint;

       %let convert=%substr(%upcase(&convert),1,1);

       %let sas_data_names=%substr(%upcase(&sas_data_names),1,1);
       %let sas_var_names=%substr(%upcase(&sas_var_names),1,1);
       %let v8_fmts=%substr(%upcase(&v8_fmts),1,1);

       %let debug=%substr(%upcase(&debug),1,1);
       %if &debug eq N %then %let noprint=noprint;

       %if %length(&study) eq 0 %then %do;
              %put ERROR: No study specified;
              %goto exit_macro;
       %end;
       %if %length(&view_type) eq 0 %then %do;
              %put ERROR: No view type specified;
              %goto exit_macro;
       %end;
       %let view_type=%upcase(%sysfunc(dequote(&view_type)));
       %if %length(&libref) eq 0 %then %let libref=WORK;
       %let libref=%upcase(%sysfunc(dequote(&libref)));
       %if %sysfunc(libref(&libref)) %then %do;
              %put ERROR: libref &libref not defined;
              %goto exit_macro;
       %end;

       %let out_type=%trim(%upcase(%sysfunc(dequote(&out_type))));
       %if &out_type ne TABLE & &out_type ne VIEW %then %do;
              %put ERROR: out_type must be either TABLE or VIEW;
              %goto exit_macro;
```

```
	%end;

	%if %length(&rename) ne 0 %then %do;
		%let rename=%qupcase(&rename);
		%let rnms=%words(&rename,delim=%str( =));
		%if %eval(&rnms - (&rnms/2)*2) %then %do;
			%put ERROR: rename statement was misspecified (&rename);
			%goto exit_macro;
		%end;
	%end;

	proc sql &noprint;
		connect to oracle (user="%sysfunc(dequote(&user))"
		orapw="%sysfunc(dequote(&orapw))"
		path="%sysfunc(dequote(&path))") ;

		/* extract clinical_study_id for this study as key
		   for subsequent selections */
		select clinical_study_id into :id
		from connection to ORACLE
		(select clinical_study_id,
		     study
		from CLINICAL_STUDIES)
		where study="&study";

		%if &sqlrc ne 0 %then %do;
			%put ERROR: Querying clinical_study_id for &study produced an SQL error
(&sqlrc);
			%goto exit;
		%end;
		%if %length(&id) eq 0 %then %do;
			%put ERROR: Macro variable %superq(&id) representing clinical_study_id not
created;
			%goto exit;
		%end;

		/* get tid and ktid (key variables) for each dataset of this study */
		%local tid_list ktid_list name_list sname_list;
		select
			view_template_id,
			key_template_id,
			name,
			sas_name into
			:tid_list   separated by ' ',
			:ktid_list  separated by ' ',
			:name_list  separated by ' ',
			:sname_list separated by ' '
		from connection to ORACLE
		(select unique
			view_template_id,
			key_template_id,
			name,
			sas_name
		from data_extract_views
		where clinical_study_id=&id);

		%if &sqlrc ne 0 %then %do;
			%put ERROR: Extracting template id%str(%')s produced an SQL error (&sqlrc);
			%goto exit;
		%end;

		%local ndata;
		%let ndata=&sqlobs;

		%put ndata=&ndata;

		/* just use the OC dataset names by default */
		%if &sas_data_names eq N %then %let sname_list=&name_list;

		%local i tid ktid name sname;
		%do i=1 %to &ndata;
```

```
                        %let   tid=%qscan(  &tid_list,&i,%str( ));
                        %let  ktid=%qscan( &ktid_list,&i,%str( ));
                        %let  name=%qscan( &name_list,&i,%str( ));
                        %let sname=%qscan(&sname_list,&i,%str( ));
                        /* if an external view was created then there is no information on
                           variable attributes in any of the meta tables
                           tid and ktid will be either periods '.' or blank
                           Present solution is to extract the data as is,
                           without labels or formats */
                        %if %length(%sysfunc(compress( &tid,'.'))) eq 0 |
                                %length(%sysfunc(compress(&ktid,'.'))) eq 0 %then %do;
                                create table &libref..&name as
                                (select * from connection to ORACLE
                                (select * from &study.$&view_type..&name));
                                %put;
                                %put NOTE: No variable attributes could be extracted for dataset
&libref..&name;
                                %put NOTE: Run the SAS program in the extract folder for this
dataset if available;
                                %put;
                        %end;
                        %else %do;
                                %get_data(&tid,&ktid,&name,&sname);
                        %end;
                %end;

                %exit:
                disconnect from oracle ;
        quit;

        %if &debug EQ N %then %do;
                proc datasets library=work memtype=data nolist;
                        delete __oc:;
                quit;
        %end;

        %exit_macro:
%mend;

%macro get_data(tid,ktid,name,sname);

        create table __oc_dat_info as
        select * from connection to ORACLE
        (select distinct
                to_number(Null) dcm_id,
                to_number(Null) dcm_que_dcm_subset_sn,
                &tid template_id,
                &ktid key_template_id,
                %str(%'&name%') view_name,
                tc.name oracle_name,
                tc.sas_name,
                tc.sas_label,
                'ORACLE_VARIABLE' attribute_name,
                em.data_type_code type,
                em.length,
                to_char(null) defined_format,
                em.sas_format sas_format
        from template_columns tc,
                 extract_macros em
        where tc.template_id = &ktid
        and tc.key_extract_macro_id = em.extract_macro_id

        UNION

        select distinct
                dcmq.dcm_id,
                dcmq.dcm_que_dcm_subset_sn,
                &tid template_id,
                &ktid key_template_id,
                %str(%'&name%') view_name,
                tc.name oracle_name,
```

```
                tc.sas_name,
                tc.sas_label,
                tc.attribute_name,
                dcmq.question_data_type_code type,
                dcmq.length,
                dvg.name defined_format,
                case
                        when tc.attribute_name='DVG_NUMBER'
                        then substr(dvg.name,1,8)||'.'
                        when dcmq.question_data_type_code = 'NUMBER'
                        then dcmq.length+least(1,dcmq.decimal_places) || '.' ||
                        dcmq.decimal_places
                        when dcmq.question_data_type_code in ('CHAR','DATE')
                        then '$'||dcmq.length||'.'
                        else to_char(Null)
                end as sas_format
        from template_columns tc,
                 view_template_questions vtq,
                 view_question_mappings vqm,
                 dcm_questions dcmq,
                 discrete_value_groups dvg,
                 dci_modules dm
        where tc.template_id = &tid
        and tc.template_question_id = vtq.view_template_question_id
        and vtq.view_template_question_id = vqm.parent_question_id
        and vtq.question_id = vqm.question_id
        and vqm.dcm_question_id = dcmq.dcm_question_id
        and vtq.occurrence_sn = dcmq.occurrence_sn
        and dcmq.discrete_val_grp_id = dvg.discrete_value_grp_id (+)
        and dcmq.discrete_val_grp_subset_nm=dvg.discrete_val_grp_subset_num (+)
        and dcmq.dcm_id = dm.dcm_id
        and dcmq.dcm_que_dcm_subset_sn = dm.dcm_subset_sn)
        having dcm_que_dcm_subset_sn =
                max(dcm_que_dcm_subset_sn) or
                dcm_que_dcm_subset_sn = .;

        %local i nvar;
        %let nvar=&sqlobs;
        select oracle_name,
                %if &sas_var_names eq N %then %do;
                        oracle_name as sas_name,
                %end;
                %else %do;
                        sas_name,
                %end;
                 sas_label,
                %if %length(&rename) ne 0 %then %do;
                        case defined_format
                        %do i=1 %to &rnms %by 2;
                                when "%scan(&rename,&i,%str( =))" then
"%scan(&rename,%eval(&i+1),%str( =))"
                        %end;
                                else defined_format
                        end as defined_format,
                %end;
                %else %do;
                        defined_format,
                %end;
                        sas_format,
                        type,
                        length
        into :var1-:var&nvar,
                 :svar1-:svar&nvar,
             :lab1-:lab&nvar,
                :fmt1-:fmt&nvar,
                :sasfmt1-:sasfmt&nvar,
                :type1-:type&nvar,
                :length1-:length&nvar
        from __oc_dat_info;

        %if %length(&nvar) eq 0 %then %do;
```

```
		%put ERROR: No variables found for TID=&tid, KTID=&KTID, name=&name, sname=&sname;
		%goto exit;
	%end;
	%if &nvar < 1 %then %do;
		%put ERROR: Zero variables found for TID=&tid, KTID=&KTID, name=&name,
sname=&sname;
		%goto exit;
	%end;
	%if &nvar > 100 %then %do;
		%put ERROR: More than 100 variables found for TID=&tid, KTID=&KTID, name=&name,
sname=&sname;
		%goto exit;
	%end;

	/* extract the dataset from OC without any formatting,
		check whether the variable names derived from OC are correct */

	create table __octmp as
	select *
	from connection to ORACLE
	(select *
	from &study.$&view_type..&name);

	%if &sqlrc ne 0 %then %do;
		%put ERROR: Table &study.$&view_type..&name could not be extracted;
		%goto exit;
	%end;

	%local j allvars;
	%do j=1 %to &nvar;
		%let allvars=&allvars &&var&j;
	%end;
	%if %words(%match(&allvars,%varlist(__octmp))) ne &nvar %then %do;
		%put WARNING: The OC tables did not produce the correct set of variables;
		%put WARNING: &sname: %varlist(__octmp);
		%put WARNING: derived: &allvars;
		%put WARNING: &libref..&sname will be created without attributes;
		create table &libref..&sname as select * from __octmp;
		%goto exit;
	%end;

	/* create the dataset with attributes */
	%local comma cfmt ifmt nfmt;

	create &out_type &libref..&sname as
	select
	%let comma=,;
	%do j=1 %to &nvar;
/*		%put <&&var&j> <&&type&j> <&&fmt&j> <&&sasfmt&j> ;*/
		%if &j eq &nvar %then %let comma=;
		%if &&type&j eq TIME & %substrn(&&sasfmt&j,1,1) eq $ & &convert eq Y %then %do;
			&&var&j as &&svar&j.._ORIG label="&&lab&j" format=$6. &comma
			case
				when compress(&&var&j,'. ') = '' then .
				else input(&&var&j,ANYDTTME.)
			end as &&svar&j label="&&lab&j" format=hhmm. &comma
		%end;
		%else %if &&type&j eq DATE & %substrn(&&sasfmt&j,1,1) eq $ & &convert eq Y %then
%do;
			&&var&j as &&svar&j.._ORIG label="&&lab&j" format=&&sasfmt&j &comma
			case
				when compress(&&var&j,'. ') = '' then .
				else input(&&var&j,ANYDTDTE.)
			end as &&svar&j label="&&lab&j" format=date9. &comma
		%end;
		%else %if %length(&&fmt&j) eq 0 & %length(&&sasfmt&j) ne 0 %then %do;
			/* no user defined format: */
			&&var&j as &&svar&j label="&&lab&j" format=&&sasfmt&j &comma
		%end;
		%else %if &&type&j eq CHAR & &convert eq Y %then %do;
			%if &v8_fmts eq Y %then %do;
```

```
                        %let cfmt=$%substrn(&&fmt&j,1,7).;
                        %let ifmt= %substrn(&&fmt&j,1,7).;
                        %let nfmt= %substrn(&&fmt&j,1,7).;
                %end;
                %else %do;
                        %let cfmt=$&&fmt&j...;
                        %let ifmt= &&fmt&j...;
                        %let nfmt= &&fmt&j...;
                %end;
                &&var&j as &&svar&j.._ORIG
                        label="&&lab&j" format=&cfmt length=&&length&j &comma
                case
                        when compress(&&var&j,'. ') = '' then .
                        else input(&&var&j,&ifmt)
                end as &&svar&j label="&&lab&j" format=&nfmt &comma
        %end;
        %else %if &&type&j eq NUMBER & &convert eq Y %then %do;
                &&var&j as &&svar&j label="&&lab&j" format=&nfmt &comma
        %end;
        %else %do;
                &&var&j as &&svar&j label="&&lab&j" &comma
        %end;
%end;
from connection to ORACLE
(select *
from &study.$&view_type..&name);

/*      disconnect from oracle ;*/
/*quit;*/

        %exit:
%mend;

%macro substrn(string, position, length);
        %if %length(&length) eq 0 %then %sysfunc(substrn(&string,&position));
        %else %sysfunc(substrn(&string,&position,&length));
%mend;

/*
Based on %rxmatch by Roland Rashleigh-Berry (rolandberry@hotmail.com)
*/
%macro prxmatch(list,prxpattern);
        %local prx i;
        %let prx=%qsysfunc(prxparse(&prxpattern));
        %do i=1 %to %words(&list);
                %if %sysfunc(prxmatch(&prx,%scan(&list,&i,%str( )))) %then %scan(&list,&i,%str(
));
        %end;
        %syscall prxfree(prx);
%mend;

/*
Macros WORDS MATCH NODUP and VARLIST
by Roland Rashleigh-Berry (rolandberry@hotmail.com)
These macros and further inforamtion on their use are available at
http://www.datasavantconsulting.com/roland/sasautos.html
*/

%macro words(str,delim=%str( ));
        %local i;
        %let i=1;
        %do %while(%length(%qscan(&str,&i,&delim)) GT 0);
                %let i=%eval(&i + 1);
        %end;
        %eval(&i - 1)
%mend;

%macro match(ref,list,nodup=yes,casesens=no,fixcase=no);
        %local error list2 nref nlist i j item match refitem;
        %let error=0;
```

```
	%global _nomatch_;
	%let _nomatch_=;

	%let nodup=%upcase(%substr(&nodup,1,1));
	%let casesens=%upcase(%substr(&casesens,1,1));
	%let fixcase=%upcase(%substr(&fixcase,1,1));

	%if "&nodup" EQ "Y" %then %let list2=%nodup(&list,casesens=&casesens);
	%else %let list2=&list;

	%let nref=%words(&ref);
	%let nlist=%words(&list2);

	%if not &nref %then %do;
		%put ERROR: (match) No elements in reference list;
		%let error=1;
	%end;

	%if not &nlist %then %do;
		%put ERROR: (match) No elements in list under test;
		%let error=1;
	%end;

	%if &error %then %goto error;

	%do i=1 %to &nlist;
		%let item=%scan(&list2,&i,%str( ));
		%let match=NO;
		%do j=1 %to &nref;
			%let refitem=%scan(&ref,&j,%str( ));
			%if "&casesens" EQ "N" %then %do;
				%if "%upcase(&item)" EQ "%upcase(&refitem)" %then %do;
					%let match=YES;
					%let j=&nref;
				%end;
			%end;
			%else %do;
				%if "&item" EQ "&refitem" %then %do;
					%let match=YES;
					%let j=&nref;
				%end;
			%end;
		%end;
		%if &match EQ YES %then %do;
			%if "&fixcase" EQ "N" %then &item;
			%else &refitem;
		%end;
		%else %let _nomatch_=&_nomatch_ &item;
	%end;

	%goto skip;
	%error: %put ERROR: (match) Leaving match macro due to error(s) listed.;
	%skip:
%mend;

%macro nodup(list,casesens=no);
	%local i j match item error;
	%let error=0;
	%if not %length(&casesens) %then %let casesens=no;
	%let casesens=%upcase(%substr(&casesens,1,1));

	%if not %index(YN,&casesens) %then %do;
		%put ERROR: (nodup) casesens must nbe set to yes or no;
		%let error=1;
	%end;

	%if &error %then %goto error;

	%do i=1 %to %words(&list);
		%let item=%scan(&list,&i,%str( ));
		%let match=NO;
```

```
                %if &i LT %words(&list) %then %do;
                        %do j=%eval(&i+1) %to %words(&list);
                                %if &casesens EQ Y %then %do;
                                        %if "&item" EQ "%scan(&list,&j,%str( ))" %then %let
match=YES;
                                %end;
                                %else %do;
                                        %if "%upcase(&item)" EQ "%upcase(%scan(&list,&j,%str( )))"
%then %let match=YES;
                                %end;
                        %end;
                %end;
                %if &match EQ NO %then &item;
        %end;

        %goto skip;
        %error: %put ERROR: (nodup) Leaving nodup macro due to error(s) listed.;
        %skip:
%mend;

%macro varlist(ds);
        %local dsid rc nvars i varlist;

        %let dsid=%sysfunc(open(&ds,is));
        %if &dsid EQ 0 %then %do;
                %put ERROR: (varlist) Dataset &ds not opened due to the following reason:;
                %put %sysfunc(sysmsg());
        %end;
        %else %do;
                %let nvars=%sysfunc(attrn(&dsid,nvars));
                %if &nvars LT 1 %then %put ERROR: (varlist) No variables in dataset &ds;
                %else %do;
                        %do i=1 %to &nvars;
                                %if %length(&varlist) EQ 0 %then %let
varlist=%sysfunc(varname(&dsid,&i));
                                %else %let varlist=&varlist %sysfunc(varname(&dsid,&i));
                        %end;
                  %end;
                %let rc=%sysfunc(close(&dsid));
        %end;
        &varlist
%mend;
```