Paper 064-31

Using SAS® Enterprise Guide® to Code When You're Not a Programmer

Britta K. August, CareSource Management Group, Dayton, Ohio

ABSTRACT

If you are new to SAS® and have never been a programmer, SAS® Enterprise Guide® is an excellent tool to get you started. With a few tips to help out, you can tackle some seemingly daunting tasks right away. Examples to be covered include the difference between the inner and outer SQL joins; two methods to approach analysis; and ways to organize tasks, queries and code efficiently by taking advantage of icon names in the process flow view. Once you begin to create your own data sets using EG queries, you will quickly encounter the need to store either a reference to the data created, or the data itself. An approach to saving data will also be presented. All examples presented were created in SAS® Enterprise Guide® 3.0 and are intended for a beginning level.

INTRODUCTION

SAS® Enterprise Guide® allows you to access data with an intuitive interface. If you have never been a programmer or have had little experience in retrieving data, a few tips here will help you be more effective faster. The knowledge of how to use queries with inner and outer joins will launch you into using the tool right away. A couple methods to organize your work in a SAS® Enterprise Guide® project can help keep you moving in the right direction, and a method to save your results will definitely help keep your sanity because lost data can send the most patient person into a panic.

WHAT IS AN INNER JOIN AND OUTER JOIN, AND WHY DO I CARE ANYWAY?

If you have multiple data sets that both contain pieces of the answer to your question, the difference between an inner and outer join will be critical to achieving the correct results. We will use the following data sets in examples of both types of joins:

Data set Asthma_Claims

Physician_or_Facility	Patient_Nbr	Service_Date
Dr. Williams	31-1234	01/23/2005
Urgent Care of Miami Valley	31-2345	02/15/2005
Dr. Barde	31-3456	03/20/2995
Dr. Hartnell	31-9876	06/30/2005
Miami Valley Hospital Emergency Dept.	31-1234	06/30/2005
Dr. Hunter	31-5555	05/01/2005
Dr. Feng	31-8787	06/19/2005
Mercy Hospital	31-9876	07/05/2005
Dr. Feng	31-2255	09/15/2005
Dr. Bullington	31-8989	07/31/2005
Miami Valley Hospital Emergency Dept.	31-5555	07/04/2005
Dr. Sampers	31-1234	04/15/2005
Dr. Kardan	31-8888	02/27/2005
Mercy Hospital	31-9876	06/08/2005
Miami Valley Hospital Emergency Dept.	31-4567	07/04/2005

Data set Patient_Demographics

Patient_Nbr	Patient_Name	State	Age
31-1234	Mary Jones	OH	19
31-3333	John Smith	MN	5
31-5655	Chase Colton	LA	9
31-2255	Cameron Suarez	NY	50
31-2345	Raj Mehta	OH	43
31-3456	Carol Van Dyke	MI	2
31-4567	Jessica Miller	OH	5
31-5555	Lisa Helm	MI	16

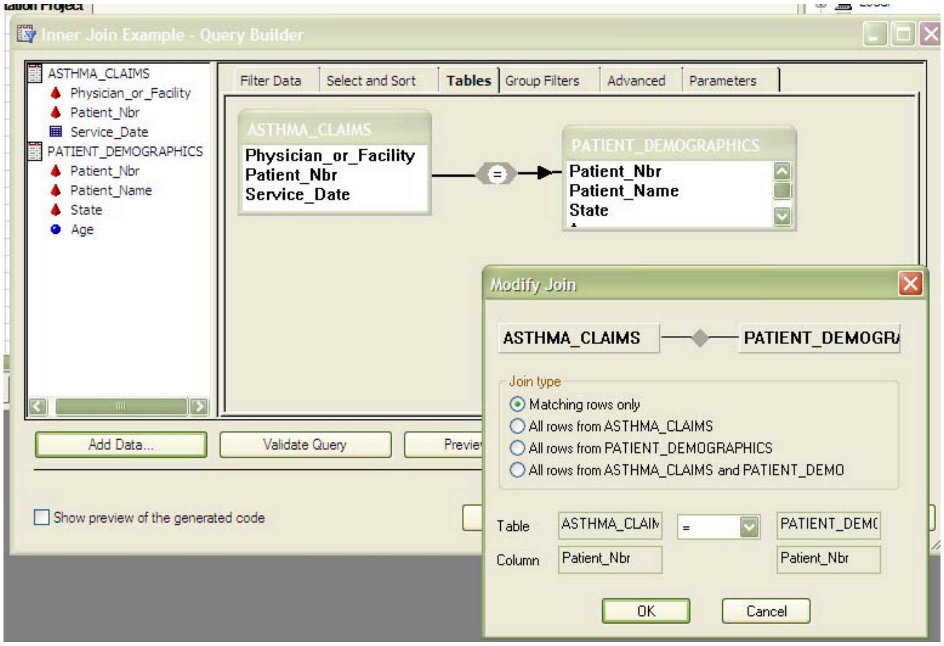
31-6599	Brian McKenney	IN	14
31-8787	Sneeha Mahambrey	LA	65
31-8888	Jose Suarez	MI	22
31-8989	Theresa Simpson	FL	11
31-9876	Chris O'Campo	NC	12
31-4141	Janet Robinson	OH	8
31-2252	Morgan Watterson	SC	2

INNER JOIN
An inner join will return only the data in each table that matches based on criteria you set. To create an inner join in a SAS® Enterprise Guide® query, start a query on one data set, then click the add data button in your query builder to join to a second data set. If your two sets each have columns with matching names, the program will assume the first match found should comprise the defaulting inner join. (and by the way – you can change this setting in *tools, options, Query by unchecking "automatically attempt to join tables in query"*) If the tables do not contain any columns with identical names, simply drag a column from one table on top of the related column in the second table.

If we asked the question "how old are the patients with asthma claims?" we would use an inner join and get the following results…

Patient_Nbr	Patient_Name	Age
31-1234	Mary Jones	19
31-2255	Cameron Suarez	50
31-2345	Raj Mehta	43
31-3456	Carol Van Dyke	2
31-4567	Jessica Miller	5
31-5555	Lisa Helm	16
31-8787	Sneeha Mahambrey	65
31-8888	Jose Suarez	22
31-8989	Theresa Simpson	11
31-9876	Chris O'Campo	12

The tables tab in our SAS® ENTERPRISE GUIDE® query builder looks like this…

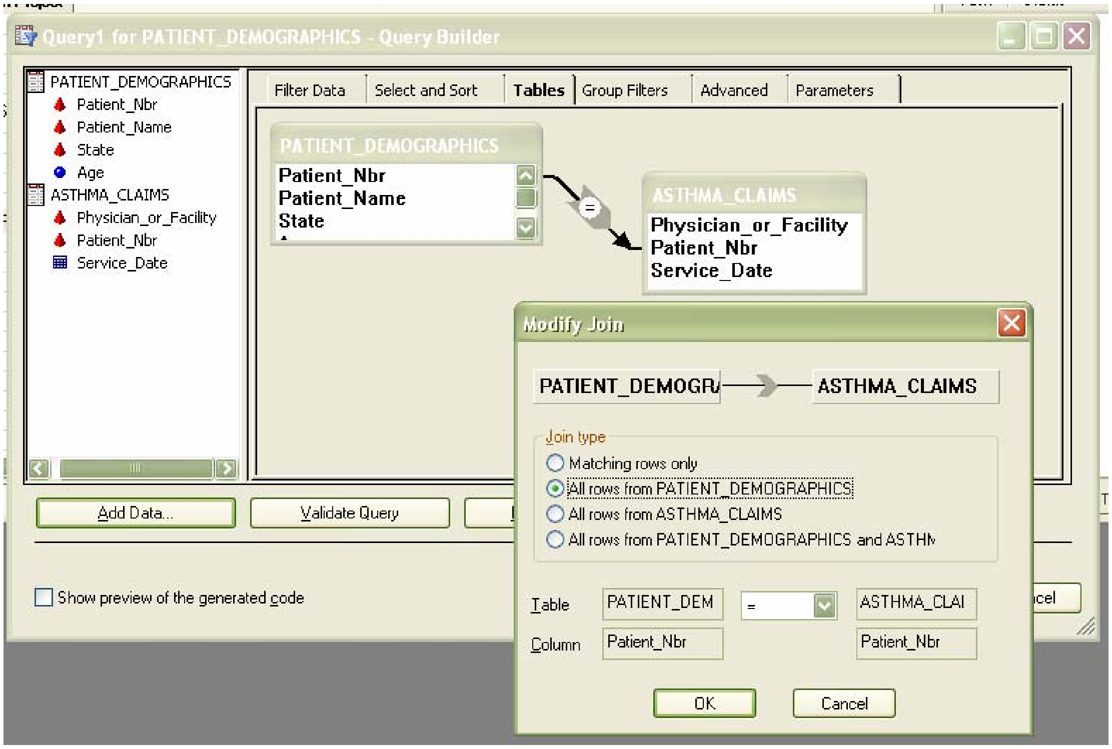


Join type
● Matching rows only
○ All rows from ASTHMA_CLAIMS
○ All rows from PATIENT_DEMOGRAPHICS
○ All rows from ASTHMA_CLAIMS and PATIENT_DEMO

Table	ASTHMA_CLAIM	=	PATIENT_DEM(
Column	Patient_Nbr		Patient_Nbr

OUTER JOIN
An outer join will return all the data from one table, and only the data from the other table that matches. For rows in the first table that have no match in the second, any columns selected from the second table will contain missing values. To create an outer join in a query, you will modify the join by right clicking on the arrow between the tables.

Again using our Asthma Claims and Patient Demographics data sets, let's ask the question, "which patients did not have asthma claims"? After we add both tables, we will change our join to "All rows from PATIENT_DEMOGRAPHICS".



Modify Join

PATIENT_DEMOGR → ASTHMA_CLAIMS

Join type

○ Matching rows only
● All rows from PATIENT_DEMOGRAPHICS
○ All rows from ASTHMA_CLAIMS
○ All rows from PATIENT_DEMOGRAPHICS and ASTHM

Table: PATIENT_DEM = ASTHMA_CLAI

Column: Patient_Nbr Patient_Nbr

Next we will remove all columns except both Patient_Nbr columns and the Patient_Name column and uncheck the box for "Allow Duplicate Rows" on the Select and Sort tab. This will yield the following results…

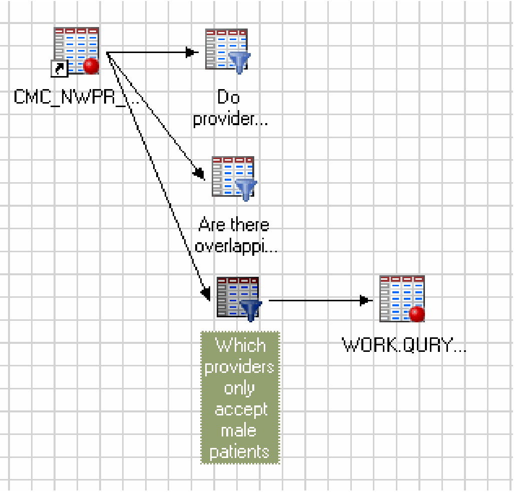
Patient_Nbr	Patient Name	Patient Nbr (from Asthma Claims)
31-1234	Mary Jones	31-1234
31-2252	Morgan Watterson	
31-2255	Cameron Suarez	31-2255
31-2345	Raj Mehta	31-2345
31-3333	John Smith	
31-3456	Carol Van Dyke	31-3456
31-4141	Janet Robinson	
31-4567	Jessica Miller	31-4567
31-5555	Lisa Helm	31-5555
31-5655	Chase Colton	
31-6599	Brian McKenney	
31-8787	Sneeha Mahambrey	31-8787
31-8888	Jose Suarez	31-8888
31-8989	Theresa Simpson	31-8989
31-9876	Chris O'Campo	31-9876

PARTS IS PARTS: METHODS TO FOLLOW FOR THOSE WHO HAVE NEVER HAD TO GET THEIR OWN DATA

Now that you have a general idea of how to use queries to get at your data, here are a couple suggested methods to follow to keep your analysis intact.

METHOD 1: TALKING TO THE DATA

Sometimes we may need to ask a lot of questions that will help us make decisions about the next steps to take with our data. In these cases, I find it extremely helpful to rename my queries with the question I am asking. When I have exhausted my inquiries, I can delete the resulting data sets from the project so that only the questions remain. This allows me to recreate the answer to support my findings if it is ever needed.



METHOD 2: CREATING MORE IN-DEPTH ANALYSIS BY BUILDING PROCESS FLOWS

Along with renaming my queries to reflect the question asked, I find breaking up a large question into small, logical pieces to be easy to both follow and debug.

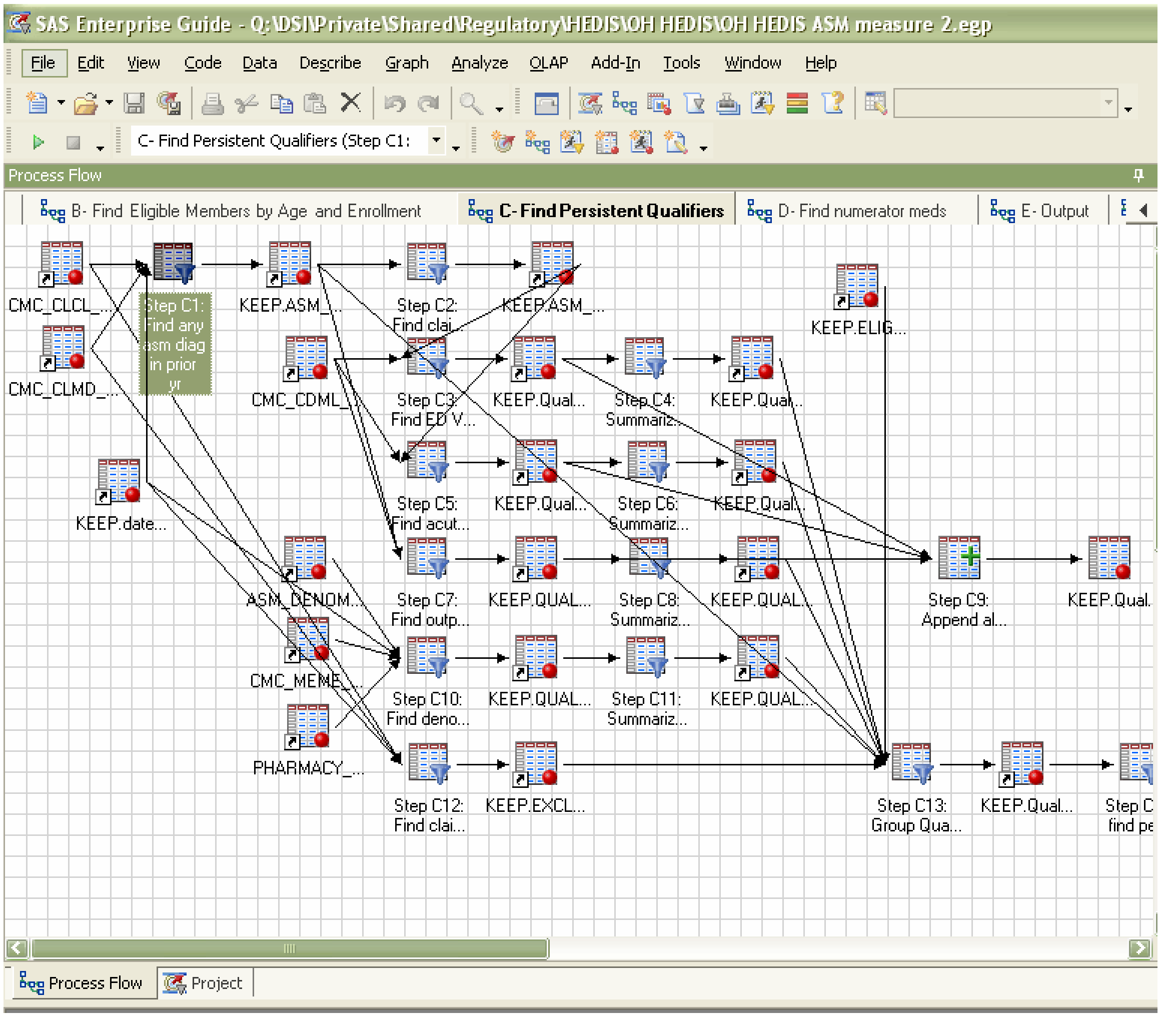
For example, I was asked to replicate the HEDIS asthma measure for my company. In short, this will measure members of our insurance plan with persistent asthma who have received medications to control their illness. The denominator consists in part of claims received for asthma that indicate that a person is a persistent asthmatic. The claims could be

- One or more emergency department visits with a primary diagnosis of asthma
- One or more inpatient stays with a primary diagnosis (Dx) of asthma
- Four or more outpatient visits with any Dx of asthma.

I will not cover every step shown in the screen shot below, but a shortened version has been bulleted here.

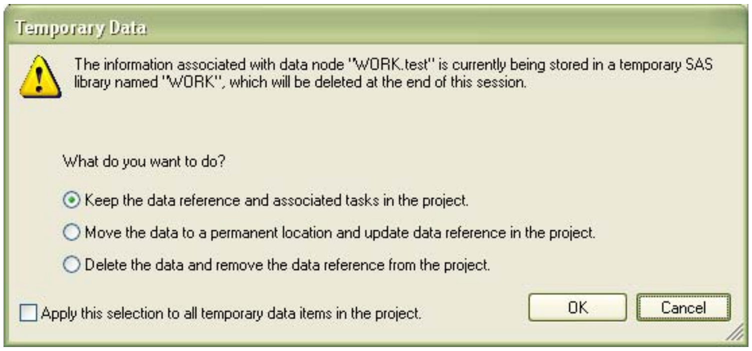
- I began first by finding all claims with any asthma Dx in step C1.
- I then found claims with a primary Dx of asthma by building from my results from C1.
- Next I looked to see if the claims with a primary Dx of asthma met the criteria for an emergency department visit.
- Fourth I compared the same list of primary asthma Dx claims to the inpatient criteria.
- Then I compared the list of all asthma claims to the outpatient criteria.
- Now that I had all the claims for my persistent criteria, I put them in one table and summarized them.

Each successive step built a piece needed for the outcome, and often these steps built upon the previous step. The screen shot below shows each of these steps (and more). It is most important to note that each step has been named to reflect the task at hand and output named appropriately. I also took care to organize the icons in the process flow window so that the order of steps was easily recognizable in pattern similar to the way one would read a book. I.E. Steps are followed first left to right, then top to bottom.



KEEPING YOUR SANITY

In the setup of your preferred working mode, incorporating a plan to save queries upon queries is essential. If you have created any queries with results in the WORK library, the SAS® Enterprise Guide® default settings prompt you with the following message box upon saving your project…



Temporary Data

The information associated with data node "WORK.test" is currently being stored in a temporary SAS library named "WORK", which will be deleted at the end of this session.

What do you want to do?

● Keep the data reference and associated tasks in the project.

○ Move the data to a permanent location and update data reference in the project.

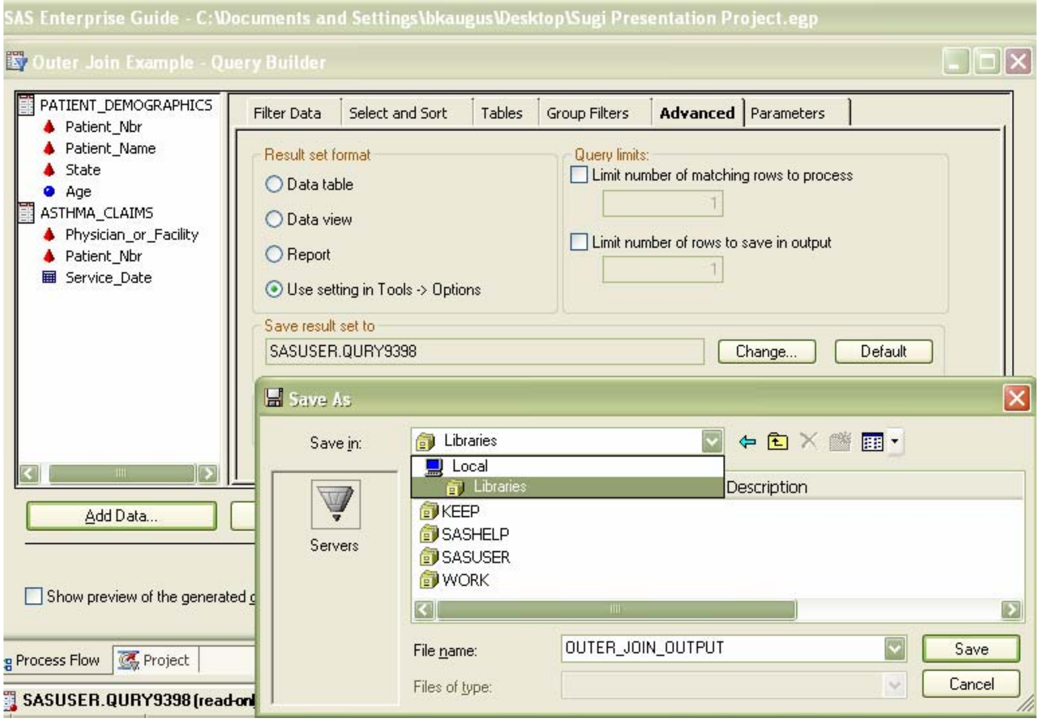
○ Delete the data and remove the data reference from the project.

□ Apply this selection to all temporary data items in the project.

OK Cancel

- If you select the first radio button, your WORK data set will remain as icons in your process flow window.
- The second choice will prompt you to save your WORK data set to a permanent library.
- Lastly, the the third will remove the data set entirely. **Be very careful with that third option!** The deletion is permanent and very painful if made mistakenly.

There is, however, another option. The tools, options window contains a checkbox captioned "automatically save references to temporary data" which, when checked, forces you to save all data to a permanent location just as if you always selected the second radio button. In order to utilize this feature effectively, I have created a SAS® program that resides in the project that contains only a LIBNAME statement that I call my KEEP library. If I have any data sets that I do not want to remove, or if they have queries built upon them, I send them to the KEEP library, by changing the output location on the advanced tab to my KEEP library, then calling the data set by a meaningful name.


SAS Enterprise Guide - C:\Documents and Settings\bkaugus\Desktop\Sugi Presentation Project.egp

Outer Join Example - Query Builder

PATIENT_DEMOGRAPHICS
- Patient_Nbr
- Patient_Name
- State
- Age

ASTHMA_CLAIMS
- Physician_or_Facility
- Patient_Nbr
- Service_Date

Result set format:
- ○ Data table
- ○ Data view
- ○ Report
- ● Use setting in Tools -> Options

Query limits:
- □ Limit number of matching rows to process: 1
- □ Limit number of rows to save in output: 1

Save result set to: SASUSER.QURY9398

□ Show preview of the generated q

SASUSER.QURY9398 (read-onl

Save As

Save in: Libraries

- Local
- Libraries
- KEEP
- SASHELP
- SASUSER
- WORK

File name: OUTER_JOIN_OUTPUT

Files of type:

CONCLUSION

The understanding of inner and outer joins can get you started in SAS® Enterprise Guide® immediately and creates more avenues to ask increasingly complex questions. Hopefully the common sense approaches to the organization of your process flow window will help you to avoid some common frustrations. Once a process flow is created, saving data in a "KEEP" library can *keep* you sane.

ACKNOWLEDGEMENTS

I would like to thank the entire Decision Support & Informatics team at CareSource Management Group for their kind words of support and wisdom.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please feel free to contact the author at:

Britta K. August
CareSource Management Group
One Dayton Centre
1 South Main Street
Dayton, Ohio 45402
(937) 531-2922
britta.august@csmg-online.com