A Dose of SAS® to Brighten Your Healthcare Blues

Gabriela Cantu[1], Christopher Klekar[1]

[1] Center for Clinical Effectiveness, Baylor Scott & White Health, Dallas, Texas

ABSTRACT

The development and adoption of electronic health records (EHR) is creating exciting times in healthcare research. Today, copious amounts of EHR and secondary data lie at the tips of the researcher's fingers and robust data management techniques are essential in transforming the data to maximize its potential. Data derived from the EHR is critical for grant applications, pilot studies and is becoming the main source of data for research projects. The role of the data manager becomes crucial to research projects utilizing EHR and secondary data in order to maximum it's analysis potential.

Data managers encounter roadblocks when dealing with healthcare data and using some SAS® tips and tricks on bulky, cumbersome datasets can transform them into clean and simple ones that are analysis ready. Dealing with longitudinal data, containing multiple observations per patient is the livelihood of researchers; however, poor restructuring and misidentification of the necessary data points can frustrate the analysis process. PROC TRANSPOSE is often the first PROC considered when a data manager is trying to restructure longitudinal data for analytics. In this paper examples are shown that detail how a FIRST. and LAST. function can overcome the limits of PROC TRANSPOSE and conditional IF/THEN statements within the DATA procedure, how a PROC SQL procedure can be used to eliminate unwanted records within a longitudinal dataset and how a PROC TRANSPOSE and MERGE procedure can be used to make comparisons between a first visit and follow-up visits. This paper attempts to answer the age old question of "What is a researcher to do, when needing to look down longitudinal data and manipulate it?" Have no fear, utilization of PROC SQL and a LEFT JOIN in a novel way will save the day…and your sanity.

INTRODUCTION

EHR is a compilation of patient health information produced during one or more visits to a healthcare site. In the healthcare industry the terms EHR, Personal Health Records (PHR), Electronic Patient Records (EPR) and Electronic Medical Record (EMR) are used interchangeably. EHR data can be complex and integrate multiple data elements including demographic information, patient vitals, blood work and laboratory data, physical exam information, medications, medical history, current procedural data, and billing information. The information may be stored using an Oracle® or a Microsoft® SQL Server relational database with the information organized into a series of independent tables that are inter-connected through unique identification keys.[1]

While the structure of EHR data is relevant to patient clinical management, the organization of the data is often less than ideal for use by analyst for research and quality reporting purposes. To be successful, data managers must be efficient in identifying methods to organize and restructure data and to assure its usefulness in analysis. Methods traditionally used and initially thought of when trying to accomplish data transformations can prove to be limiting and cumbersome when facing EHR data. Data managers are challenged to "think outside" the proverbially SAS® box when encountering these data types.

This paper will focus on identifying some of the non-traditional methods used by our team to restructure this valuable but complex data so that it can be used for healthcare research projects and quality reporting.

FIRST. /LAST. EXAMPLE:

Data managers/programmers for research and quality projects in a healthcare setting spend much of their time identifying the correct "who" and correct "what." In this example, the right lab test for all patients needs to be identified. More specifically, the lab test needs to be the first done post-admission. If the patient doesn't have a post-admission blood test, then the first instance prior to admission needs to be kept and those records flagged.

Knowing that this is a multi-record per patient dataset, the first inclination is to use PROC TRANSPOSE. Because SAS® only adds/subtracts things on the same horizontal line; transposing each of the lab test dates would be a reasonable first step. After that subtract the time from admission, and then use an array to find the correct record. This methodology would require a lot of transposes, since there are over 15 different types of lab tests and no limit to the number of each kind of test a patient has performed in one day. Additionally, due to the complexity there are several places to make an error. In this situation, take a step back and evaluate the process. Thinking outside the box, it would be simpler plan to first break up the data sets. Do all the records really need to be kept together?

1) Create a count and order the labs available per patient and per lab test type. This will give identify those patients who only have one test available, regardless of whether it's done prior to or post admission.

```
proc sort data=example_lab;
by visitidentifier  resultcatalogname  resultdtm ;
run;

data example_lab_order ;
set example_lab;
/*ordering the labs*/
  count + 1;
  by visitidentifier resultcatalogname resultdtm;
  if first.resultcatalogname then count = 1;
run;
```

	visitidentifier	AdmitDtm	DischargeDtm	actual	ResultDTM	ResultCatalogName	UOM	count
1	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000		05JAN2010:17:5...	BF Glucose Source	Unknown	1
2	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	27.3	31DEC2009:17:2...	Bicarbonate Calculated	mmol/L	1
3	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	8.8	31DEC2009:17:3...	Calcium Blood Level	mg/dL	1
4	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	9.0	01JAN2010:07:4...	Calcium Blood Level	mg/dL	2
5	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	8.5	02JAN2010:06:3...	Calcium Blood Level	mg/dL	3
6	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	8.6	03JAN2010:05:2...	Calcium Blood Level	mg/dL	4
7	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	8.5	04JAN2010:05:5...	Calcium Blood Level	mg/dL	5
8	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	8.4	05JAN2010:05:1...	Calcium Blood Level	mg/dL	6
9	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	8.5	06JAN2010:04:1...	Calcium Blood Level	mg/dL	7
10	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	8.4	07JAN2010:06:1...	Calcium Blood Level	mg/dL	8
11	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	8.9	08JAN2010:05:2...	Calcium Blood Level	mg/dL	9
12	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	8.9	09JAN2010:05:3...	Calcium Blood Level	mg/dL	10
13	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	9.0	10JAN2010:05:5...	Calcium Blood Level	mg/dL	11
14	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	8.9	11JAN2010:06:3...	Calcium Blood Level	mg/dL	12
15	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	9.1	12JAN2010:05:0...	Calcium Blood Level	mg/dL	13
16	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	9.0	13JAN2010:04:2...	Calcium Blood Level	mg/dL	14
17	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	2.5	31DEC2009:17:3...	Creatinine	mg/dL	1
18	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	2.3	01JAN2010:07:4...	Creatinine	mg/dL	2
19	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	2.0	02JAN2010:06:3...	Creatinine	mg/dL	3
20	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	1.7	03JAN2010:05:2...	Creatinine	mg/dL	4
21	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	1.7	04JAN2010:05:5...	Creatinine	mg/dL	5
22	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	1.9	05JAN2010:05:1...	Creatinine	mg/dL	6
23	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	1.8	06JAN2010:04:1...	Creatinine	mg/dL	7
24	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	2.0	07JAN2010:06:1...	Creatinine	mg/dL	8
25	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	2.3	08JAN2010:05:2...	Creatinine	mg/dL	9
26	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	2.3	09JAN2010:05:3...	Creatinine	mg/dL	10
27	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	2.2	10JAN2010:05:5...	Creatinine	mg/dL	11
28	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	2.3	11JAN2010:06:3...	Creatinine	mg/dL	12
29	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	2.3	12JAN2010:05:0...	Creatinine	mg/dL	13
30	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	2.2	13JAN2010:04:2...	Creatinine	mg/dL	14
31	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	145	31DEC2009:17:3...	Glucose Blood Random	mg/dL	1
32	100233332	31DEC2009:20:4...	13JAN2010:16:45:00.000	95	01JAN2010:07:4...	Glucose Blood Random	mg/dL	2

2) Next, prep the data to evaluate the timeliness of these tests by creating the "timetolabs" variable then flag those results that occurred prior to admission (the bad ones!). Also, prep 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 lastly a flag variable for each lab test.

```
data example_lab_prep;
format timetolabs hour. ;
set example_lab_order ;
/*calculating time to labs, seeing which preceeds admission*/
  timetolabs=(resultdtm-admitdtm);
  if resultdtm>=admitdtm then priorflag=0; else priorflag=1;
  /*getting the var nams ready for transposing*/
resultcatname2= translate(trim(resultcatalogname), '_',' ');
resultcatnameuom=catx('_',translate(trim(resultcatalogname), '_',' '),'uom');
resultcatflag= catx('_',translate(trim(resultcatalogname), '_',' '),'flag');
run;
```

	visitidentifier	timetolabs	AdmitDtm	DischargeDtm	actual	ResultDTM	ResultCatalogName	UOM	count	priorflag	resultcatname2	resultcatnameuom	resultcatflag
1	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…		05JAN2010:17:5…	BF Glucose Source	Unknown	1	0	BF_Glucose_Sour…	BF_Glucose_Source_…	BF_Glucose_So…
2	100233332	3	31DEC2009:20:4…	13JAN2010:16:4…	27.3	31DEC2009:17:2…	Bicarbonate Calculated	mmol/L	1	1	Bicarbonate_Calcu…	Bicarbonate_Calculat…	Bicarbonate_Cal…
3	100233332	3	31DEC2009:20:4…	13JAN2010:16:4…	8.8	31DEC2009:17:3…	Calcium Blood Level	mg/dL	1	1	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
4	100233332	11	31DEC2009:20:4…	13JAN2010:16:4…	9.0	01JAN2010:07:4…	Calcium Blood Level	mg/dL	2	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
5	100233332	34	31DEC2009:20:4…	13JAN2010:16:4…	8.5	02JAN2010:06:3…	Calcium Blood Level	mg/dL	3	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
6	100233332	57	31DEC2009:20:4…	13JAN2010:16:4…	8.6	03JAN2010:05:2…	Calcium Blood Level	mg/dL	4	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
7	100233332	81	31DEC2009:20:4…	13JAN2010:16:4…	8.5	04JAN2010:05:5…	Calcium Blood Level	mg/dL	5	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
8	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	8.4	05JAN2010:05:1…	Calcium Blood Level	mg/dL	6	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
9	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	8.5	06JAN2010:04:1…	Calcium Blood Level	mg/dL	7	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
10	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	8.4	07JAN2010:06:1…	Calcium Blood Level	mg/dL	8	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
11	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	8.9	08JAN2010:05:2…	Calcium Blood Level	mg/dL	9	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
12	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	8.9	09JAN2010:05:3…	Calcium Blood Level	mg/dL	10	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
13	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	9.0	10JAN2010:05:5…	Calcium Blood Level	mg/dL	11	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
14	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	8.9	11JAN2010:06:3…	Calcium Blood Level	mg/dL	12	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
15	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	9.1	12JAN2010:05:0…	Calcium Blood Level	mg/dL	13	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
16	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	9.0	13JAN2010:04:2…	Calcium Blood Level	mg/dL	14	0	Calcium_Blood_Le…	Calcium_Blood_Level…	Calcium_Blood_…
17	100233332	3	31DEC2009:20:4…	13JAN2010:16:4…	2.5	31DEC2009:17:3…	Creatinine	mg/dL	1	1	Creatinine	Creatinine_uom	Creatinine_flag
18	100233332	11	31DEC2009:20:4…	13JAN2010:16:4…	2.3	01JAN2010:07:4…	Creatinine	mg/dL	2	0	Creatinine	Creatinine_uom	Creatinine_flag
19	100233332	34	31DEC2009:20:4…	13JAN2010:16:4…	2.0	02JAN2010:06:3…	Creatinine	mg/dL	3	0	Creatinine	Creatinine_uom	Creatinine_flag
20	100233332	57	31DEC2009:20:4…	13JAN2010:16:4…	1.7	03JAN2010:05:2…	Creatinine	mg/dL	4	0	Creatinine	Creatinine_uom	Creatinine_flag
21	100233332	81	31DEC2009:20:4…	13JAN2010:16:4…	1.7	04JAN2010:05:5…	Creatinine	mg/dL	5	0	Creatinine	Creatinine_uom	Creatinine_flag
22	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	1.9	05JAN2010:05:1…	Creatinine	mg/dL	6	0	Creatinine	Creatinine_uom	Creatinine_flag
23	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	1.8	06JAN2010:04:1…	Creatinine	mg/dL	7	0	Creatinine	Creatinine_uom	Creatinine_flag
24	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	2.0	07JAN2010:06:1…	Creatinine	mg/dL	8	0	Creatinine	Creatinine_uom	Creatinine_flag
25	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	2.3	08JAN2010:05:2…	Creatinine	mg/dL	9	0	Creatinine	Creatinine_uom	Creatinine_flag
26	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	2.3	09JAN2010:05:3…	Creatinine	mg/dL	10	0	Creatinine	Creatinine_uom	Creatinine_flag
27	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	2.2	10JAN2010:05:5…	Creatinine	mg/dL	11	0	Creatinine	Creatinine_uom	Creatinine_flag
28	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	2.3	11JAN2010:06:3…	Creatinine	mg/dL	12	0	Creatinine	Creatinine_uom	Creatinine_flag
29	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	2.3	12JAN2010:05:0…	Creatinine	mg/dL	13	0	Creatinine	Creatinine_uom	Creatinine_flag
30	100233332	**	31DEC2009:20:4…	13JAN2010:16:4…	2.2	13JAN2010:04:2…	Creatinine	mg/dL	14	0	Creatinine	Creatinine_uom	Creatinine_flag
31	100233332	3	31DEC2009:20:4…	13JAN2010:16:4…	145	31DEC2009:17:3…	Glucose Blood Random	mg/dL	1	1	Glucose_Blood_R…	Glucose_Blood_Rand…	Glucose_Blood_…
32	100233332	11	31DEC2009:20:4…	13JAN2010:16:4…	95	01JAN2010:07:4…	Glucose Blood Random	mg/dL	2	0	Glucose_Blood_R…	Glucose_Blood_Rand…	Glucose_Blood_…
33	100233332	34	31DEC2009:20:4…	13JAN2010:16:4…	125	02JAN2010:06:3…	Glucose Blood Random	mg/dL	3	0	Glucose_Blood_R…	Glucose_Blood_Rand…	Glucose_Blood_…

3) Pull out just those with lab variables prior to admission and select their last lab test of each category available. This will identify the lab test closest to when the patient was admitted to the hospital.

```
/*priorflag=1 are those who only have labs done prior to admission*/
data prior_labs;
set example_lab_prep;
by visitidentifier resultcatalogname;
/*outputting the lab test before admission, if no lab test occurred post admission*/
if last.resultcatalogname and priorflag=1 then output;
run;
/*3122*/
```

4) Pulling out just those with post admission tests and selecting their first test value. By switching from the LAST. function to the FIRST. function, this provides the lab test closest to admission in this dataset.

```
proc sort data=example_lab_prep;
by visitidentifier resultcatalogname count;
run;

/*outputting the first post-admit test*/
data after_labs;
set example_lab_prep;
where priorflag=0;
by visitidentifier resultcatalogname;
if first.resultcatalogname and priorflag=0 then output;
run;/*66273*/
```

5) Merge those pre-admission and post-admission tests into one database ready to be transposed:

```
/*merging the pre-admit and post-admit labs*/
data lab_fortranspose;
merge prior_labs after_labs;
by visitidentifier resultcatalogname;
run;/*69395*/
```

6) Finally you can transpose the datasets and get them merge back to reflect the right lab tests, with appropriate flags and units.

```
proc transpose data= lab_fortranspose out=transpose_labs1 ;
    by visitidentifier;
    id resultcatname2 ;
    var actual;
run;/*5404*/
proc transpose data= lab_fortranspose out=transpose_labs2;
    by visitidentifier;
    id resultcatnameuom ;
    var uom;
run; /*5404*/
proc transpose data= lab_fortranspose out=transpose_labs3 ;
    by visitidentifier;
    id resultcatflag;
    var priorflag;
run; /*5404*/
/*merging the tranposed 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;/*5404*/
```

Breaking apart the dataset and using a FIRST. and LAST. function helped identify the correct records by looking down datasets. PROC TRANSPOSE can be powerful when used appropriately, but isn't always the right SAS® function to start off with. The FIRST. and LAST. functions were needed to identify those records needing to be transposed.

Here's the code to double check that no patient was lost:

```
proc sql;
create table count as
select unique(visitidentifier), count(unique(resultcatalogname)) as test_count
from  example_lab_prep;
run;/*5404*/
```

```
proc sql;
create table count2 as
select unique(visitidentifier), count(unique(resultcatalogname)) as test_count2
from  lab_fortranspose;
run;/*5404*/
```

90-DAY PRIOR ADMIT EXAMPLE:

As stated before, SAS® only manipulates things within the observation, in other words, the manipulation only occurs on the same horizontal line or row. In this PROC SQL merge/match example, the job is to identify patients having hospital admission for the same diagnosis during the prior 90-days. Patients with prior admissions would have their record flagged so the record can be excluded from the final analysis dataset. For this example, it was necessary to look "upward" through the discharge dates in a normalized hospital administrative dataset to find all the possible prior 90-day discharges for the patient by comparing this date with a admit date that would normally occur much later in the dataset. The comprehensive list of all patient records compiled monthly from the electronic health record database and the hospital accounting database used in this example is called the ADMINISTRATIVE dataset.

PROC TRANSPOSE could be used to align the index admit date with the prior discharge date within the same observation, but it was determined after many fruitless days of coding that it would be more effective to use a PROC SQL procedure:

1) The ADMINISTRATIVE dataset (736,178 records and 372 variables) was used as the primary source of all data (the universal_ID variable is the unique patient identifier in this dataset):

	visitidentifier	patientidentifier	universal_id	los	hospitalcode	site	facility	ADMIT_DATE	DISCHARGE_DATE	ssno	icd9code1	icd9code2	
1	10023331	110301	U-999091	2	19	0	hospital_8	13AUG2008	15AUG2008	XXX-XX-XXXX	V30.00	767.2	V05.
2	10023332	110302	U-999092	31	1	1000	hospital_4	20MAR2011	20APR2011	XXX-XX-XXXX	764.93	769	765.2
3	10023333	110303	U-999093	2	10	7000	hospital_5	24MAR2011	26MAR2011	XXX-XX-XXXX	V30.00	V05.3	
4	10023334	110304	U-999094	1	18	11000	hospital_10	01APR2010	02APR2010	XXX-XX-XXXX	413.9	250.00	790.
5	10023335	110305	U-999095	3	18	11000	hospital_10	27MAR2010	30MAR2010	XXX-XX-XXXX	410.71	272.0	414.
6	10023336	110306	U-999096	6	10	7000	hospital_5	25AUG2009	31AUG2009	XXX-XX-XXXX	296.64	309.81	280.
7	10023337	110307	U-999097	3	1	1000	hospital_4	05JUN2012	08JUN2012	XXX-XX-XXXX	715.35	427.31	285.1
8	10023338	110307	U-999097	10	1	1000	hospital_4	10JUL2012	20JUL2012	XXX-XX-XXXX	707.03	707.24	427.
9	10023339	110307	U-999098	14	8	1000	hospital_11	20JUL2012	03AUG2012	XXX-XX-XXXX	730.08	707.24	707.
10	10023340	110307	U-999097	17	1	1000	hospital_4	03AUG2012	20AUG2012	XXX-XX-XXXX	348.39	707.24	584.
11	10023341	110307	U-999098	32	8	1000	hospital_11	20AUG2012	21SEP2012	XXX-XX-XXXX	730.28	707.24	707.
12	10023342	110308	U-999099	2	3	5000	hospital_1	03JUN2011	05JUN2011	XXX-XX-XXXX	569.1	507.1	751.
13	10023343	110309	U-999100	14	3	5000	hospital_1	06JUL2011	20JUL2011	XXX-XX-XXXX	211.3	560.2	584.
14	10023344	110309	U-999100	13	3	5000	hospital_1	23JUL2011	05AUG2011	XXX-XX-XXXX	038.9	567.21	584.
15	10023345	110309	U-999100	21	3	5000	hospital_1	04JAN2012	25JAN2012	XXX-XX-XXXX	569.69	997.49	584.
16	10023346	110310	U-991001	4	1	1000	hospital_4	07NOV2011	11NOV2011	XXX-XX-XXXX	346.71	300.00	339.
17	10023347	110311	U-991002	5	4	4000	hospital_9	09MAR2011	14MAR2011	XXX-XX-XXXX	212.5	518.0	244.
18	10023348	110312	U-991003	52	1	1000	hospital_4	16APR2012	07JUN2012	XXX-XX-XXXX	805.02	785.52	038.4

2) A master patient dataset, PATIENTLIST (1,027 records with 127 variables), containing only patients meeting predefined exclusion and inclusion criteria, such as the ICD-9 code for disease of interest, the correct date range, the correct age range, the correct facilities, etc. was created from this ADMINISTRATIVE dateset:

	visitidentifier	patientidentifier	universal_id	los	hospitalcode	site	facility	admit_date	discharge_date	ssno	ADMIT_DATETIME	DISCHAR
1	10021317	110301	U-991033	2	1	1000	hospital_4	16OCT2012	18OCT2012	XXX-XX-XXXX	16OCT2012:01:45:00	18C
2	10022318	110343	U-991034	5	1	1000	hospital_4	18MAY2012	23MAY2012	XXX-XX-XXXX	18MAY2012:18:43:00	23M
3	10023319	110344	U-991035	12	1	1000	hospital_4	11MAR2012	23MAR2012	XXX-XX-XXXX	11MAR2012:22:16:00	23M
4	10024320	110333	U-991036	2	3	5000	hospital_1	19MAY2013	21MAY2013	XXX-XX-XXXX	19MAY2013:06:09:00	21M
5	10025321	110354	U-991037	7	3	5000	hospital_1	28JUL2010	04AUG2010	XXX-XX-XXXX	28JUL2010:16:20:00	04A
6	10026322	110367	U-991038	17	5	6000	hospital_3	21MAY2010	07JUN2010	XXX-XX-XXXX	21MAY2010:16:13:00	07J
7	10027323	110365	U-991039	7	5	6000	hospital_3	08NOV2012	15NOV2012	XXX-XX-XXXX	08NOV2012:12:30:00	15N
8	10028324	110567	U-991040	7	1	1000	hospital_4	12DEC2012	19DEC2012	XXX-XX-XXXX	12DEC2012:23:52:00	19D
9	10029325	110319	U-991041	2	3	5000	hospital_1	04APR2010	06APR2010	XXX-XX-XXXX	04APR2010:21:30:00	06A
10	10020326	110328	U-991042	4	3	5000	hospital_1	04DEC2012	08DEC2012	XXX-XX-XXXX	04DEC2012:12:00:00	08D
11	10021327	110337	U-991043	5	15	8000	hospital_2	03FEB2010	08FEB2010	XXX-XX-XXXX	03FEB2010:17:13:00	08F
12	10022328	110346	U-991044	4	15	8000	hospital_2	23JUL2009	27JUL2009	XXX-XX-XXXX	23JUL2009:19:35:00	27J
13	10023329	110355	U-991045	4	3	5000	hospital_1	03MAY2012	07MAY2012	XXX-XX-XXXX	03MAY2012:03:30:00	07M
14	10024330	110302	U-991023	37	3	5000	hospital_1	24JAN2011	02MAR2011	XXX-XX-XXXX	24JAN2011:17:19:00	02M
15	10025380	110364	U-991110	9	2	2000	hospital_6	22DEC2012	31DEC2012	XXX-XX-XXXX	22DEC2012:03:44:00	31D
16	10026381	110373	U-991133	10	1	1000	hospital_4	02MAY2013	12MAY2013	XXX-XX-XXXX	02MAY2013:21:04:00	12M
17	10027382	110382	U-991124	10	2	2000	hospital_6	12OCT2012	22OCT2012	XXX-XX-XXXX	12OCT2012:19:51:00	22C
18	10028383	110391	U-991125	8	5	6000	hospital_3	28NOV2010	06DEC2010	XXX-XX-XXXX	28NOV2010:09:05:00	06D
19	10029384	110319	U-991126	5	17	10000	hospital_7	30AUG2012	04SEP2012	XXX-XX-XXXX	30AUG2012:15:11:00	04S

3) Macro variables (&impdate1 through &impdate7) to set the correct date for each facility (hospital_1 through hospital_7) were defined using the following example code for hospital facility #1:

```
%let impdate1 = "13JUN2009"d;
```

4) Macro variable lists (&ids_hosp1 through &ids_hosp7) to set the patients at each facility (hospital_1 through hospital_7) were defined using master patient dataset, PATIENTLIST, and the following example code for hospital facility #2:

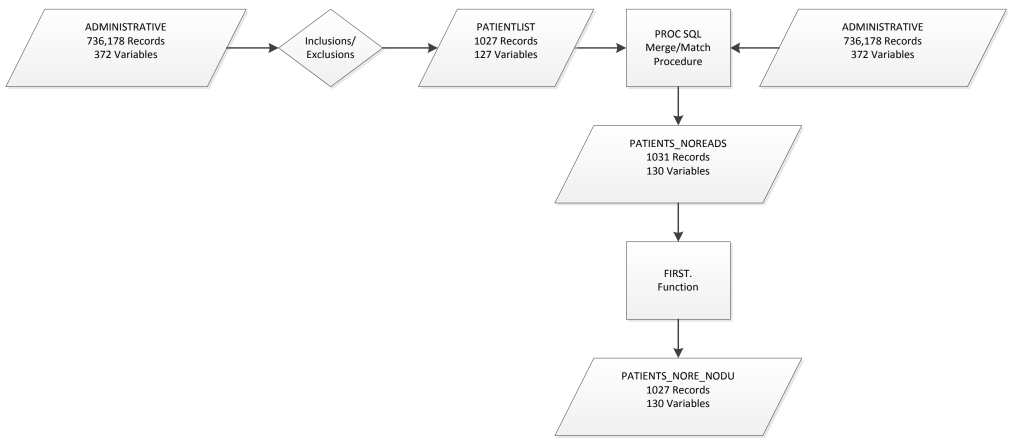
```
proc sql;
 select distinct quote(strip(patientidentifier))
     into :ids_hosp2 separated by ","
 from  outdata.patientlist
 where facility= 'hospital_2';
 quit;
```

5) A PROC SQL procedure was used to LEFT JOIN this master dataset to the 736,178 record ADMINISTRATIVE dataset in order to align the index admit date with the prior discharge date to create the PATIENTS_NOREADS dataset. This resulting dataset had 1031 records and 130 variables. The highlighted area shows those patients that had a prior admission; however, these patients were discharged 837, 604 and 349 prior to the index admit date with none being discharged in the previous 90 days:

	visitidentifier	patientidentifier	universal_id	los	hospitalcode	site	facility	admit_date	discharge_date	ssno	pre90_facility	pre90dishcarge_date	pre90daysbefore
1	10027382	110382	U-991124	10	2	2000	hospital_6	12OCT2012	22OCT2012	XXX-XX-XXXX		.	.
2	10020396	110389	U-991140	2	3	5000	hospital_1	01MAY2011	03MAY2011	XXX-XX-XXXX	hospital_1	14JAN2009	837
3	10020397	110399	U-991152	2	3	5000	hospital_1	28OCT2010	30OCT2010	XXX-XX-XXXX	hospital_1	03MAR2009	604
4	10020398	110490	U-991189	3	15	8000	hospital_2	18MAR2009	21MAR2009	XXX-XX-XXXX	hospital_2	03APR2008	349
5	10024320	110333	U-991036	2	3	5000	hospital_1	19MAY2013	21MAY2013	XXX-XX-XXXX		.	.
6	10024330	110302	U-991023	37	3	5000	hospital_1	24JAN2011	02MAR2011	XXX-XX-XXXX		.	.
7	10021317	110301	U-991033	2	1	1000	hospital_4	16OCT2012	18OCT2012	XXX-XX-XXXX		.	.
8	10029384	110319	U-991126	5	17	10000	hospital_7	30AUG2012	04SEP2012	XXX-XX-XXXX		.	.
9	10027392	110391	U-991136	3	5	6000	hospital_3	23MAY2010	26MAY2010	XXX-XX-XXXX		.	.
10	10022318	110343	U-991034	5	1	1000	hospital_4	18MAY2012	23MAY2012	XXX-XX-XXXX		.	.
11	10020385	110328	U-991127	2	3	5000	hospital_1	10AUG2010	12AUG2010	XXX-XX-XXXX		.	.
12	10029394	110362	U-991138	4	3	5000	hospital_1	01APR2010	05APR2010	XXX-XX-XXXX		.	.
13	10026381	110373	U-991133	10	1	1000	hospital_4	02MAY2013	12MAY2013	XXX-XX-XXXX			

After the PROC SQL merge/match, a simple DATA step using a FIRST. function was used to filter out the all but the most recent duplicated event. Patients with multiple prior admission events represented approximately 0.4% of the analysis population and the final analytical dataset contained 1027 unique patient events.

The process flow to create the desired analytical dataset for this example is:



The PROC SQL code for this project is:

```
proc sql;
 create table work.patients_noreads as
 select  a.*
     ,   b.facility as pre90_facility
     ,   b.discharge_date as pre90discharge_date
     ,   (a.admit_date - b.discharge_date) as pre90daysbefore
 from    outdata.patientlist a
     left join
         (select account_number
             ,   hospitalcode
             ,   site
             ,   ssno
             ,   facility
             ,   patientidentifier
             ,   admit_date
             ,   discharge_date
             ,   icd9code1
             ,   universal_id
          from indata.administrative
          where  (strip(icd9code1)= &icd9ofinterest)
             and ((facility= 'hospital_1' and admit_date < &impdate1 and patientidentifier in (&ids_hosp1))
             or  (facility=  'hospital_2' and admit_date < &impdate2 and patientidentifier in (&ids_hosp2))
             or  (facility=  'hospital_3' and admit_date < &impdate3 and patientidentifier in (&ids_hosp3))
             or  (facility=  'hospital_4' and admit_date < &impdate4 and patientidentifier in (&ids_hosp4))
             or  (facility=  'hospital_5' and admit_date < &impdate5 and patientidentifier in (&ids_hosp5))
             or  (facility=  'hospital_6' and admit_date < &impdate6 and patientidentifier in (&ids_hosp6))
             or  (facility=  'hospital_7' and admit_date < &impdate7 and patientidentifier in (&ids_hosp7)))) b
     on a.univeral_id = b.universal_id and a.patientssno = b.patientssno
 where (a.admit_date - b.discharge_date) < 0 or (a.admit_date - b.discharge_date) > 90;
 quit;
```

LOOKING DOWN LONGITUDINAL DATA:

By nature, EHR data provides a longitudinal look at a patient. For researchers and those interested in looking at how patients are faring over time, this represents a wealth of opportunity. Over a particular time period, patients can be represented a multitude of times in any particular data extract taken from an EHR system. This potentially means a list of patients with multiple unique rows in the dataset.

One of the most challenging things in our histories as SAS® programmers has been looking down a dataset (in this case a series of patient records) and grabbing important information (in this case the next patient encounter date and information from that encounter) and then reporting it on that information on the first record.

Here is an example of what a series of records in our EHR might look like after putting the pieces of the relational database together (the universal_ID variable is the unique patient identifier in this dataset):

	UNIVERSAL_ID	facility	LOS	AGE	SEX	RACE	ETHNICITY	PAYER	ADMIT_DATE	DISCHARGE_DATE	dx1	dx2	dx3
37	U-999991	hospital_7	2	41	M	W	N	Other	19AUG2011	21AUG2011	435.9	403.91	423.9
38	U-999991	hospital_7	10	41	M	W	N	Other	04SEP2011	14SEP2011	588.81	403.91	585.6
39	U-999991	hospital_7	4	42	M	W	N	Medicare	27FEB2013	03MAR2013	996.81	403.91	275.41
40	U-999991	hospital_7	7	42	M	W	N	Medicare	29MAR2013	05APR2013	996.81	285.21	457.8
41	[illegible]	hospital_7	1	[illegible]	M	W	N	Other	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
42	U-999976	hospital_7	2	64	M	W	N	Other	08NOV2008	10NOV2008	414.01	153.8	496
43	U-999976	hospital_7	32	65	M	W	N	Other	08MAR2010	09APR2010	491.21	038.12	415.19
44	U-999976	hospital_7	3	67	M	W	N	Other	17SEP2011	20SEP2011	569.62	348.39	518.83
45	U-999976	hospital_7	7	68	M	W	N	Other	29OCT2012	05NOV2012	569.69	348.39	518.84
46	U-999976	hospital_7	11	68	M	W	N	Other	08NOV2012	19NOV2012	998.59	518.84	348.30
47	U-999976	hospital_7	4	68	M	W	N	Other	10DEC2012	14DEC2012	491.21	518.84	V46.2
48	U-999976	hospital_7	11	68	M	W	N	Other	23DEC2012	03JAN2013	428.23	486	518.84
49	U-999972	hospital_7	7	46	M	B	N	Other	19NOV2008	26NOV2008	403.91	585.6	789.59
50	U-999972	hospital_4	1	48	M	B	N	Other	16MAR2011	17MAR2011	276.7	585.6	403.01
51	U-999972	hospital_4	9	48	M	B	N	Other	20JUN2011	29JUN2011	682.6	585.6	008.45
52	U-999972	hospital_4	9	49	M	B	N	Medicare	23SEP2012	02OCT2012	996.62	038.11	348.31
53	U-999972	hospital_4	5	50	M	B	N	Medicare	14NOV2012	19NOV2012	482.9	585.6	403.91
54	U-999972	hospital_4	7	50	M	B	N	Medicare	14MAY2013	21MAY2013	824.6	585.6	588.81
55	U-999972	hospital_4	1	50	M	B	N	Medicare	02JUN2013	03JUN2013	276.69	585.6	403.01
56	U-999972	hospital_4	1	50	M	B	N	Medicare	22JUL2013	23JUL2013	996.1	585.6	403.91
57	U-999972	hospital_4	13	50	M	B	N	Medicare	03AUG2013	16AUG2013	824.4	585.6	996.49
58	U-999972	hospital_8	1	46	M	B	N	Other	05APR2009	06APR2009	514	996.73	403.91
59	U-999972	hospital_8	3	48	M	B	N	Other	07NOV2010	10NOV2010	403.01	585.6	287.5
60	U-999972	hospital_8	2	48	M	B	N	Other	15NOV2010	17NOV2010	276.7	403.01	585.6
61	U-999972	hospital_8	2	48	M	B	N	Medicare	21AUG2011	23AUG2011	276.7	585.6	403.91
62	U-99997	hospital_8	4	42	F	B	N	Other	18APR2008	22APR2008	626.2	218.1	218.2
63	U-999968	hospital_7	3	66	M	W	N	Medicare	01DEC2011	04DEC2011	223.0	571.8	412
64	U-999949	hospital_7	1	47	M	B	N	Self_Pay	09FEB2009	10FEB2009	727.00	305.1	
65	U-999943	hospital_9	2	40	F	W	N	Other	11APR2013	13APR2013	574.00	V85.42	493.90
66	U-999941	hospital_8	1	69	F	W	N	Medicare	28DEC2009	29DEC2009	250.02	599.0	041.3
67	U-999926	hospital_4	1	41	F	W	N	Other	01MAY2009	02MAY2009	427.31	276.8	427.89
68	U-999925	hospital_7	4	41	F	B	N	Other	03JUL2012	07JUL2012	428.41	042	284.19
69	U-999913	hospital_7	7	30	F	B	N	Other	05SEP2008	12SEP2008	654.21	289.81	648.91
70	U-999913	hospital_7	3	31	F	O	N	Other	29JAN2010	01FEB2010	654.21	289.81	648.91
71	U-999913	hospital_7	3	34	F	O	N	Other	20JUN2012	23JUN2012	654.21	658.01	289.81
72	U-999903	hospital_7	4	62	F	W	N	Other	05NOV2012	09NOV2012	346.71	579.3	723.0
73	U-99989	hospital_8	5	47	F	W	H	Other	15MAY2013	20MAY2013	415.19	453.42	300.00

Let's say, as a researcher it is important to identify any patient with a subsequent encounter at a healthcare system **AND** then make a comparisons between the first visit and their follow-up visits. For example, what if you were looking to see if they had the same principle diagnosis, DRG code, and see the changes in their BMI on their on their next visit. In this instance, you would need to look down the dataset and report back values related to diagnosis code, DRG code and BMI on the subsequent visit. You can see from the example data above that there are many individuals with multiple repeat visits.

For the purposes of this example let’s focus on Universal_ID number “U-99991”:

	UNIVERSAL_ID	facility	LOS	AGE	SEX	RACE	ETHNICITY	DRG	PAYER	ADMIT_DATE	DISCHARGE_DATE	dx1	BMI
1	U-999991	hospital_7	3	37	M	W	N	700	Other	22APR2008	25APR2008	996.81	41
2	U-999991	hospital_7	15	38	M	W	N	659	Other	26MAR2009	10APR2009	996.81	39.6
3	U-999991	hospital_7	2	41	M	W	N	069	Other	19AUG2011	21AUG2011	435.9	38
4	U-999991	hospital_7	10	41	M	W	N	674	Other	04SEP2011	14SEP2011	588.81	37.3
5	U-999991	hospital_7	4	42	M	W	N	652	Medicare	27FEB2013	03MAR2013	996.81	34
6	U-999991	hospital_7	7	42	M	W	N	661	Medicare	29MAR2013	05APR2013	996.81	34.3

This patient has 6 visits it contributes to this dataset since 2008, all with varying data points. As a programmer with minimal exposure to PROC SQL and extensive knowledge of the DATA step and PROC TRANSPOSE, the natural inclination maybe to transpose each variable individually and merge those transposed datasets back together. In the real world, this could account for a lot of transpositions and the creation of numerous datasets. To bypass this mess, here is an array that can be adapted to do several things, including transposing all variables in one step; however, for the purposes of this example, it will provide only the following information for the following inpatient visit: BMI, DRG codes, and principal diagnosis code:

```
proc sql;
    select put(max(counter),1.),
           put(max(counter)-1,1.)
        into :last,
            :dim
        from (select count(universal_ID) as counter
                from work.for_paper_ex1
                    group by universal_ID);
quit;
```

```
data for_paper_ex2;
length facility $40.;
     do _n_=1 by 1 until (last.universal_ID);
          set for_paper_ex1;
              by universal_ID;

              array fix             (*) date_1 - date_&last;
              array fix2            (*) date1_1 - date1_&last;
              array stay            (*) los_1-los_&last;
              array exit            (*) discharge_1-discharge_&last;
              array aged            (*) ages_1-ages_&last;
              array pay             (*) $ payment_1-payment_&last;
              array dis_disp        (*) $ disdisp_1-disdisp_&last;
              array facilitycode    (*) $40.     facils_1-facils_&last;
              array drgcode         (*) $codes_1-codes_&last;
              array drgcode2        (*) $codes1_1-codes1_&last;
              array diag            (*) $40. dxcodes_1-dxcodes_&last;
              array diag2           (*) $40. dxcodes2_1-dxcodes2_&last;
              array bmis            (*) bmis_1-bmis_&last;
              array bmis2           (*) bmis2_1-bmis2_&last;
              array fac             (*) fac_1-fac_&last;
                   fix              (_n_)=admit_date;
                   stay             (_n_)=los;
                   exit             (_n_)=discharge_date;
                   aged             (_n_)=age;
                   pay              (_n_)=payer;
                   dis_disp         (_n_)=disposition_code;
                   facilitycode     (_n_)=facility;
                   drgcode          (_n_)=drg;
                   diag             (_n_)=dx1;
                   bmis             (_n_)=bmi;

                do i=1 to &dim.;
                     fix2          (i)=fix(i+1);
                     drgcode2      (i)=drgcode(i+1);
                     bmis2         (i)=bmis (i+1);
                     diag2         (i)=diag (i+1);

       end;
                end;
```

```
            do j=1 to dim(fix);
                admit_date_clean              = fix(j);
                los_array                     = stay(j);
                next_inpatient_start_date     = fix2(j);
                discharge_date_clean          = exit (j);
                age_clean                     = aged(j);
                payment_clean                 = pay (j);
                disposition_clean             = dis_disp (j);
                facility_code_clean           = facilitycode (j);
                drg_code_clean                = drgcode (j);
                Next_drg_code                 = drgcode2 (j);
                primdx_clean                  = diag (j);
                next_prim_dx_clean            = diag2 (j);
                bmi_clean                     = bmis   (j);
                next_bmi_clean                = bmis2 (j);

if fix(j)=. then leave;
        output;
    end;
format admit_date_clean next_inpatient_start_date discharge_date_clean date9. facility $40.;

drop    age facility los sex race ethnicity payer bmi drg dx1 payer
        admit_date discharge_date disposition_code
        date_1 - date_&last
        los_1-los_&last
        discharge_1-discharge_&last
        ages_1-ages_&last
        payment_1-payment_&last
        disdisp_1-disdisp_&last
        facils_1-facils_&last
        codes_1-codes_&last
        dxcodes_1-dxcodes_&last
        dxcodes2_1-dxcodes2_&last
        bmis_1-bmis_&last
        date1_1 - date1_&last
        codes1_1-codes1_&last
        bmis2_1-bmis2_&last
        fac_1-fac_&last
        i
        j;
run;
```

	UNIVERSAL_ID	admit_date_clean	next_inpatient_start_date	drg_code_clean	Next_drg_code	primdx_clean	next_prim_dx_clean	bmi_clean	next_bmi_clean
1	U-999991	22APR2008	26MAR2009	700	659	996.81	996.81	41	39.6
2	U-999991	26MAR2009	19AUG2011	659	069	996.81	435.9	39.6	38
3	U-999991	19AUG2011	04SEP2011	069	674	435.9	588.81	38	37.3
4	U-999991	04SEP2011	27FEB2013	674	652	588.81	996.81	37.3	34
5	U-999991	27FEB2013	29MAR2013	652	661	996.81	996.81	34	34.3
6	U-999991	29MAR2013	.	661		996.81		34.3	.

The key to this array is highlighted in red; it determines which variables are brought to the previous visit's record. As you can see the resultant dataset contains the output from the array and the data from the next visit labeled as such. The above array, although effective, really isn't efficient. It produces a lot of unnecessary bi-product, most of which isn't seen because of the drop statement at the tail end of the code. Additionally, there are a lot of key strokes involved, which in the programming world can translate to misspellings, missed ';' and other opportunities for your code to break.

Programmers familiar with PROC SQL and those "thinking outside the box" can simplify the code to look down a dataset. The use of a left join to join the original dataset to itself, but looking for the next patient encounter, identified by the calculation of the time to readmission, can simplify the code and produce the same results without renamed variables (aside for the ones you want).

```
proc sql;
    create table for_paper_ex5 as select
        a.*
        ,b.admit_date as next_inpatient_start_date
        ,b.drg as Next_drg_code
        ,b.bmi as next_bmi_clean
        ,b.dx1 as primdx_clean

        from    for_paper_ex1 a
                    left  join
                for_paper_ex1 b
                    on
            a.universal_ID = b.universal_ID  and (b.admit_date - a.discharge_date >0)
        group by a.admit_date
    having (b.admit_date - a.discharge_date)= min(b.admit_date - a.discharge_date)
order by a.universal_id, a.admit_date ;
quit;
```

	UNIVERSAL_ID	ADMIT_DATE	DISCHARGE_DATE	next_inpatient_start_date	DRG	Next_drg_code	dx1	primdx_clean	BMI	next_bmi_clean
1	U-999991	22APR2008	25APR2008	26MAR2009	700	659	996.81	996.81	41	39.6
2	U-999991	26MAR2009	10APR2009	19AUG2011	659	069	996.81	435.9	39.6	38
3	U-999991	19AUG2011	21AUG2011	04SEP2011	069	674	435.9	588.81	38	37.3
4	U-999991	04SEP2011	14SEP2011	27FEB2013	674	652	588.81	996.81	37.3	34
5	U-999991	27FEB2013	03MAR2013	29MAR2013	652	661	996.81	996.81	34	34.3
6	U-999991	29MAR2013	05APR2013	.	661		996.81		34.3	.

CONCLUSION

If you are a researcher who uses EHR and secondary data, it's sometimes necessary to stray from your comfort zone and get creative in how you handle traditional data restructuring procedures. PROC SQL, as well as inventive uses of already familiar SAS® procedures/functions, proves to be good tools that can provide the necessary programming flexibility. PROC SQL can reduce the number of variable re-names and lessen the need for creating in-between variables while achieving the same end as a DATA step as well as providing a means to look up and down datasets. Meticulous pre-planning can also aid in a data manager's success of restructuring longitudinal data. Here are some helpful tips:

1) Map out the analysis dataset structure.
   In this step, BE BOLD and break the datasets into manageable sizes, if needed. This could save in trying to account for multiple conditions in IF/THEN statements.
2) Outline the variables that will be needed to achieve the desired results.
   In this step, again BE BOLD and eliminate superfluous variables, this will help you visualize where your data needs to go.
3) Determine and/or create unique variable key(s) that will aid in joining the datasets.
   In this step, BE CREATIVE and put in the most thought into this part.

To be successful, data managers must be efficient in identifying methods to organize and restructure data to assure its usefulness in analysis and a little dose of the SAS® can brighten any data manager's healthcare blues.

REFERENCES

[1] GE Healthcare. 2005. Data Dictionary for Centricity® Physician Office – EMR 2005. Milwaukee. WI.