**Sample SAS Code for EHR/EMR Project**

## Data Extraction/Transform/Load with Pass Through Facility

PROC SQL;

CONNECT to ORACLE ( USER = xxxxx PASSWORD = xxxxxxxxxxxxx PATH = billing);

CREATE TABLE january\_charge\_summary AS

SELECT \* FROM CONNECTION TO ORACLE

(SELECT TO\_NUMBER (patient\_number), SUM (charges) as charges

FROM charge\_detail

WHERE bill\_date BETWEEN TO\_DATE ('xx/xx/xxxx','MM/DD/YYYY') AND

TO\_DATE ('xx/xx/xxxx','MM/DD/YYYY')

GROUP BY TO\_NUMBER (patient\_number));

DISCONNECT FROM ORACLE;

QUIT;

## Data Cleaning

\*Use a SAS macro to ensure patient privacy and minimize potential risks for all de-identified patient data as well as to detect unusual outliers

%macro check\_masking (lib=, outdsn=);

proc contents data=&lib..\_all\_ out=memname (keep=memname) noprint;

run;

proc sort data=memname nodupkey;

by memnam

run;

data \_null\_;

set memname;

call symputx( 'mem'||left(\_n\_), memname);

call symputx( 'nmem', \_n\_);

run;

data maskname;

set \_null\_;

run;

%do j=1 %to &nmem; ;

%put \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* J=&j DATA =&&mem&j \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

proc contents data=&lib..&&mem&j out=varname (keep=name nobs) noprint;

run;

data \_null\_;

set varname;

call symputx( 'var'||left(\_n\_), name);

call symputx( 'nvar', strip(\_n\_));

call symputx( 'nobs', strip(nobs));

run;

data mask;

set \_null\_;

run;

%do i=1 %to &nvar; ;

%put \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* I=&i VAR=&&var&i \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

data temp1;

set &lib..&&mem&j ;

p=verify ( upcase(&&var&i), 'X .') ;

if p>0 then p=1;

keep p;

run;

proc means data=temp1 nway noprint;

var p;

output out=temp2 sum=sum;

run;

data temp3;

set temp2;

length result $200 ratio $50;

if sum=0 then result = "&&var&i = Null values -- missing or de-identified by X";

else result = "&&var&i = Valid values";

ratio =strip(sum)||' / '||strip("&nobs");

percent=sum\*100/&nobs;

format percent 6.1 ;

run;

data mask;

set mask temp3 ;

run;

%end;

%let table=%scan(&&mem&j, 2,.);

data mask;

set mask;

length table $50;

table =propcase("&&mem&j");

Variable = strip( scan(result, 1, '='));

Description= strip( scan(result, 2, '='));

run;

title "\*\*================ J=&j DATA=&&mem&j ================\*\*\*";

proc sort data=mask ;

by sum result ;

run;

proc print data=mask;

var Table variable description ratio percent;

run;

data maskname;

set maskname mask;

run;

%end;

data &outdsn;

set maskname;

run;

%mend;

%check\_masking (lib=xdata, outdsn=varchk);

\*Create a count and order the labs available per patient and per lab test type in order to identify those patients who only have one test available, regardless of whether it’s done prior to or post admission.

proc sort data=sample\_lab;

by visitidentifier resultcatalogname resultdtm;

run;

data sample\_lab\_order;

set sample\_lab;

/\*order the labs\*/

count +1;

by vistiidentifier resultcatalogname resultdtm;

if first.resultcatalogname then count=1;

run;

\*Next, evaluate the timeliness of these tests by creating the “timetolabs” variable and thereafter flag those results that occurred prior to admission. Also, prepare some variables to be used in the resultant dataset: one for the lab test’s name, one for the units of measurement associated with the lab tests, and a flag variable for each lab test.

data sample\_lab\_prep;

format timetolabs hour. ;

set sample\_lab\_order;

/\*calculate time to labs, identify those that precede admission\*/

timetolabs=(resultdtm-admitdtm);

if resultdtm>admitdtm then priorflag-0;

else priorflag=1;

/\*prepare variables for transposing\*/

resultcatname2 = translate(trim(resultcatalogname), ‘\_’, ‘ ‘);

resultcatnameuom = catx(‘\_’,translate(trim(resultcatalogname),’\_’,’ ‘), ‘uom’);

resultcatflag = catx(‘\_’,translate(trim(resultcatalogname),’\_’,’ ‘), ‘flag’);

run;

\*Extrapolate those with lab variables prior to admission and select their last lab test of each category available in order to identify the lab test closest to when the patient was admitted to the hospital.

/\*priorflag=1 are those who had completed labs prior to admission\*/

data prior\_labs;

set sample\_lab\_prep;

by visitidentifier resultcatalogname;

/\*output the lab test prior to admission if no lab test occurred post-admission\*/

if last.resultcatalogname and priorflag=1 then output;

run;

\*Extrapolate those with post-admission tests and select their first test value.

proc sort data=sample\_lab\_prep;

by visitidentifier resultcatalogname count;

run;

/\*output the first post-admission test\*/

data post\_labs;

set sample\_lab\_prep;

where priorflag=0;

by visitidentifier resultcatalogname;

if first.resultcatalogname and priorflag=0 then output;

run;

\*Merge those pre-admission and post-admission tests into one database to prepare for transposing

/\*merge pre-admission and post-admission labs\*/

data lab\_transpose;

merge prior\_labs post\_labs;

by visitidentifier resultcatalogname;

run;

\*Finally, transpose the datasets in order to reflect the correct lab tests with appropriate flags and units

proc transpose data=lab\_transpose out=transpose\_labs1;

by visitidentifier

id resultcatname2;

var actual;

run;

proc transpose data=lab\_transpose out=transpose\_labs2;

by visitidentifier

id resultcatnameuom;

var uom;

run;

proc transpose data=lab\_transpose out=transpose\_labs3;

by visitidentifier

id resultcatflag;

var priorflag;

run;

/\*merge the transposed items\*/

proc sql;

create table inter2.transpose\_labs as

select a.\*

,b.\*

,c.\*

from transpose\_labs1 (drop=\_name\_ \_label\_) a

inner join transpose\_labs2 b on a.visitidentifier=b.visitidentifier

inner join transpose\_labs3 c on a.visitidentifier=c.visitidentifier

order by visitidentifier;

quit;

\*Verify that no patient was lost.

proc sql;

create table count as

select unique(visitidentifier), count(unique(resultcatalogname)) as test\_count

from sample\_lab\_prep;

run;

proc sql;

create table count2 as

select unique(visitidentifier), count(unique(resultcatalogname)) as test\_count2

from lab\_transpose;

run;