Paper 1281-2014

Empowering Clinical Research Staff with SAS® Enterprise Guide®

Chris Schacherer, Clinical Data Management Systems, LLC

ABSTRACT

The availability of specialized programming and analysis resources in academic medical centers is often limited, creating a significant challenge for clinical research. The current work describes how Base SAS® and SAS® Enterprise Guide® are being used to empower research staff so that they are less reliant on these scarce resources.

INTRODUCTION

The role SAS software plays in the world of private-sector pharmaceutical and biotechnology research is well-documented—from managing, cleaning, and analyzing data to reporting and submission. In these environments, SAS often plays a highly prescribed role in very specific work performed by a relatively small number of highly-trained individuals. As a function, the usage of "statistical" and "programming" resources is carefully allocated to projects based on organizational strategy and tactics to bring products to market. Outside of those roles assigned to use SAS, access to the software is limited. Conversely, many academic medical centers are affiliated with universities which hold Training and Research licenses for SAS that make access to the software relatively inexpensive. Of course, in these institutions, it is also often the case that access to seasoned SAS programmers and statisticians is somewhat limited—creating tremendous opportunities for less experienced programmers and analysts to learn SAS and apply it to a number of interesting challenges that arise in the clinical research enterprise.

The current work describes how BASE SAS and SAS Enterprise Guide were used to improve the efficiency of a translational research program in an academic medical center. The research program described involves physicians and scientists from several disciplines who are involved in pursuing cures for disease through both basic-science (bench) research and clinical trials of potential treatments. As these organizations often have three equally important goals of treating patients, training the next generation of clinicians and scientists, and conducting research that stretches the boundaries of human knowledge, the people conducting clinical research in these organizations are often pulled in many directions on any given day. In the remainder of this paper we will explore how SAS was applied to help support their work.

THE ENVIRONMENT & WORKFLOWS

One of the cornerstones of this research program was a unique research database, "ClinicalCoreLab" (CCL) that allowed clinical data coordinators to track characteristics related to the natural history of research participants' solid-tumor cancers. As patients enrolled in research studies, their demographics, disease characteristics, treatments, treatment outcomes, and vital status follow-up information were all tracked over time in significant detail. In addition, the database supported a comprehensive tissue banking system capable of tracking research tissue samples following clinic-based biopsies or surgical excisions. Every movement, processing, and authorized use of these tissues for research was carefully tracked in the database. In addition, patients consenting to the use of their tumors for research purposes were also asked to allow the research program to request their tumor tissue from other hospitals and pathologists for use in these studies. These requests and the tracking of the shipment of those specimens to the research facility (and back to their source institution) were also carefully tracked over time. Because the research program was very data rich, it seemed natural that SAS would be a significant component of the research infrastructure.

BASE SAS OPERATIONAL SUPPORT.

One of the job functions that needed support was that of the Study Coordinator. These individuals manage research protocols—ensuring coordination of a number of activities from consenting patients in the clinic, monitoring adherence to study treatment protocols, drawing blood, maintaining communication with enrolled patients, and coordinating study visits with the clinical care team. One daily task that was ripe with opportunity for improvement was the integration of data from multiple sources that is necessary in order for study coordinators to plan their day. A form of the "two-monitor problem", study coordinators often need to access multiple systems and manually cross-check data from those systems in order to put together a daily work plan. For example, it is often necessary for study coordinators and research nurses to cross-reference the day's clinic schedule with the enrollment information on their studies to know which patients on their studies will be in a given clinic on a given day. If a study-participant coming to the clinic on a given day is due to be re-consented on a revised version of the protocol or will be available for a study-related blood-draw, the study coordinator needs to know this information so they can coordinate study activities with normal clinic operations and minimize the burden on the study participant. Traditionally, this entails pulling up the clinic schedule on one monitor, opening the protocol management system on another monitor, and manually checking each patient on the clinic schedule to see if (a) they are participating in any of the protocols the study coordinator is managing and (b) they are due for any study-related activities (blood/tissue sample collection, radiologic assessment, reconsenting on a new version of the protocol's consent form, etc.). This can easily require an hour of work every morning before the start of the clinic day.

AUTOMATED SCHEDULE INTEGRATION

Using BASE SAS, a solution was created to perform this work for the study coordinator—saving them roughly five hours every week. In the following example, a study-specific work-list is created that will allow the study coordinator or research nurse to know which patients will be available for study-related blood draws today, what time their appointments are, and in which clinic. This information will be delivered to study coordinators automatically by e-mail every morning at 6:00 a.m.

The first step is to create connections to the Oracle-based "ClinicalCoreLab" and to the hospital scheduling system. In each case, a SAS library is defined that forms a symbolic link to the data tables in the relevant Oracle and Microsoft database schemas. In the first case, the library "ccl" is created to connect SAS to the Oracle database schema "clincore" on the "research" database server. The library has all of the access privileges to read and write data to/from tables in the "clincore" schema that the Oracle user identified by the "&userid" macro variable has been assigned[1]. Similarly, the "sched" library is defined to give user "cschacherer" access to the "schedods" schema—which is an operational data store that shadows the production scheduling system. This catalog exists in the DBO schema of the SQL Server Database running on the server "HOSPXYZ1".

```
LIBNAME ccl ORACLE
   USER = &userid
   PATH = "research"
 SCHEMA = clincore
     PW = &password;

LIBNAME sched OLEDB PROVIDER=SQLOLEDB.1 REQUIRED=Yes
         USER = cschacherer
   DATASOURCE = "HOSPXYZ1"
   PROPERTIES = ('initial catalog'=schedods 'Persist Security Info'=True)
       SCHEMA = 'DBO';
```

In the preceding code, notice the reference to the macro variables "userid" and "password". SAS (1990) defines macro variables in the following way: "Whereas the value of a DATA step variable depends on the observation being processed, a macro variable has a single value that remains constant until explicitly changed" (p. 25). Likewise, Carpenter (1997) describes macro variables as "…variables [that are] stored in memory and are not associated with a data set…" (p.1). The reason a macro variable is used to store the password is so that you do not need to store your Oracle password in the SAS code. This is particularly important if the program is to be shared by multiple people on the research team. Most organizations would frown on storing a clear-text file on the network that contained something like the following (where 'mysecretpassword123' is the actual Oracle password for userid 'chris':

```
LIBNAME ccl ORACLE
   USER = chris
   PATH = "research"
 SCHEMA = clincore
     PW = mysecretpassword123;
```

For this reason, each member of the research team that will be using a SAS program to connect to the Oracle database first creates a SAS dataset (containing their connection information) in their personal network directory. In the following example, the SAS dataset "recipes"[2] is created with the values "username" and "userpass". Note that in addition to being saved in a secure network location, with an innocuous dataset name, the value of "userpass" is also encoded using PROC PWENCODE to keep passwords safe from "casual, non-malicious viewing…" (SAS, 2011).

```
PROC PWENCODE IN='mysecretpassword' METHOD=SAS002;
RUN;
```

Once the PROC PWENCODE step is executed, the encoded version of the password is output to the LOG—with the encoding method ({sas002}) indicated as part of the encoded password.

[1] Note that if user "chris" has the Oracle "drop table" privilege assigned in the database, deleting the table through the LIBNAME does drop the table from the Oracle database, so be careful and make sure you understand your user privileges on the database.

[2] Admittedly, attempting to hide content with an innocuous filename is a pretty weak security measure, but it is combined here with your network security as a last line of defense to make this information less obvious than a dataset named (say) "PASSWORDS".

```
99   PROC PWENCODE IN=XXXXXXXXXXXXXXXXXX METHOD=SAS002;
100  RUN;

{sas002}5FAE9D593AF70EB951C670803B44F19538CDCDB301E8A357563480B0

NOTE: PROCEDURE PWENCODE used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

The encoded password is then used as the value of "userpass" in the "recipes" dataset.

```
LIBNAME safe '\\research01\clincore\users\cschacherer';

DATA safe.recipes;
 username='Chris';
 userpass='''{sas002}75FF5D504546D3563DC8361D092F20F22FC8909B''';
RUN;
```

With this dataset saved in your personal network directory (which in this case is mapped to "F:\" for everyone in the organization), future SAS programs can include the following code to assign the values of the "userid" and "password" macro variables.

```
LIBNAME safe 'F:\';

PROC SQL NOPRINT;
 SELECT username INTO :userid
   FROM safe.recipes;
 SELECT userpass INTO :password
   FROM safe.recipes;
QUIT;
```

Now, when the LIBNAME statement is executed, the connection to Oracle is always made with the credentials of the person executing the SAS program.

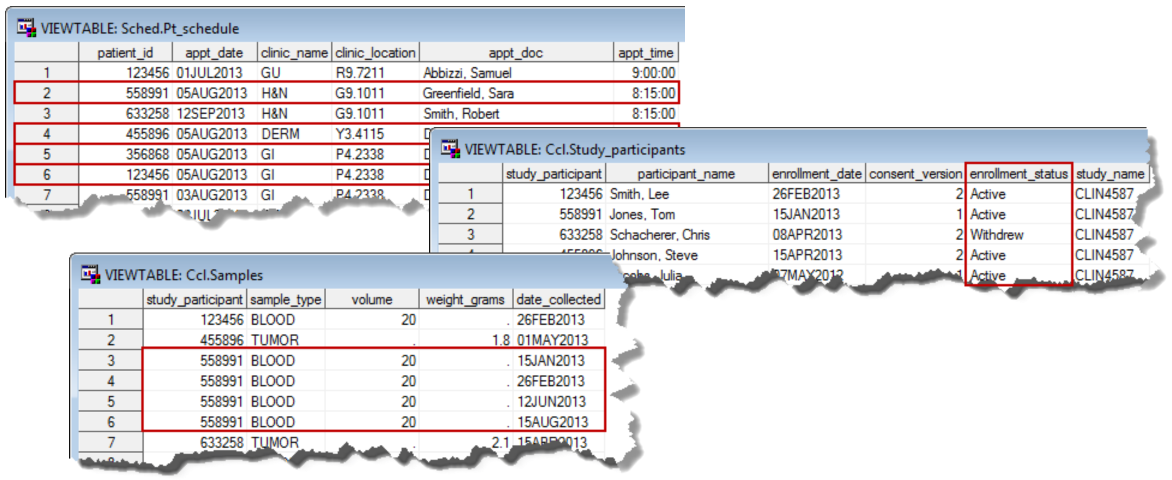
```
LIBNAME ccl ORACLE
   USER = &userid
   PATH = "research"
 SCHEMA = clincore
     PW = &password;
```

With access to the databases established, it is a fairly easy task to identify those patients on study "CLIN4587" that (a) will be coming to the hospital today and (b) have not yet provided the maximum allowed volume of blood samples over the course of their treatment. In the following code PROC SQL is used to join data from the ClinicalCoreLab (CCL) database and the hospital scheduling system to arrive at a list of patients who meet the following criteria: (a) the study participant has an appointment scheduled today, (b) they are currently enrolled on study "CLIN4587", and (c) they have not yet contributed the maximum allowable blood volume specified in the research protocol (80 ml). [For a more in-depth treatment of PROC SQL, see Lafler (2003, 2004, 2005) and Schacherer & Detry (2010).]

```
PROC SQL;
 CREATE TABLE work_queue AS
   SELECT a.study_participant AS patient_id,a.participant_name,
          a.enrollment_date,a.consent_version,
          c.clinic_name,c.clinic_location,c.appt_doc,c.appt_time FORMAT TIMEAMPM.,
          SUM(b.volume) AS current_volume
     FROM ccl.study_participants a,
          ccl.samples b,
          sched.pt_schedule c
    WHERE a.study_name = 'CLIN4587' AND
          a.enrollment_status='Active' AND
          a.study_participant = b.study_participant AND
          b.study_participant = c.patient_id AND
          c.appt_date="&sysdate"d
```

```
      GROUP BY a.study_participant,a.enrollment_date,a.consent_version,
               c.clinic_name,c.clinic_location,c.appt_doc,c.appt_time
        HAVING SUM(b.volume) < 80;
    QUIT;
```

Prior to this solution, study coordinators had to manually cross-reference information from these two systems, apply all of the logical constraints necessary to derive their daily work-list.



VIEWTABLE: Sched.Pt_schedule

	patient_id	appt_date	clinic_name	clinic_location	appt_doc	appt_time
1	123456	01JUL2013	GU	R9.7211	Abbizzi, Samuel	9:00:00
2	558991	05AUG2013	H&N	G9.1011	Greenfield, Sara	8:15:00
3	633258	12SEP2013	H&N	G9.1011	Smith, Robert	8:15:00
4	455896	05AUG2013	DERM	Y3.4115	[illegible]	[illegible]
5	356868	05AUG2013	GI	P4.2338	[illegible]	[illegible]
6	123456	05AUG2013	GI	P4.2338	[illegible]	[illegible]
7	558991	03AUG2013	GI	P4.2338	[illegible]	[illegible]

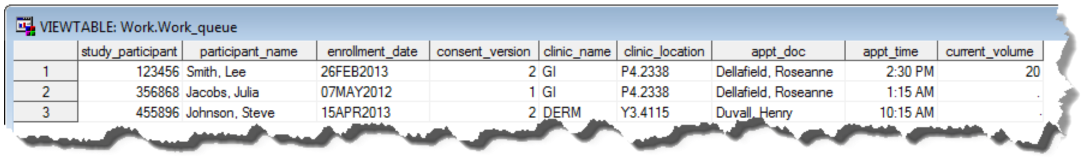
VIEWTABLE: Ccl.Study_participants

	study_participant	participant_name	enrollment_date	consent_version	enrollment_status	study_name
1	123456	Smith, Lee	26FEB2013	2	Active	CLIN4587
2	558991	Jones, Tom	15JAN2013	1	Active	CLIN4587
3	633258	Schacherer, Chris	08APR2013	2	Withdrew	CLIN4587
[illegible]	[illegible]	Johnson, Steve	15APR2013	2	Active	CLIN4587
[illegible]	[illegible]	[illegible] Julia	07MAY2012	1	Active	CLIN4587

VIEWTABLE: Ccl.Samples

	study_participant	sample_type	volume	weight_grams	date_collected
1	123456	BLOOD	20	.	26FEB2013
2	455896	TUMOR	.	1.8	01MAY2013
3	558991	BLOOD	20	.	15JAN2013
4	558991	BLOOD	20	.	26FEB2013
5	558991	BLOOD	20	.	12JUN2013
6	558991	BLOOD	20	.	15AUG2013
7	633258	TUMOR	.	2.1	15APR2013

Now, the SAS program easily creates a work-list for the study coordinator to use in organizing his/her work-day.



VIEWTABLE: Work.Work_queue

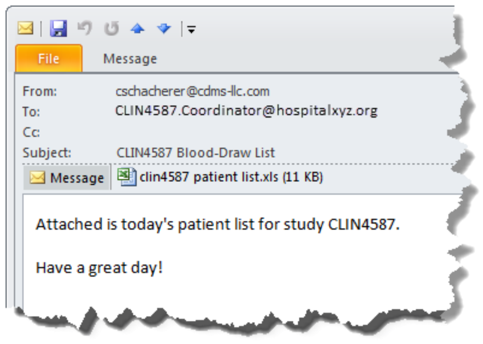
	study_participant	participant_name	enrollment_date	consent_version	clinic_name	clinic_location	appt_doc	appt_time	current_volume
1	123456	Smith, Lee	26FEB2013	2	GI	P4.2338	Dellafield, Roseanne	2:30 PM	20
2	356868	Jacobs, Julia	07MAY2012	1	GI	P4.2338	Dellafield, Roseanne	1:15 AM	.
3	455896	Johnson, Steve	15APR2013	2	DERM	Y3.4115	Duvall, Henry	10:15 AM	.

The program then outputs this list to an Excel® file, and it is ready to be e-mailed to the study coordinator. For more information about how to create e-mail distributions using the FILENAME statement, please refer to Schacherer (2012a), DeGuire (2007), and Tilanus (2008).

```
    PROC EXPORT DATA= WORK.WORK_QUEUE
                OUTFILE= "C:\clin4587 patient list.xls"
                DBMS=EXCEL REPLACE;
         SHEET="clin4587";
    RUN;

    FILENAME pt_list EMAIL;

    DATA _NULL_;
     FILE pt_list     to=" CLIN4587.Coordinator@hospitalxyz.org"
                  subject="CLIN4587 Blood-Draw List"
                   attach="C:\monthly claims report.pdf";
     PUT "Attached is today's patient list for study CLIN4587.";
     PUT " ";
     PUT "Have a great day!";
    RUN;
```

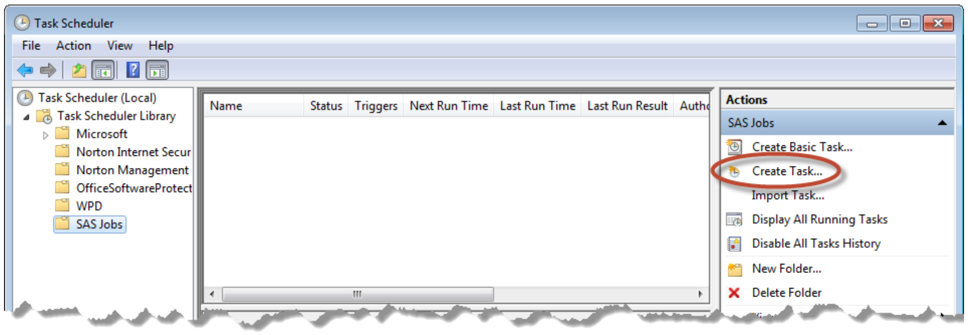


From: cschacherer@cdms-llc.com
To: CLIN4587.Coordinator@hospitalxyz.org
Cc:
Subject: CLIN4587 Blood-Draw List
Message | clin4587 patient list.xls (11 KB)

Attached is today's patient list for study CLIN4587.

Have a great day!

With the e-mail delivery of the daily work-list complete, the last step in automating this otherwise manual process is to schedule the program to be executed daily. This can be accomplished with the Windows® Task Scheduler (Schacherer, 2012b). In Windows 7, to launch the Task Scheduler, select Start►All Programs►Administrative Tools►Task Scheduler and click "Create Task".


Task Scheduler

File Action View Help

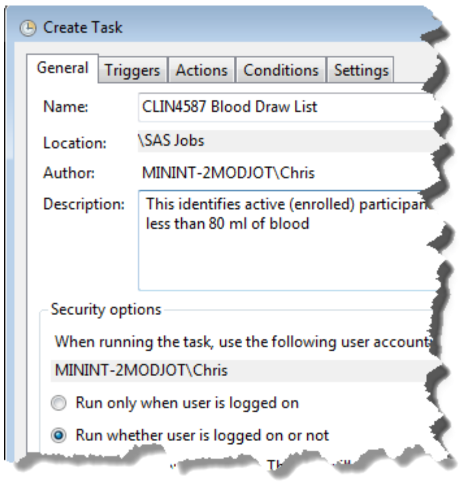
Task Scheduler (Local)
- Task Scheduler Library
  - Microsoft
  - Norton Internet Secur
  - Norton Management
  - OfficeSoftwareProtect
  - WPD
  - SAS Jobs

Name	Status	Triggers	Next Run Time	Last Run Time	Last Run Result	Autho

Actions

SAS Jobs
- Create Basic Task...
- Create Task...
- Import Task...
- Display All Running Tasks
- Disable All Tasks History
- New Folder...
- Delete Folder

After giving your new task a name and description go to the "Triggers" tab and specify the schedule on which you would like to run your program (e.g., weekly, daily, etc.).


Create Task

General | Triggers | Actions | Conditions | Settings

Name: CLIN4587 Blood Draw List

Location: \SAS Jobs

Author: MININT-2MODJOT\Chris

Description: This identifies active (enrolled) participan less than 80 ml of blood

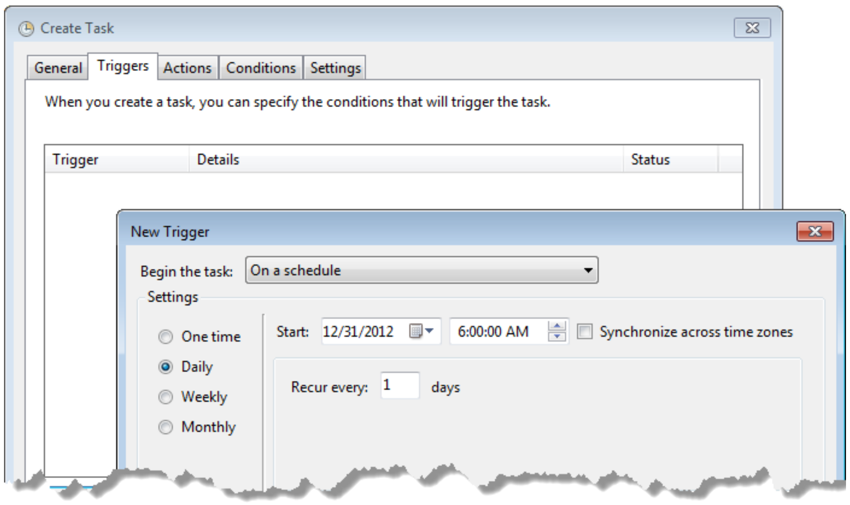
Security options

When running the task, use the following user account

MININT-2MODJOT\Chris

( ) Run only when user is logged on

[✓ Run whether user is logged on or not]


Create Task

General | Triggers | Actions | Conditions | Settings

When you create a task, you can specify the conditions that will trigger the task.

Trigger	Details	Status

New Trigger

Begin the task: On a schedule

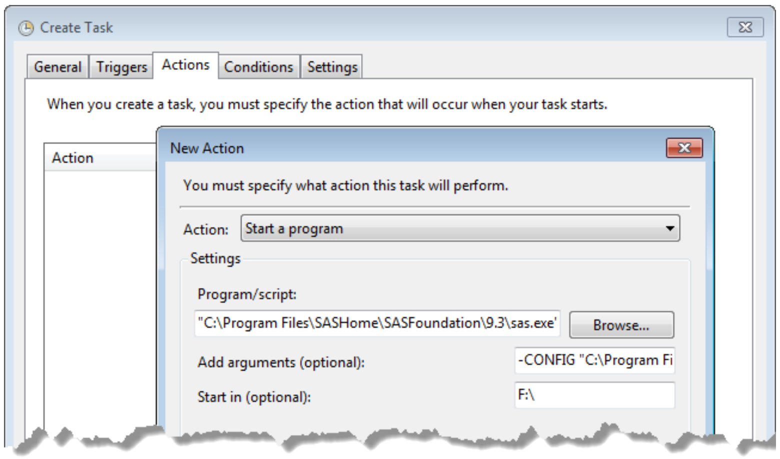
Settings

( ) One time [✓ Daily] ( ) Weekly ( ) Monthly

Start: 12/31/2012 6:00:00 AM ☐ Synchronize across time zones

Recur every: 1 days

In the "Actions" tab (below), you specify that you want the scheduled task to run "SAS.exe" and pass arguments specifying (a) the SAS configuration file you wish to use (-CONFIG argument) and (b) the SAS program to execute (-SYSIN argument). In this example the default SAS configuration file is specified and the –SYSIN argument directs the task to the program "CLIN4587 Blood Draws.sas" located in the "F:\" drive.


Create Task

General | Triggers | Actions | Conditions | Settings

When you create a task, you must specify the action that will occur when your task starts.

Action

New Action

You must specify what action this task will perform.

Action: Start a program

Settings

Program/script: "C:\Program Files\SASHome\SASFoundation\9.3\sas.exe" Browse...

Add arguments (optional): -CONFIG "C:\Program Fi

Start in (optional): F:\

-CONFIG "C:\Program Files\SASHome\SASFoundation\9.3\nls\en\SASV9.CFG"
-SYSIN "CLIN4587 Blood Draws.sas "

A similar model has been used to automate many other tasks as well. For example, by connecting to the surgery scheduling system, the research team can identify specific surgeries being performed for the treatment of a particular diagnosis and use web services calls from within SAS (Schacherer, 2012a) to check whether the patient has given permission to the use of their tissue for research purposes. As patients fitting different protocol-specific criteria are identified, e-mails and/or web-service calls can then be used to request tissue samples from the Organizational Tissue Banking System (OTBS). Similarly, data coordinators can be alerted when patients undergo exams in clinics that may indicate a possible comorbidity that needs to be documented, or be alerted to changes in a study participant's approval for the use of their tissue in research.

PROVIDING STUDY PERSONNEL WITH THE POWER TO KNOW^{TM}

The SAS programming resources working with the research program thoroughly enjoyed developing these previously-described operational support tools. However, there was also a significant (almost overwhelming) need to provide a wide variety of ad-hoc data extractions to support the individual research projects being conducted. When the research program began use of the ClinicalCoreLab database to capture clinical data and track tissue samples most investigators were thrilled just to be able to capture data in a consistent, integrated manner for the first time. However, as investigators began to realize the power of these data to answer more and more sophisticated questions, the number and complexity of the ad-hoc data requests began to grow. Pretty soon the requests for analytical support far out-paced the available hours anticipated for SAS programming resources. The fact that demand far exceeded capacity was proof that the CCL database had successfully hit the mark with regard to the design and granularity of the data being captured, but now the problem became how to scale operations to meet the demand for SAS analytics and unleash the power of these data to drive research.

The solution involved using Enterprise Guide® to empower other members of the research team to be able to fulfill the majority of requests for information from the CCL database. Previous attempts to expand SAS programming capabilities had met with varying levels of success. Initially in response to this increased demand for analytics support, data coordinators were provided traditional training in programming within SAS Display Manager. The advantage of this approach was that these individuals (unlike other available full-time SAS programmers) were intimately familiar with the data. They knew what they needed to accomplish with the data, and all they needed were the tools to manipulate the data in the desired manner. By providing them with some introductory SAS training, it was thought, they could quickly and easily help reduce the burden on more experienced analysts. Schacherer & Westra (2010) provide an in-depth discussion of training SAS programmers in healthcare analytics and the core principles were applied to training clinical research staff.

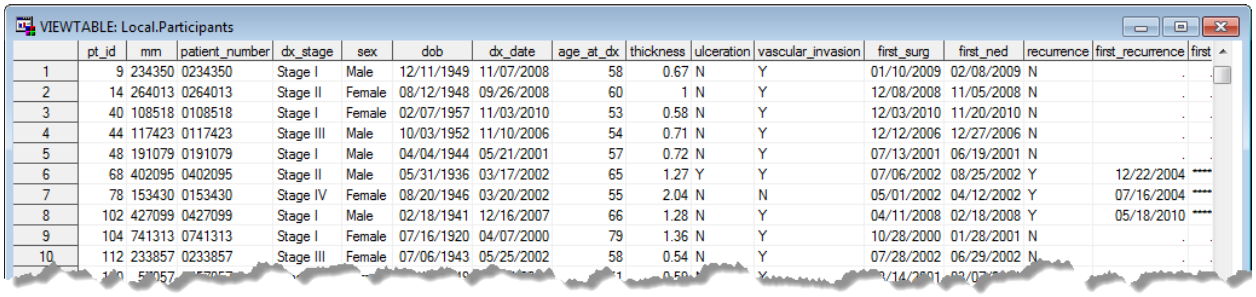
The down-side of this approach, of course, is that some individuals ended up struggling with the programming involved in fulfilling a given request long past the point where it was productive to have them working on it. Others ended up being fabulous SAS programmers that enjoyed that work much more than the work to which they were primarily assigned. In some cases this benefitted the research program and the organization as a whole, but it was not the desired outcome of fulfilling more data requests in a shorter time.

After struggling with these issues for some time, the research program implemented a solution utilizing SAS Enterprise Guide® to help strike the balance between fast answers to a subset of their analytic questions and developing programming knowledge among the existing staff. The solution empowered the staff to get answers to the questions they faced day-to-day, exposed them to SAS programming concepts (providing them an opportunity to grow their analytic skills) but, most importantly, reduced the time necessary to fulfill data extract and tissue sample requests—from weeks, in some cases, to hours or minutes. The following simplified description of the solution provides a brief introduction to Enterprise Guide and describes some of the tasks that all members of the research team are now performing with ease. For a more in-depth discussion about SAS Enterprise Guide, see Schacherer (2013) and Slaughter & Delwiche (2010).

CANNED DATASETS AND THE ENTERPRISE GUIDE ENVIRONMENT

The first step in enabling this solution was to create canned datasets that were capable of answering a host of different clinical and translational questions. The CCL database has a fairly complex structure consisting of nearly 100 tables of data that can be used to summarize a given patient's disease, treatment, and follow-up history across a number of dimensions. Working with several research program investigators, a dataset of clinical data elements was defined with the goal of being able to fulfill a significant number of requests for research data through simple queries of this single dataset ("participants"). If an investigator simply needed to know demographic information, primary lesion prognostic factors, or survival times between different end points, anyone on the research team would be able fulfill the request without utilizing the finite SAS programming resource. A simplified mock-up of such a dataset is depicted below[3].

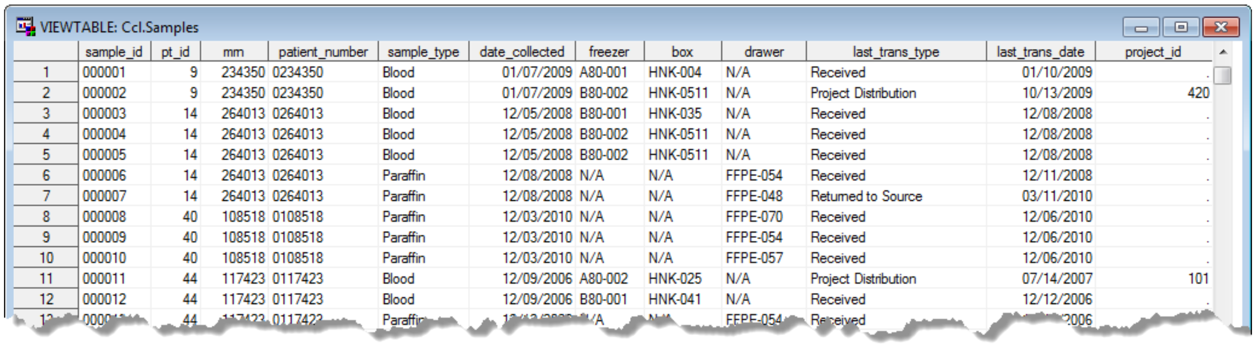
[3] Please Note: All data depicted here and elsewhere in the paper were generated by a DATA step using various random numbers, seed dates, and conditional logic to assign classifications. Any match to an actual healthcare record is unintentional (and one would think, quite unlikely).


VIEWTABLE: Local.Participants

	pt_id	mm	patient_number	dx_stage	sex	dob	dx_date	age_at_dx	thickness	ulceration	vascular_invasion	first_surg	first_ned	recurrence	first_recurrence	first
1	9	234350	0234350	Stage I	Male	12/11/1949	11/07/2008	58	0.67	N	Y	01/10/2009	02/08/2009	N	.	.
2	14	264013	0264013	Stage II	Female	08/12/1948	09/26/2008	60	1	N	Y	12/08/2008	11/05/2008	N	.	.
3	40	108518	0108518	Stage I	Female	02/07/1957	11/03/2010	53	0.58	N	Y	12/03/2010	11/20/2010	N	.	.
4	44	117423	0117423	Stage III	Male	10/03/1952	11/10/2006	54	0.71	N	Y	12/12/2006	12/27/2006	N	.	.
5	48	191079	0191079	Stage I	Male	04/04/1944	05/21/2001	57	0.72	N	Y	07/13/2001	06/19/2001	N	.	.
6	68	402095	0402095	Stage II	Male	05/31/1936	03/17/2002	65	1.27	Y	Y	07/06/2002	08/25/2002	Y	12/22/2004	****
7	78	153430	0153430	Stage IV	Female	08/20/1946	03/20/2002	55	2.04	N	N	05/01/2002	04/12/2002	Y	07/16/2004	****
8	102	427099	0427099	Stage I	Male	02/18/1941	12/16/2007	66	1.28	N	Y	04/11/2008	02/18/2008	Y	05/18/2010	****
9	104	741313	0741313	Stage I	Female	07/16/1920	04/07/2000	79	1.36	N	Y	10/28/2000	01/28/2001	N	.	.
10	112	233857	0233857	Stage III	Female	07/06/1943	05/25/2002	58	0.54	N	Y	07/28/2002	06/29/2002	N	.	.

Utilizing the canned "participants" dataset, if an investigator conducting a research study needs clinical or outcomes data, all he or she needs to do is submit a request via e-mail with a copy of the Institutional Review Board (IRB) authorization for the release of these data and any research data coordinator or research nurse on the team can fulfill the request.

The other canned dataset is the "samples" dataset. In addition to data elements describing each individual sample (type, storage location, etc.) data are also brought in from the tissue transactions subsystem of CCL so that lab technicians know the current status of each and every sample that is currently in (or has passed through) the research program's tissue bank.


VIEWTABLE: Ccl.Samples

	sample_id	pt_id	mm	patient_number	sample_type	date_collected	freezer	box	drawer	last_trans_type	last_trans_date	project_id
1	000001	9	234350	0234350	Blood	01/07/2009	A80-001	HNK-004	N/A	Received	01/10/2009	.
2	000002	9	234350	0234350	Blood	01/07/2009	B80-002	HNK-0511	N/A	Project Distribution	10/13/2009	420
3	000003	14	264013	0264013	Blood	12/05/2008	B80-001	HNK-035	N/A	Received	12/08/2008	.
4	000004	14	264013	0264013	Blood	12/05/2008	B80-002	HNK-0511	N/A	Received	12/08/2008	.
5	000005	14	264013	0264013	Blood	12/05/2008	B80-002	HNK-0511	N/A	Received	12/08/2008	.
6	000006	14	264013	0264013	Paraffin	12/08/2008	N/A	N/A	FFPE-054	Received	12/11/2008	.
7	000007	14	264013	0264013	Paraffin	12/08/2008	N/A	N/A	FFPE-048	Returned to Source	03/11/2010	.
8	000008	40	108518	0108518	Paraffin	12/03/2010	N/A	N/A	FFPE-070	Received	12/06/2010	.
9	000009	40	108518	0108518	Paraffin	12/03/2010	N/A	N/A	FFPE-054	Received	12/06/2010	.
10	000010	40	108518	0108518	Paraffin	12/03/2010	N/A	N/A	FFPE-057	Received	12/06/2010	.
11	000011	44	117423	0117423	Blood	12/09/2006	A80-002	HNK-025	N/A	Project Distribution	07/14/2007	101
12	000012	44	117423	0117423	Blood	12/09/2006	B80-001	HNK-041	N/A	Received	12/12/2006	.

The SAS program that produces each of these datasets is rerun daily and is saved to a team network drive that is mapped as Drive "S:\" on each team member's computer. Note that in more sophisticated SAS BI environments such datasets are more appropriately managed by registering them in a SAS Folder (Schacherer, 2013). However, as the research program already had access to PC SAS and Enterprise Guide through a Training and Research license they needed to see proof of the concept that an Enterprise Guide solution would fulfill their most urgent need of providing faster access to these data before investing in a larger scale solution. Fortunately, everyone on the research team was also eager to prove how this solution would allow them to show their increased value to the research program. What follows is a description of how Enterprise Guide was introduced to the team, and the types of work they performed to facilitate rapid access to these data.

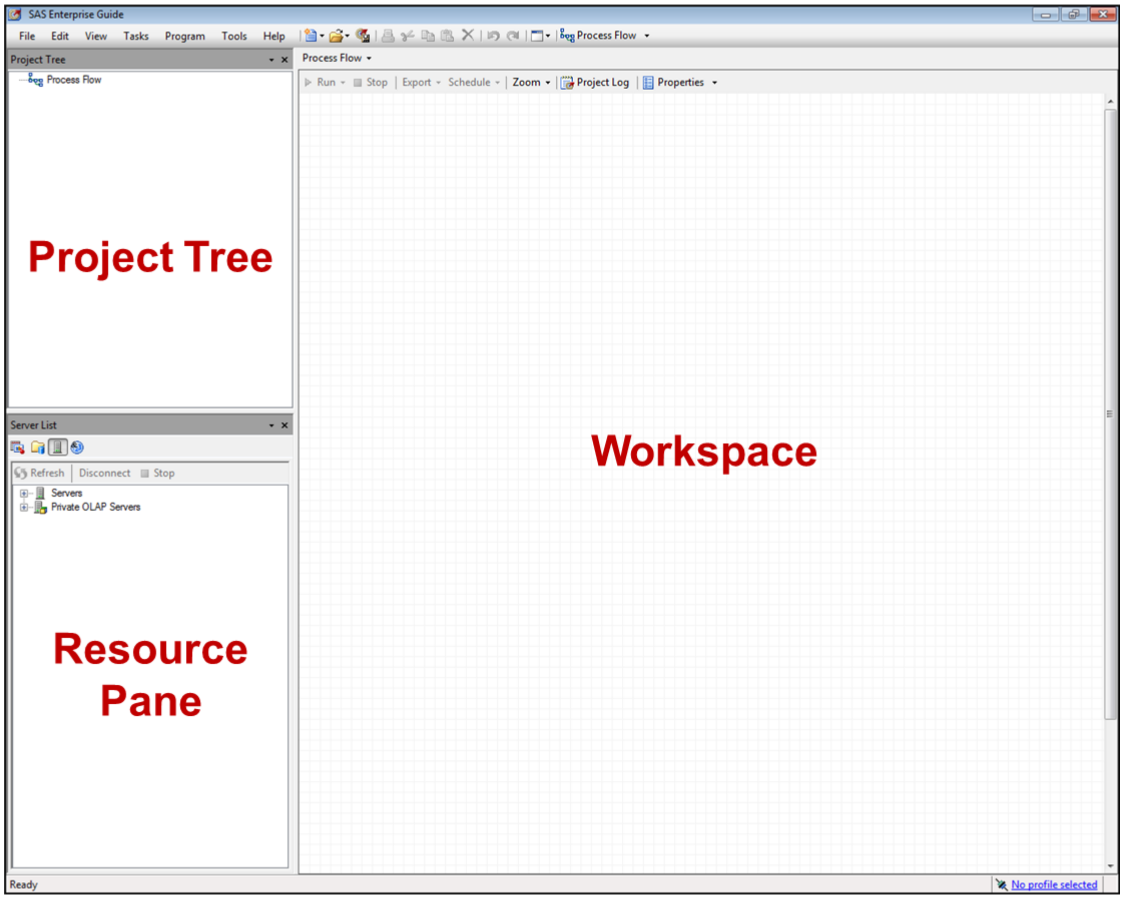
INTRODUCTION TO ENTERPRISE GUIDE

As discussed previously, most of these users had some previous introductory SAS training, but the success of that training to turn these individuals into SAS programmers naturally met with varying levels of success. First, all of these folks have other full-time work they need to perform—scheduling collection of specimens, consenting patients in the clinic, abstracting data from the medical record, etc. They were all capable of learning SAS programming and using it successfully, but expecting them to pick it up in their "spare" time was simply asking too much. What they needed was a point-and-click solution to accessing these data. What they needed was Enterprise Guide.

As described elsewhere (Slaughter & Delwich, 2010; Fecht & Dillon, 2011; Hallahan & Atkinson, 2006) at its core, Enterprise Guide is simply a graphical user interface to the SAS System. In many ways SAS Enterprise Guide plays the role of a SAS business analyst; via **Tasks** (graphical user interface widgets that each perform a different, specific type of processing) Enterprise Guide solicits information from the user about what type of data transformation, analysis, or reporting the user would like to have done to the data. Once this information is solicited, the task writes and submits the appropriate SAS code to the SAS processing engine for execution.

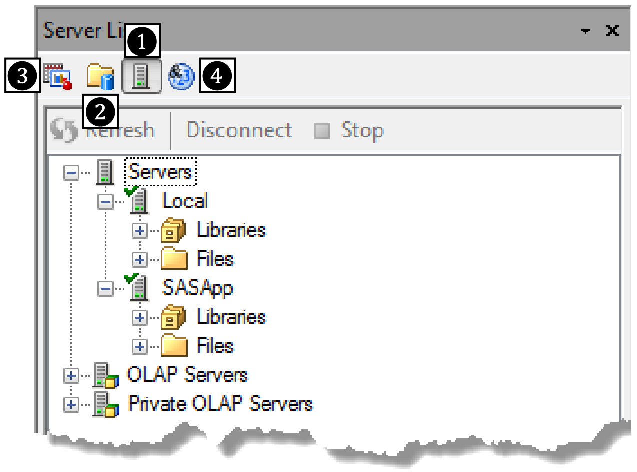
A natural starting point for learning Enterprise Guide is to explore the major components of the user interface. For most installations of SAS Enterprise Guide, the software can be launched from the Windows® Start Button and navigating to Programs►SAS►SAS Enterprise Guide.

Upon opening Enterprise Guide you are presented with a screen comprised of two docked windows (**Resource Pane** and **Project Tree**) and the **Workspace**—which technically is not a "window" since it cannot be closed, minimized, etc. independently from the application. The Workspace is where you will (well) work. In creating an Enterprise Guide **Project**[4], you will click and drag **Resources** from the Resource Pane into the Workspace. Some of these resources (**Tasks**) are tools used to manipulate other resources (**Datasets**) that are made available from within yet another resource (**Servers**). As these graphical user interface (GUI) widgets are dragged onto the **Process Flow** in the Workspace, they also appear in the Project Tree in a hierarchical arrangement that can make them a bit easier to find and manage as the Workspace fills with resources and output.



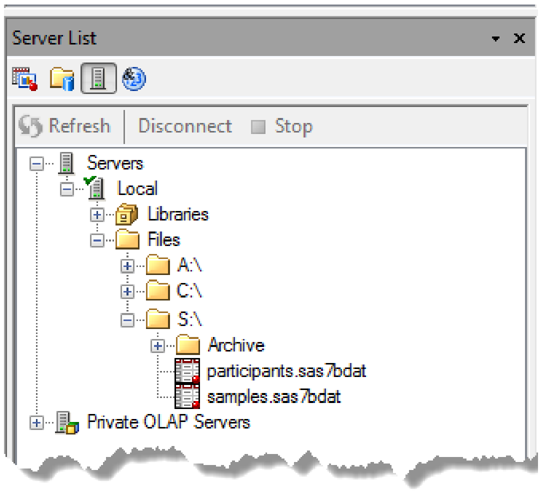
RESOURCE PANE.

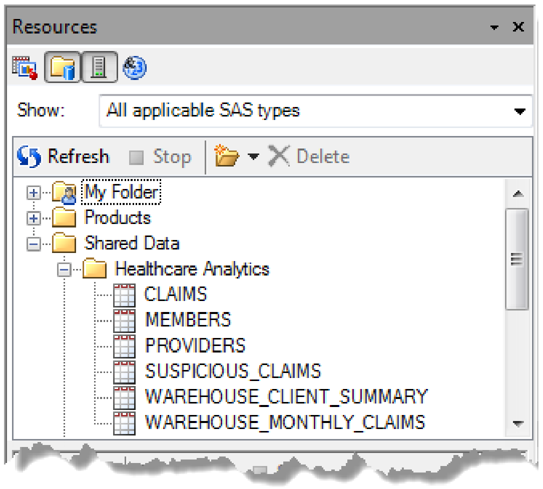
The Resource Pane, as its name suggests, organizes the many resources available to the user. The four main types of resources available to you are: ❶ Servers, ❷ SAS Folders, ❸ Tasks, and ❹ Prompts. In the adjacent screenshot, the user has access to both a "Local" server and a "SASApp" server. The latter is available only in environments that host SAS processing through a SAS server. The former indicates that there is a local installation of SAS on the workstation running Enterprise Guide. As discussed previously, Enterprise Guide is creating and submitting SAS code on your behalf. These SAS Servers are where that code is executed. In the examples that follow only the "Local" server is available to process the generated SAS code.



4 However, if you want to refer to it as a "program" most people will not argue with you—although technically, a "project" is different from a "program" (See Slaughter & Delwiche, 2010).

Expanding the "Files" on the "Local" **Server**, you can see that the "S:\" drive contains the two canned datasets that will be used by the research team. Because Local is the team member's own workstation, the drives to which one has access under "Files" are simply the drive letters currently assigned on that workstation. SAS Libraries can also be assigned on the Local server, and like using the local copy of SAS Display Manager, every time the system starts a temporary "WORK" library is established. Note: you must not forget to explain to new users that the WORK library is temporary and all SAS datasets created there are deleted at the end of the SAS session.

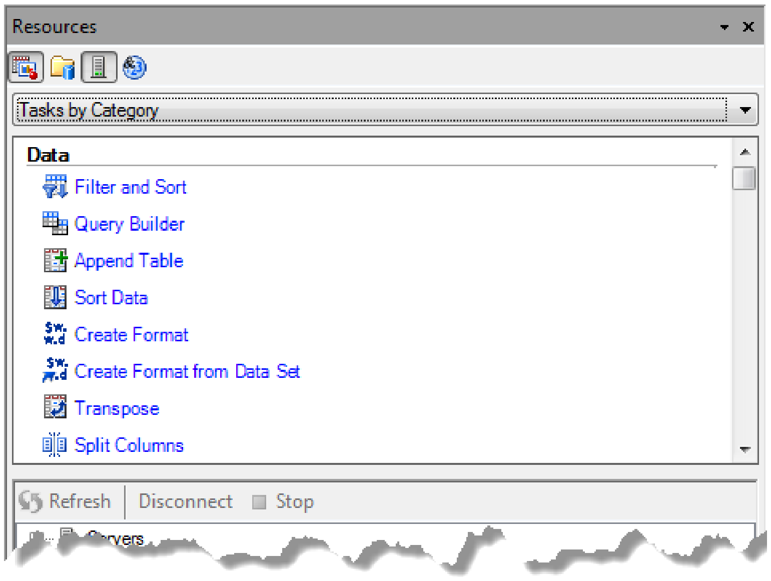




Show: All applicable SAS types

The adjacent screenshot shows the **SAS Folder** resource from a health insurance provider's SAS Workspace Server. Each of the "datasets" depicted in the Healthcare Analytics folder are actually metadata representations of data commonly used by the Healthcare Analytics team. The "Claims", "Members", and "Providers" tables are actually hosted on a Microsoft SQL Server database, the "Suspicious_Claims" dataset is actually a SAS dataset that resides on a network directory, and the "WAREHOUSE_" tables are hosted on the company's Oracle data warehouse. By representing them as SAS metadata, the SAS Folder allows a great deal of flexibility in organizing data assets logically rather than strictly by individual system. The present work does not delve into SAS Folders beyond this, but they are discussed in greater depth elsewhere (Aanderud & Hall, 2012; Schacherer, 2013).

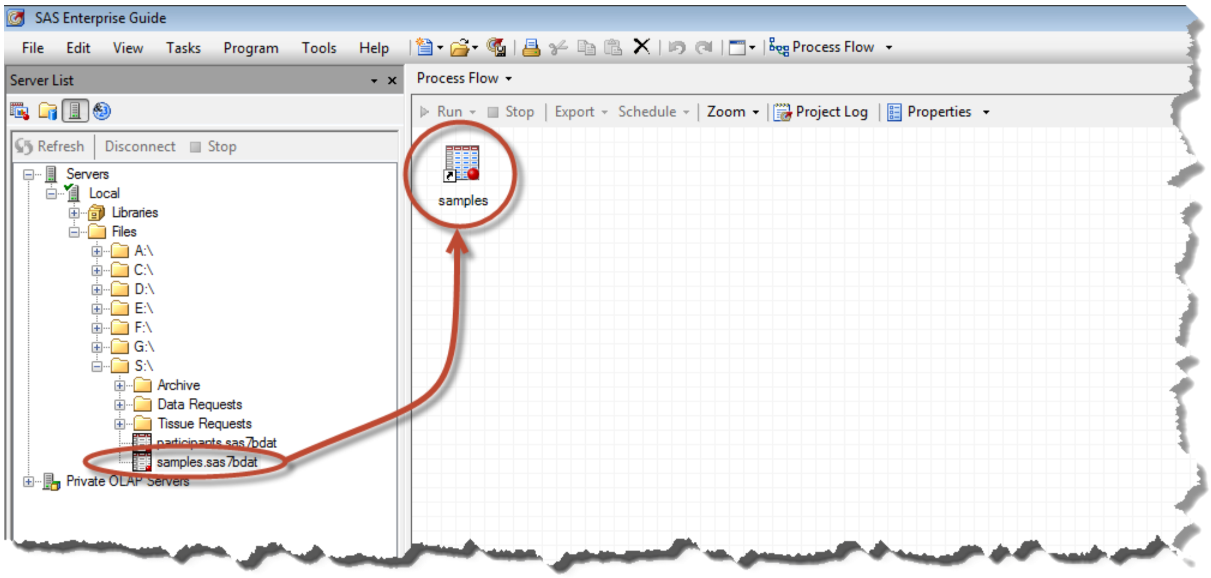
Enterprise Guide **Tasks** are specialized widgets used to perform specific types of operations—filtering and sorting data, creating bar-charts, performing t-tests, etc. Enterprise Guide tasks are not executable program units so much as they are user interfaces to those programming units. In fact, each Enterprise Guide task is a user-interface to a specific SAS PROCEDURE. By interacting with the task, what the user is doing is providing the information that the task needs in order to write a valid PROC step in the SAS programming language. Each task has its own dialog window and/or wizard that allows the user to specify the options applied to the execution of that task. To give an example of how tasks and data resources interact in an Enterprise Guide project, consider how a research team member solved the following common task.



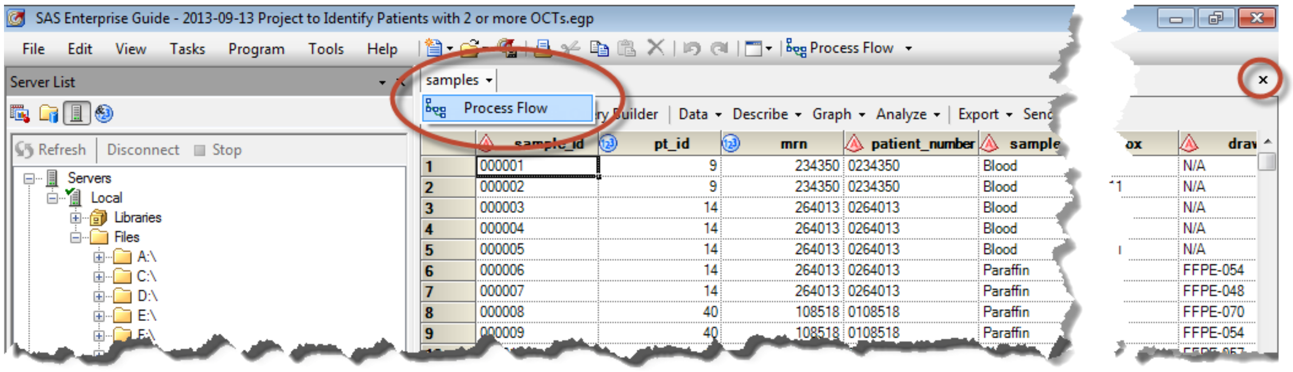
Tasks by Category

IDENTIFYING AVAILABLE TISSUE SAMPLES.

Tumor Biologist Joe Smith submits a request to identify the number of participants from whom the tissue bank has at least two OCT samples available in its holdings. If you were the lab technician fulfilling the request you would launch Enterprise Guide, navigate to the "S:\" drive and drag the "samples" dataset into the Process Flow in the Workspace.

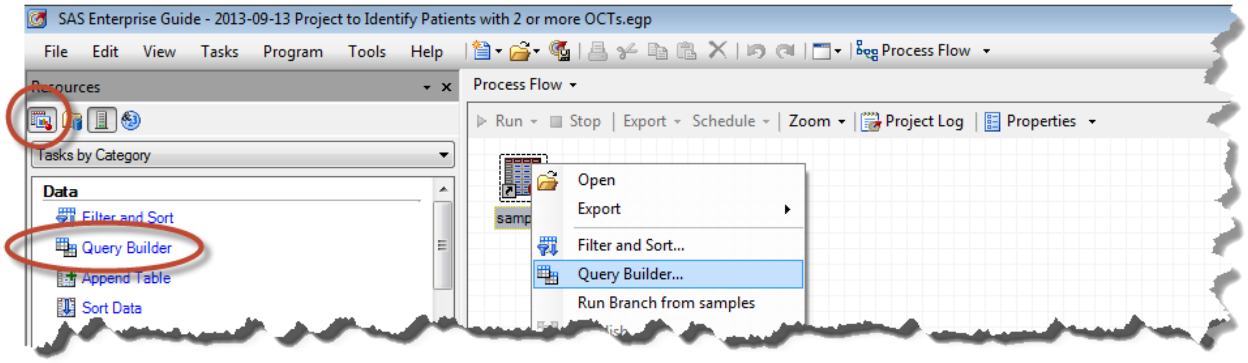


You can double-click on the dataset to see that it is the dataset described earlier. When you are ready to go back to the original "Process Flow" in which you were working, you can either navigate back by selecting the Process Flow name from the drop-down menu or by clicking the close button in the Document Window.

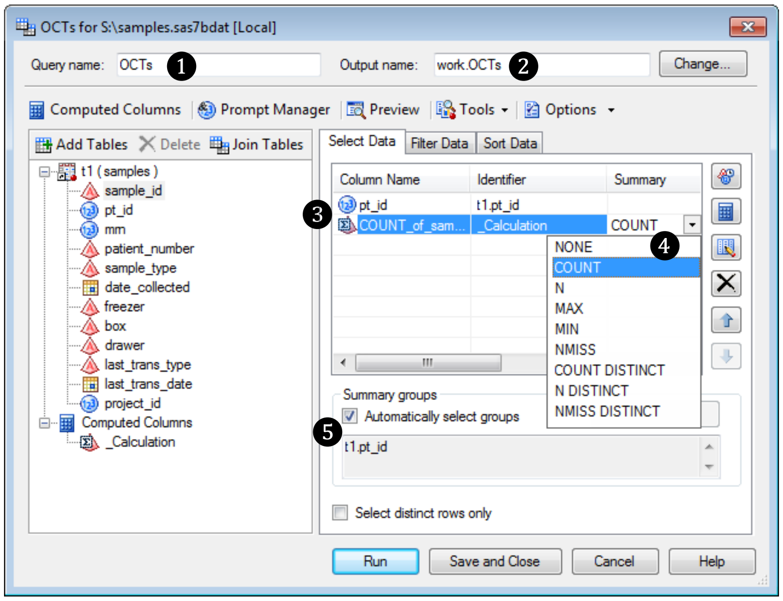

SAS Enterprise Guide - 2013-09-13 Project to Identify Patients with 2 or more OCTs.egp
File Edit View Tasks Program Tools Help
Process Flow
Server List
Refresh Disconnect Stop
Servers
Local
Libraries
Files
A:\
C:\
D:\
E:\
F:\
samples
Process Flow
ry Builder Data Describe Graph Analyze Export Send

	sample_id	pt_id	mrn	patient_number	sample	ox	draw
1	000001	9	234350	0234350	Blood		N/A
2	000002	9	234350	0234350	Blood	1	N/A
3	000003	14	264013	0264013	Blood		N/A
4	000004	14	264013	0264013	Blood		N/A
5	000005	14	264013	0264013	Blood		N/A
6	000006	14	264013	0264013	Paraffin		FFPE-054
7	000007	14	264013	0264013	Paraffin		FFPE-048
8	000008	40	108518	0108518	Paraffin		FFPE-070
9	000009	40	108518	0108518	Paraffin		FFPE-054

Next, either right mouse-click on the "samples" dataset and choose "Query Builder", or activate the Task List by clicking on the Task Resource button on the Resource Menubar and click on the "Query Builder" task in the Task List.


SAS Enterprise Guide - 2013-09-13 Project to Identify Patients with 2 or more OCTs.egp
File Edit View Tasks Program Tools Help
Process Flow
Resources
Tasks by Category
Data
Filter and Sort
Query Builder
Append Table
Sort Data
Process Flow
Run Stop Export Schedule Zoom Project Log Properties
samp
Open
Export
Filter and Sort...
Query Builder...
Run Branch from samples

The Query Builder task is one of the most heavily used tasks because of the many transformations that it can perform. Roughly 90%-95% of the research team's Enterprise Guide projects use the Query Builder task as a core component. In the current query, the lab tech needs to identify all participants with two or more OCT samples in the tissue bank holdings. First, the query will be given a name (OCTs) ❶ so it can be identified in the Process Flow. Next, the output dataset name will be assigned—in this case as the dataset "OCTs" in the "work" library ❷. After dragging the "pt_id" and "sample_id" variables onto the "Select Data" tab ❸, the function "Count" is selected from the Summary Functions drop-down list ❹, and the grouping variable is automatically assigned as "pt_id" ❺. Specified in this way, the query will provide one record for each unique participant in the "samples" dataset and will calculate the number of samples associated with each participant.



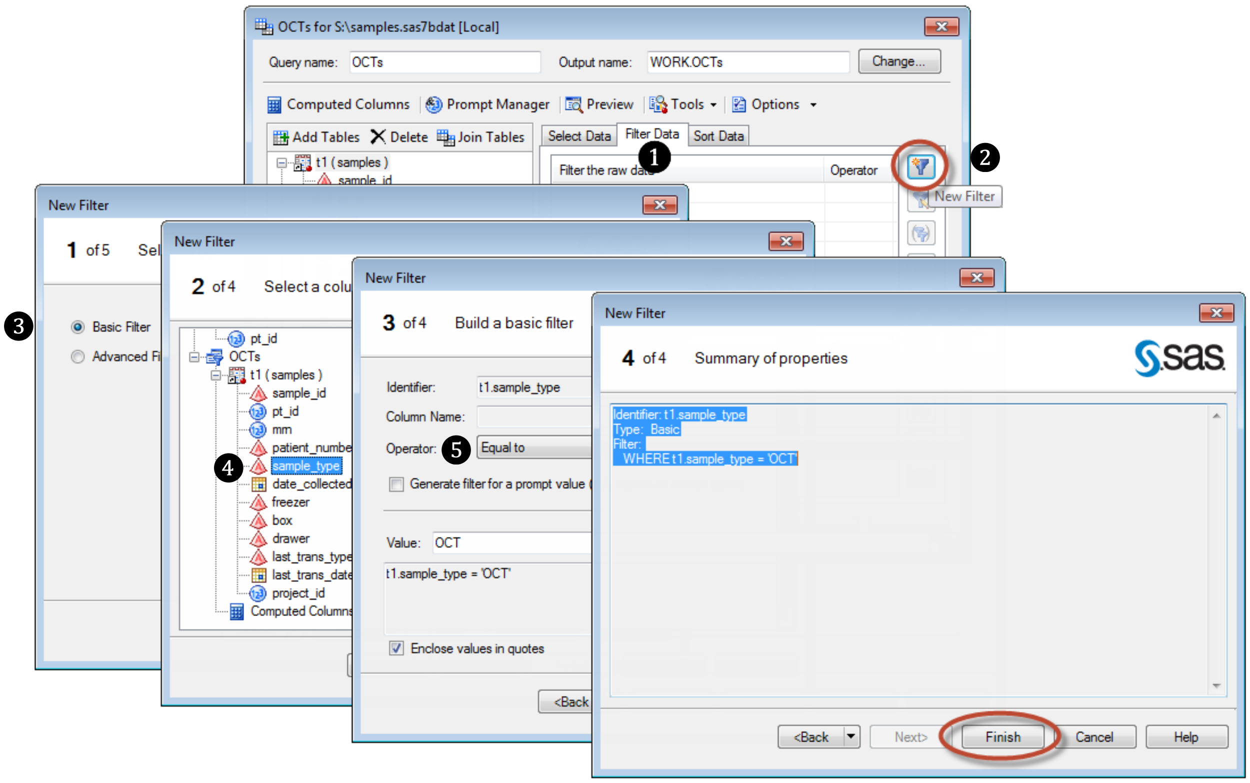
Query name: OCTs
Output name: work.OCTs

Column Name	Identifier	Summary
pt_id	t1.pt_id	
COUNT_of_sam...	_Calculation	COUNT

Summary groups
[✓] Automatically select groups
t1.pt_id

[ ] Select distinct rows only

But if you want to select only those participants with two or more OCT samples, you need to click on the "Filter Data" tab ❶ and create some filters on your query results. To start building a filter, click the "New Filter" button ❷. Like much of the functionality throughout Enterprise Guide, you are walked through the steps necessary to specify <u>how</u> you would like the associated work performed. In this case, a "Basic Filter" ❸ (one which relies only on the available data elements and values on the data record) will be built. In Step 2, select the "sample_type" as the variable by which to filter the data ❹, and in Step 3 use the Operator drop-down list ❺ to specify the type of comparison to make between the "sample_type" variable's value and the criterion value you provide. In this case only records with a "sample_type" of "OCT" will be retained in the resulting dataset. Step 4 demonstrates a step common to many Enterprise Guide tasks; before the task is executed, Enterprise Guide gives you a chance to review your specifications. When you have confirmed the criterion, click "Finish" to complete the Filter.

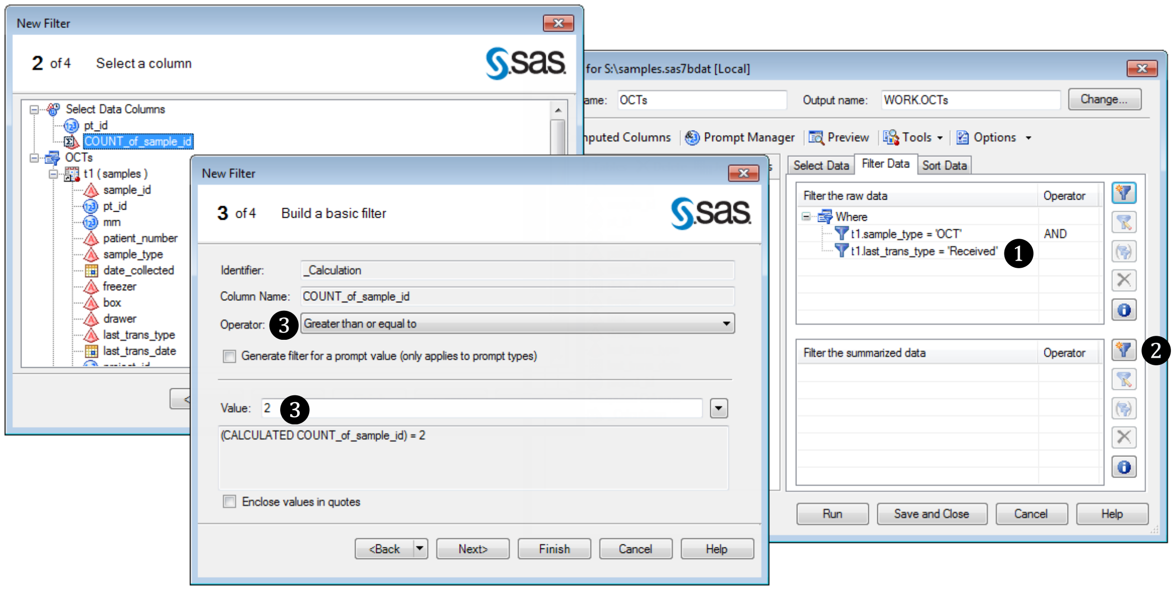


Query name: OCTs
Output name: WORK.OCTs

Identifier: t1.sample_type
Column Name:
Operator: Equal to
[ ] Generate filter for a prompt value
Value: OCT
t1.sample_type = 'OCT'
[✓] Enclose values in quotes

Identifier: t1.sample_type
Type: Basic
Filter:
WHERE t1.sample_type = 'OCT'

With the "sample_type" filter completed and another filter applied to make sure that we count only those samples currently in the inventory ❶, we need to add a filter on the summary statistic ❷. Note that summary statistic filters will only be available in the Filter tab when you have specified a summary condition in the "Select Data" tab. In this case we will specify the condition that a summarized record is only to be returned to the dataset in those cases where at least two samples are associated with the "pt_id" being summarized❸.


New Filter — 2 of 4 Select a column

Select Data Columns: pt_id, COUNT_of_sample_id
OCTs: t1 (samples): sample_id, pt_id, mm, patient_number, sample_type, date_collected, freezer, box, drawer, last_trans_type, last_trans_date

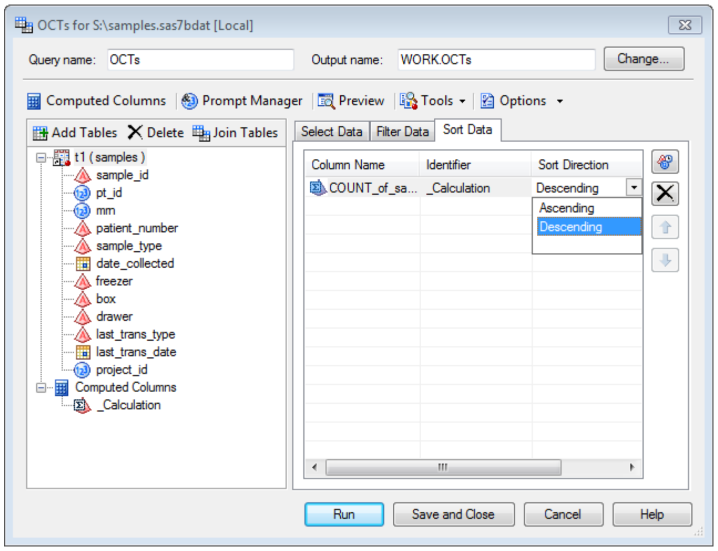
New Filter — 3 of 4 Build a basic filter

Identifier: _Calculation
Column Name: COUNT_of_sample_id
Operator: ❸ Greater than or equal to
Generate filter for a prompt value (only applies to prompt types)
Value: 2 ❸
(CALCULATED COUNT_of_sample_id) = 2
Enclose values in quotes

Filter the raw data — Operator
Where
t1.sample_type = 'OCT' AND
t1.last_trans_type = 'Received' ❶

Filter the summarized data — Operator ❷

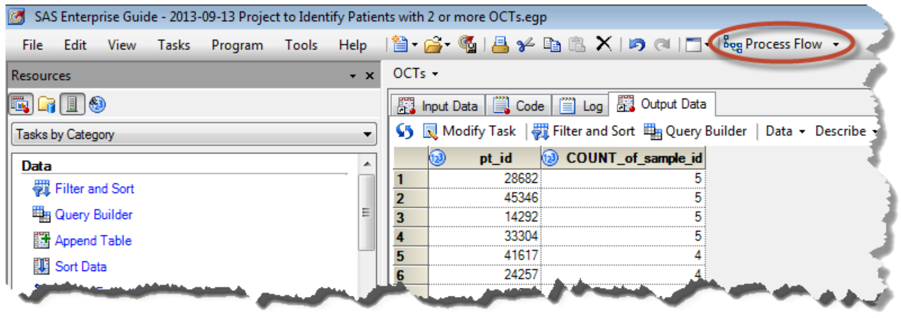
Finally, navigate to the "Sort Data" tab and specify that the resulting dataset (work.OCTs) should be sorted in descending order by the number of samples—with the highest counts at the top of the list and the lower (but still at least two or more) at the bottom of the list. When you are satisfied that the query has been specified to your precise criteria, click "Run".


OCTs for S:\samples.sas7bdat [Local]

Query name: OCTs Output name: WORK.OCTs

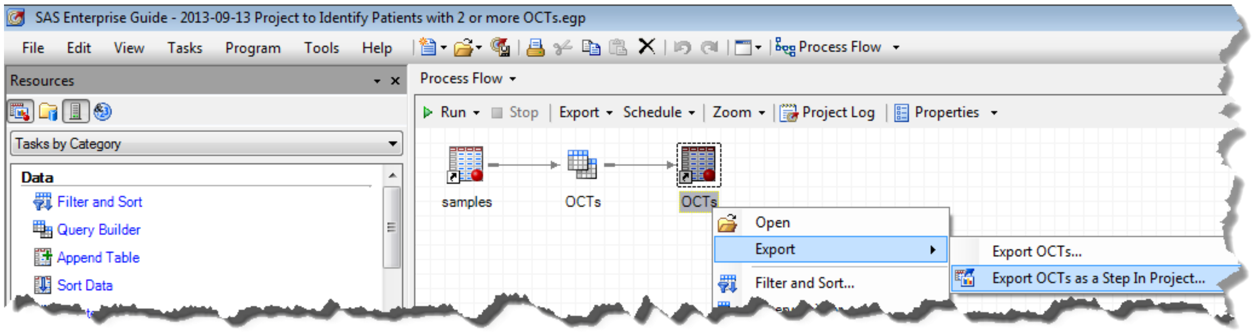
Column Name	Identifier	Sort Direction
COUNT_of_sa...	_Calculation	Descending

After the Query Builder task has run, the resulting dataset is rendered in a Document Window. To navigate back to the Process Flow again you can use the navigation drop-down or simply close the document window.

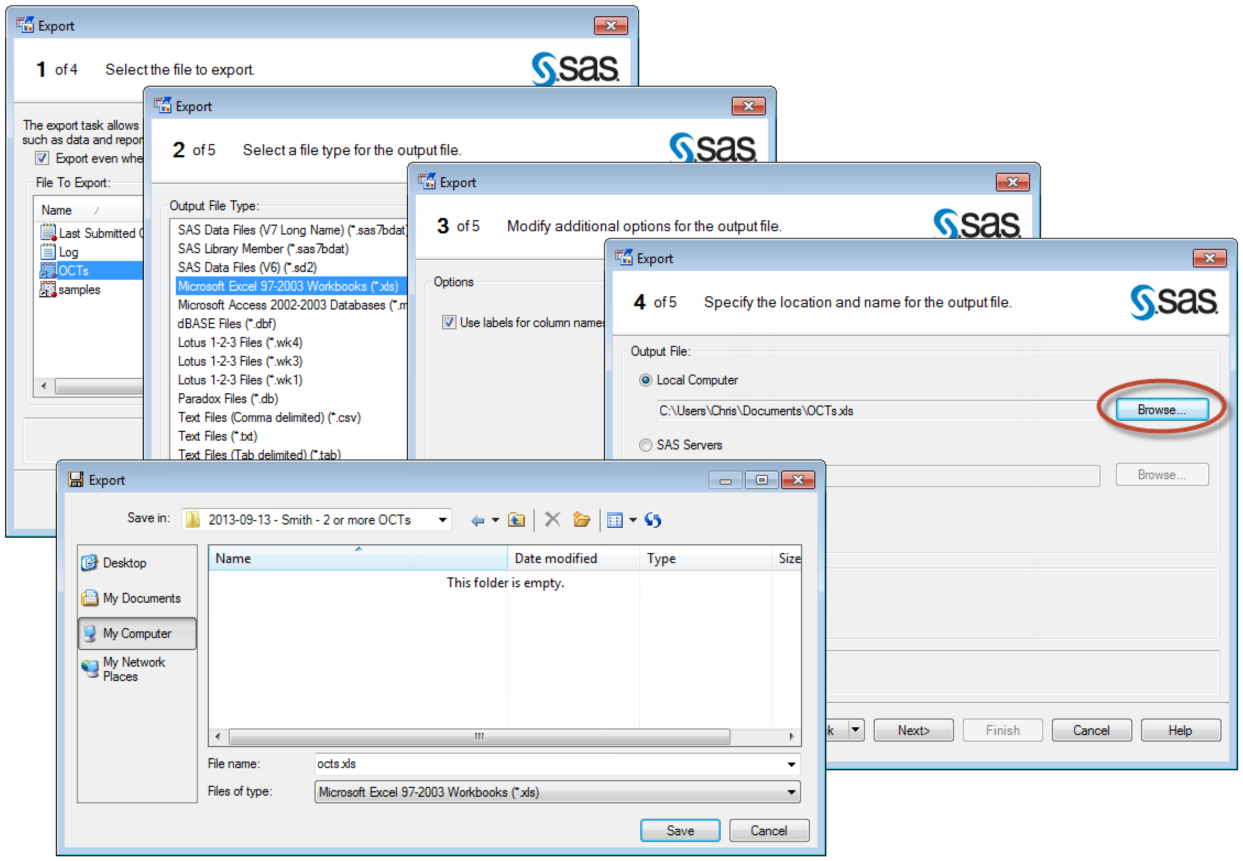

SAS Enterprise Guide - 2013-09-13 Project to Identify Patients with 2 or more OCTs.egp

	pt_id	COUNT_of_sample_id
1	28682	5
2	45346	5
3	14292	5
4	33304	5
5	41617	4
6	24257	4

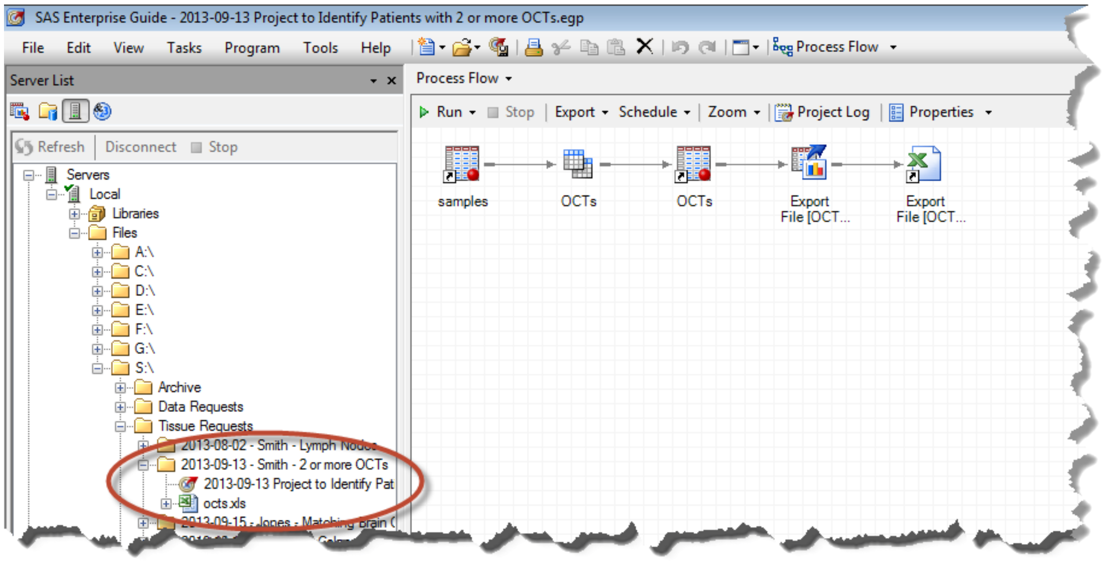
It is at this point that most new users really have an appreciation of why the workspace palettes on which the work is represented in Enterprise Guide are called "Process Flows". As you begin working with data, tasks, and output, Enterprise Guide makes the relationships between these objects explicit by linking them together in a kind of logical flow diagram. Here, the "samples" dataset is used as an input to the "OCTs" Query Builder task and the output of that task is the "OCTs" dataset. Because Dr. Smith wanted a file of the results sent to him, the OCTs dataset will be exported as an automated step in the project by right-clicking on the dataset and choosing "Export"►"Export OCTs as a Step In Project.


SAS Enterprise Guide - 2013-09-13 Project to Identify Patients with 2 or more OCTs.egp
File Edit View Tasks Program Tools Help
Process Flow
Resources
Tasks by Category
Data
Filter and Sort
Query Builder
Append Table
Sort Data
Run Stop Export Schedule Zoom Project Log Properties
samples
OCTs
OCTs
Open
Export
Filter and Sort...
Export OCTs...
Export OCTs as a Step In Project...

This opens the Export task where, again, Enterprise Guide walks through the steps necessary to elicit the information necessary to determine what to export, in what format, and where. In this case, the user browses to the "S:\Tissue Requests\2013-09-13 – Smith – 2 or more OCTs" directory and saves the dataset as "OCTs.xls"


Export
1 of 4 Select the file to export.
The export task allows such as data and repor
Export even whe
File To Export:
Name
Last Submitted
Log
OCTs
samples
Export
2 of 5 Select a file type for the output file.
Output File Type:
SAS Data Files (V7 Long Name) (*.sas7bdat)
SAS Library Member (*.sas7bdat)
SAS Data Files (V6) (*.sd2)
Microsoft Excel 97-2003 Workbooks (*.xls)
Microsoft Access 2002-2003 Databases (*.m
dBASE Files (*.dbf)
Lotus 1-2-3 Files (*.wk4)
Lotus 1-2-3 Files (*.wk3)
Lotus 1-2-3 Files (*.wk1)
Paradox Files (*.db)
Text Files (Comma delimited) (*.csv)
Text Files (*.txt)
Text Files (Tab delimited) (*.tab)
Export
3 of 5 Modify additional options for the output file.
Options
Use labels for column name
Export
4 of 5 Specify the location and name for the output file.
Output File:
Local Computer
C:\Users\Chris\Documents\OCTs.xls
Browse...
SAS Servers
Browse...
Next>
Finish
Cancel
Help
Export
Save in: 2013-09-13 - Smith - 2 or more OCTs
Desktop
My Documents
My Computer
My Network Places
Name
Date modified
Type
Size
This folder is empty.
File name: octs.xls
Files of type: Microsoft Excel 97-2003 Workbooks (*.xls)
Save
Cancel

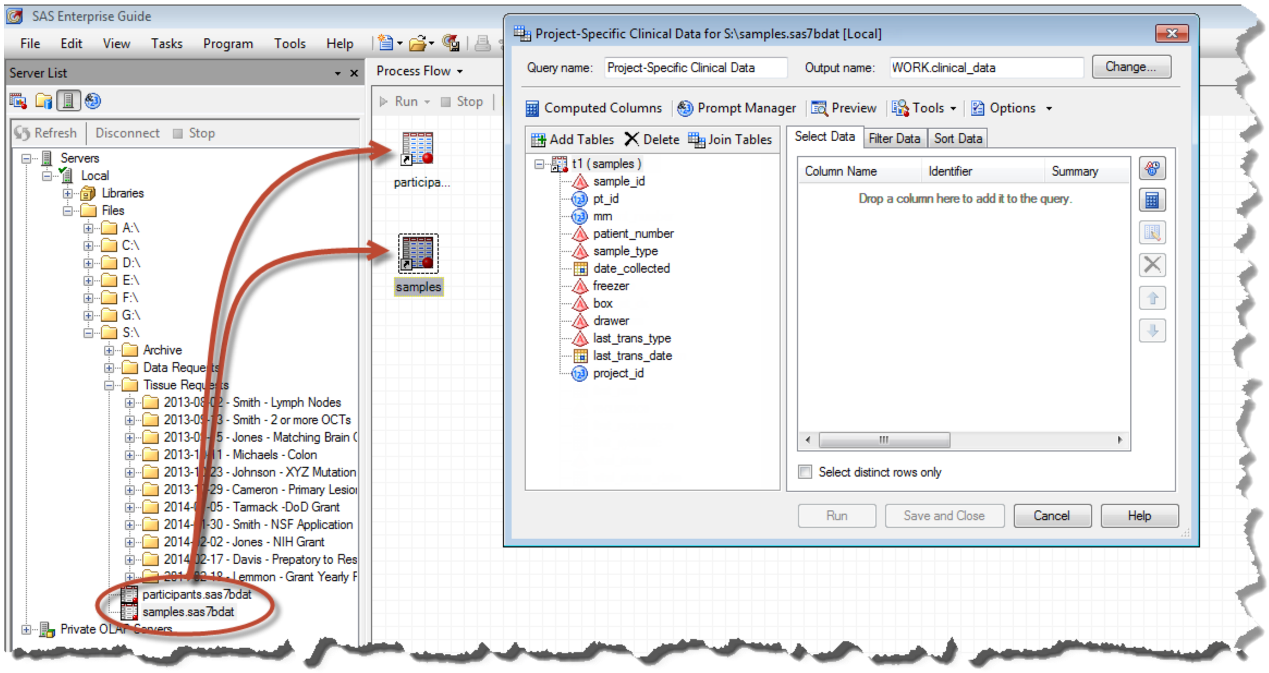
After finishing the Export task, the project will now be able to reproduce the desired dataset anytime Dr. Smith wants to revisit this question. The Enterprise Guide project will be saved to the project directory and at any time in the future that Dr. Smith wants this report rerun it will take just a few seconds to produce.


SAS Enterprise Guide - 2013-09-13 Project to Identify Patients with 2 or more OCTs.egp
File Edit View Tasks Program Tools Help
Process Flow
Server List
Refresh Disconnect Stop
Run Stop Export Schedule Zoom Project Log Properties
Servers
Local
Libraries
Files
A:\
C:\
D:\
E:\
F:\
G:\
S:\
Archive
Data Requests
Tissue Requests
2013-08-02 - Smith - Lymph Nodes
2013-09-13 - Smith - 2 or more OCTs
2013-09-13 Project to Identify Pat
octs.xls
2013-09-15 - Jones - Matching Brain C
samples
OCTs
OCTs
Export File [OCT...
Export File [OCT...

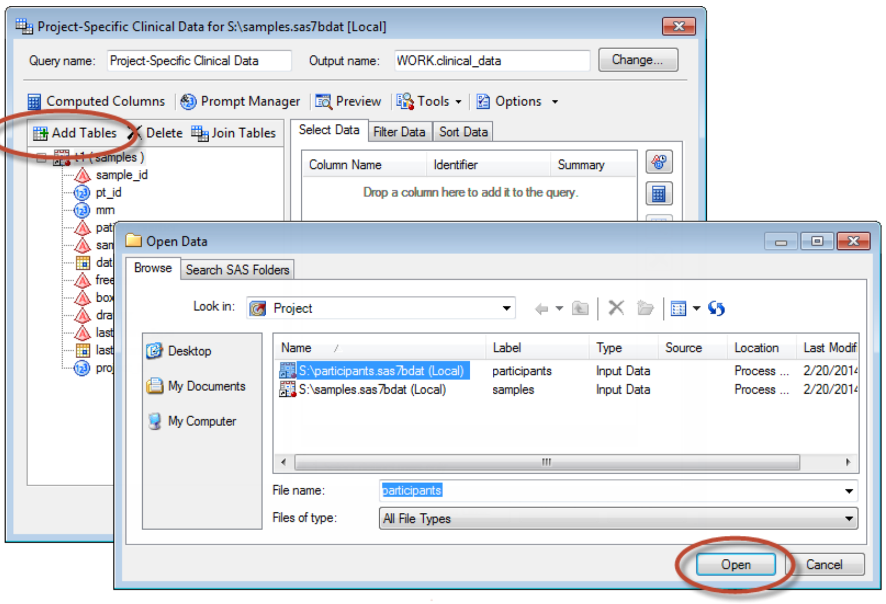
ON-DEMAND CLINICAL DATASETS.

As the number of requests for data kept rising across the entire team, certain requests started becoming very common place and the team wondered how they could further simplify these requests for data. One type of request in particular was becoming very common place. When investigators receive tissue for their lab studies they also want a dataset of the latest clinical data. Over time they will also want to continue to update these data as more information becomes available about significant clinical events that have occurred. Therefore, one solution that needed to be developed was a way to quickly and reliably produce clinical datasets for each individual research project to which tissue samples have been distributed.

This challenge introduced the team to the concept of joining data from different datasets using the Query Builder task and the Enterprise Guide Prompt (macro variables). This solution involves both the "samples" and "participants" datasets. First, a Query Builder task is launched using the "samples" dataset.


SAS Enterprise Guide
File Edit View Tasks Program Tools Help
Server List
Process Flow
Refresh Disconnect Stop
Run Stop
Servers
Local
Libraries
Files
A:\
C:\
D:\
E:\
F:\
G:\
S:\
Archive
Data Requests
Tissue Requests
2013-08-02 - Smith - Lymph Nodes
2013-09-13 - Smith - 2 or more OCTs
2013-09-15 - Jones - Matching Brain C
2013-10-11 - Michaels - Colon
2013-10-23 - Johnson - XYZ Mutation
2013-10-29 - Cameron - Primary Lesio
2014-01-05 - Tarmack -DoD Grant
2014-01-30 - Smith - NSF Application
2014-02-02 - Jones - NIH Grant
2014-02-17 - Davis - Prepatory to Res
2014-02-18 - Lemmon - Grant Yearly F
participants.sas7bdat
samples.sas7bdat
Private OLAP Servers
participa...
samples
Project-Specific Clinical Data for S:\samples.sas7bdat [Local]
Query name: Project-Specific Clinical Data
Output name: WORK.clinical_data
Change...
Computed Columns Prompt Manager Preview Tools Options
Add Tables Delete Join Tables
Select Data Filter Data Sort Data
t1 (samples)
sample_id
pt_id
mm
patient_number
sample_type
date_collected
freezer
box
drawer
last_trans_type
last_trans_date
project_id
Column Name Identifier Summary
Drop a column here to add it to the query.
Select distinct rows only
Run Save and Close Cancel Help

Next, the "participants" dataset is added to the query by clicking the "Add Tables" button, selecting the dataset from the "Open Data" dialog, and clicking "Open".



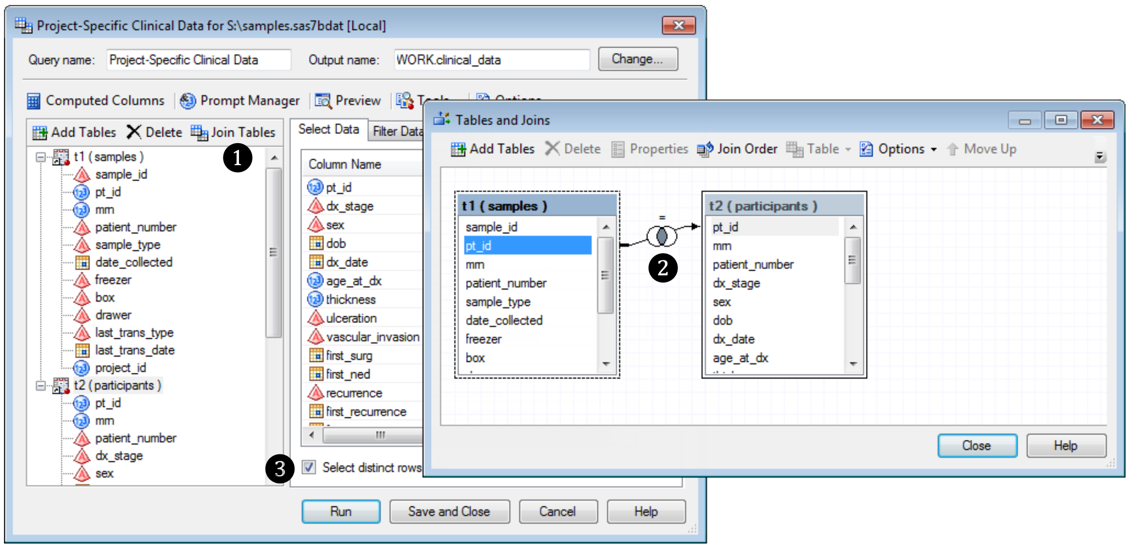
Query name: Project-Specific Clinical Data
Output name: WORK.clinical_data

Look in: Project

Name	Label	Type	Source	Location	Last Modif
S:\participants.sas7bdat (Local)	participants	Input Data		Process ...	2/20/2014
S:\samples.sas7bdat (Local)	samples	Input Data		Process ...	2/20/2014

File name: participants
Files of type: All File Types

With the "participants" dataset added to the query, the columns of clinical data are selected for inclusion in the resulting dataset and the criterion for joining the rows of data to one another (i.e., the "join condition") is specified by clicking on the "Join Tables" button ❶ and dragging the "pt_id" field from the samples dataset to the "participants" dataset ❷. This is actually one conceptual area that may take a few tries for new users to wrap their heads around. In most software people use, the software just "knows" how things are supposed to go together, so it is (for many people) a new concept that one needs to specify this relationship. In conjunction with describing the different types of join conditions to new users, you will likely want to include a discussion of the "Select Distinct Rows" criterion ❸ and the fact that if you were to not select this option, you would get a duplicate clinical record for every sample associated with a given participant.



Query name: Project-Specific Clinical Data
Output name: WORK.clinical_data

t1 (samples): sample_id, pt_id, mm, patient_number, sample_type, date_collected, freezer, box, drawer, last_trans_type, last_trans_date, project_id
t2 (participants): pt_id, mm, patient_number, dx_stage, sex

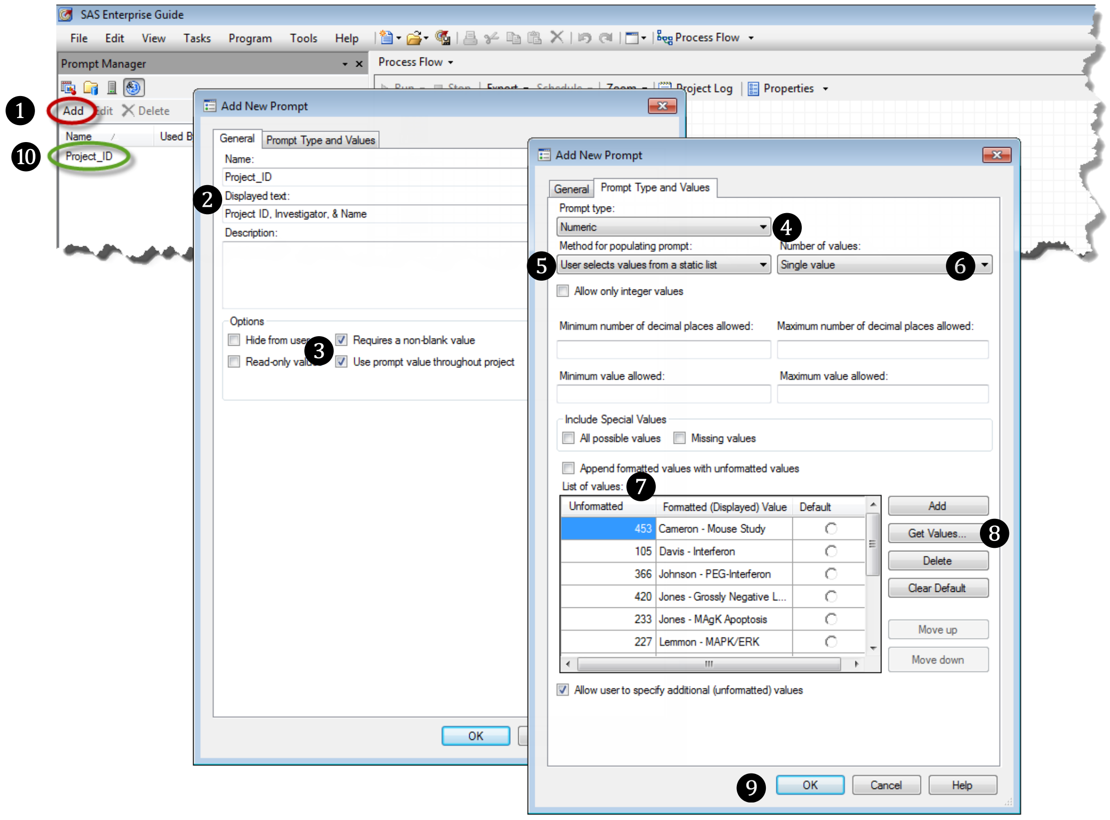
Column Name: pt_id, dx_stage, sex, dob, dx_date, age_at_dx, thickness, ulceration, vascular_invasion, first_surg, first_ned, recurrence, first_recurrence

☑ Select distinct rows

t1 (samples): sample_id, pt_id, mm, patient_number, sample_type, date_collected, freezer, box
t2 (participants): pt_id, mm, patient_number, dx_stage, sex, dob, dx_date, age_at_dx

After the join condition is created, the next step is to filter the data to reduce the dataset to only those participants associated with a sample distributed to a given project. However, the filter that will allow this project to dynamically produce clinical output for a given project quickly and efficiently involves the use of another Resource—the Enterprise Guide Prompt.

An Enterprise Guide Prompt is a GUI widget that allows users to assign values to parameters that drive a task's execution. To create a prompt, go to the "Prompt Resource" and click "Add" ❶. In the "General" tab of the "Add New Prompt" dialog, give your prompt a name and description ❷. In this example the "Project_ID" prompt is also specified to require a non-null value and once assigned the prompt will retain its value throughout the project ❸. In the "Prompt Type and Values" tab, the prompt is specified as "Numeric" ❹ because the "project_id" values in the samples dataset are numeric. When the prompt is presented to the user in the form of a drop-down list ❺, the user will be allowed to select a single value from the list ❻. The values that will populate that list can either be entered manually in the "List of Values" ❼ or you can look them up from a dataset by clicking the "Get Values" button ❽. After the prompt characteristics are specified, click "OK" ❾ to save your prompt to the project ❿.



Add New Prompt — General | Prompt Type and Values

Name: Project_ID

Displayed text: Project ID, Investigator, & Name

Description:

Options: Hide from user; Read-only value; [✓] Requires a non-blank value; [✓] Use prompt value throughout project

Add New Prompt — Prompt Type and Values

Prompt type: Numeric

Method for populating prompt: User selects values from a static list

Number of values: Single value

Allow only integer values

Minimum number of decimal places allowed: | Maximum number of decimal places allowed:

Minimum value allowed: | Maximum value allowed:

Include Special Values: All possible values; Missing values

Append formatted values with unformatted values

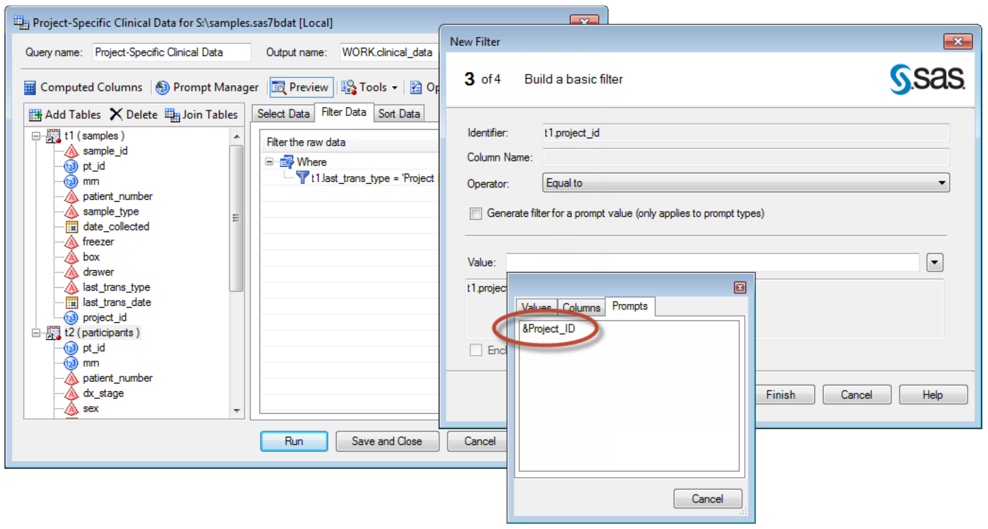
List of values:

Unformatted	Formatted (Displayed) Value	Default
453	Cameron - Mouse Study	○
105	Davis - Interferon	○
366	Johnson - PEG-Interferon	○
420	Jones - Grossly Negative L...	○
233	Jones - MAgK Apoptosis	○
227	Lemmon - MAPK/ERK	○

Add | Get Values... | Delete | Clear Default | Move up | Move down

[✓] Allow user to specify additional (unformatted) values

With the prompt created, the previous Query Builder task can now be completed by adding the "project_id" filter that will specify that the clinical dataset to be created will be limited to participants associated with samples distributed to the project having the "project_id" value specified in the prompt.



Query name: Project-Specific Clinical Data   Output name: WORK.clinical_data

Filter the raw data
Where
t1.last_trans_type = 'Project

New Filter — 3 of 4 Build a basic filter

Identifier: t1.project_id

Column Name:

Operator: Equal to

Generate filter for a prompt value (only applies to prompt types)

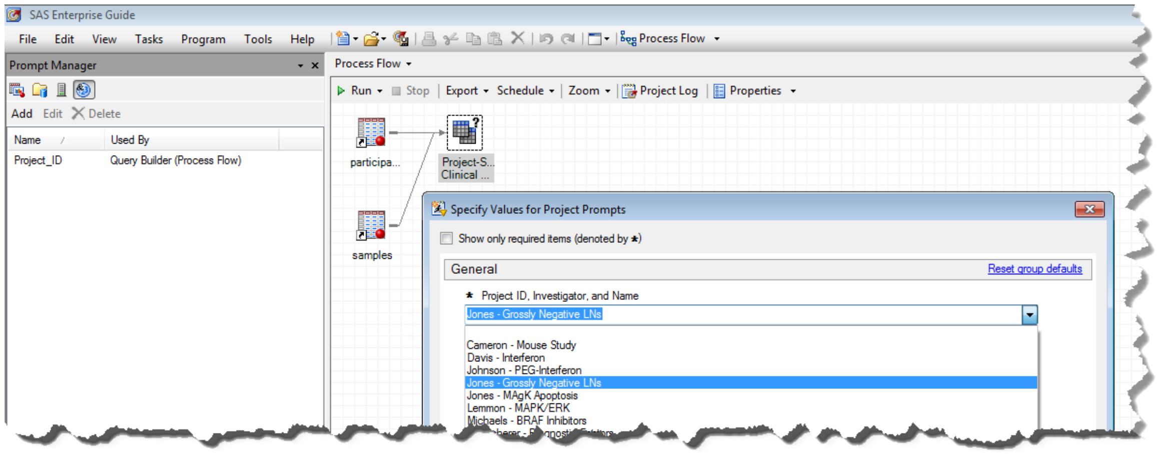
Value:

t1.projec

Enc

&Project_ID

Note that the criterion value of the "project_id" filter is assigned the "project_id" prompt as its criterion value. With this filter added to the query, now when someone runs this Enterprise Guide project, they will prompted for the value of the Project ID they want to use in filtering the result set.


SAS Enterprise Guide
File Edit View Tasks Program Tools Help
Process Flow
Prompt Manager
Add Edit Delete
Run Stop Export Schedule Zoom Project Log Properties
participa...
Project-S... Clinical ...
samples
Specify Values for Project Prompts
General
Reset group defaults

Name	Used By
Project_ID	Query Builder (Process Flow)

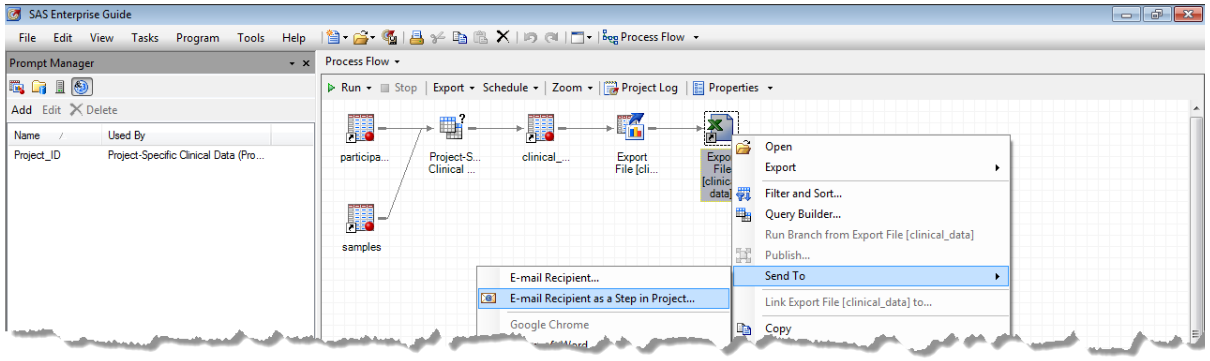
Show only required items (denoted by *)

* Project ID, Investigator, and Name

Jones - Grossly Negative LNs

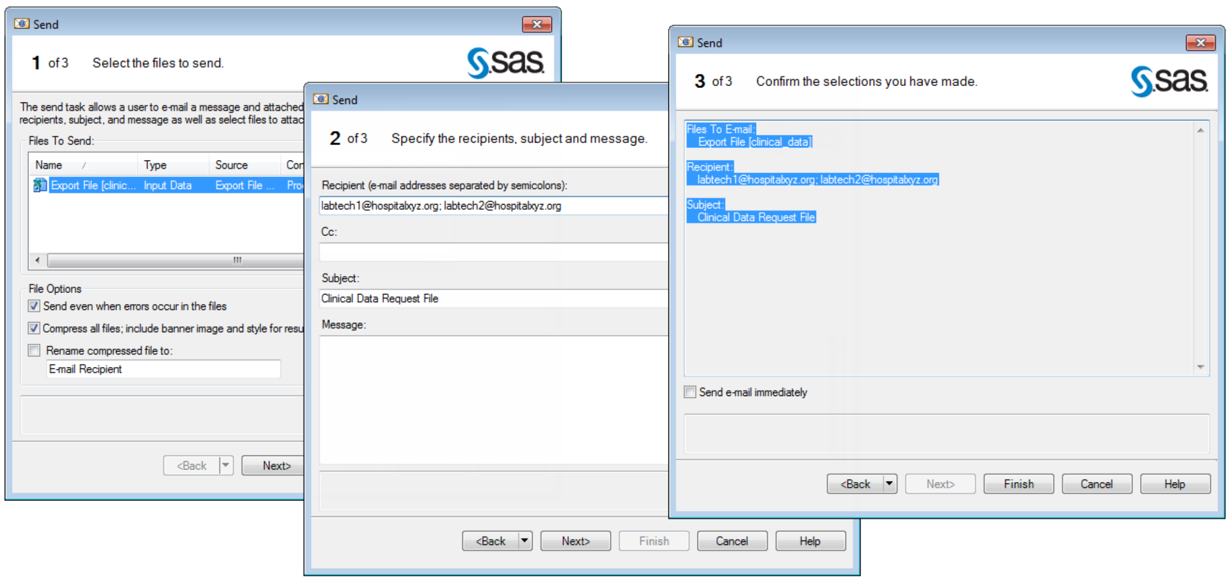
Cameron - Mouse Study
Davis - Interferon
Johnson - PEG-Interferon
Jones - Grossly Negative LNs
Jones - MAgK Apoptosis
Lemmon - MAPK/ERK
Michaels - BRAF Inhibitors

After generating the clinical dataset associated with the samples distributed to date for the "Grossly Negative LNs" study, the data are exported to Excel and then e-mailed to the lab-techs for review before forwarding them on to Dr. Jones.


SAS Enterprise Guide
File Edit View Tasks Program Tools Help
Process Flow
Prompt Manager
Add Edit Delete
Run Stop Export Schedule Zoom Project Log Properties
participa...
Project-S... Clinical ...
clinical_...
Export File [cli...
Expo File [clinic data]
samples
Open
Export
Filter and Sort...
Query Builder...
Run Branch from Export File [clinical_data]
Publish...
Send To
Link Export File [clinical_data] to...
Copy
E-mail Recipient...
E-mail Recipient as a Step in Project...
Google Chrome

Name	Used By
Project_ID	Project-Specific Clinical Data (Pro...

Like many other tasks, the E-Mail task walks the user through specification of the work they want performed, and then adds the configured task to the Process Flow. In the following task, the exported Excel file is selected for attachment to the e-mail, the recipients, subject line, and e-mail message are specified, and the E-Mail task is finished and will be represented in the Process Flow.



1 of 3 Select the files to send.

The send task allows a user to e-mail a message and attached recipients, subject, and message as well as select files to attac

Files To Send:

Name	Type	Source	Cor
Export File [clinic...	Input Data	Export File ...	Pro

File Options
[✓] Send even when errors occur in the files
[✓] Compress all files; include banner image and style for resu
[ ] Rename compressed file to:
E-mail Recipient

2 of 3 Specify the recipients, subject and message.

Recipient (e-mail addresses separated by semicolons):
labtech1@hospitalxyz.org; labtech2@hospitalxyz.org
Cc:
Subject:
Clinical Data Request File
Message:

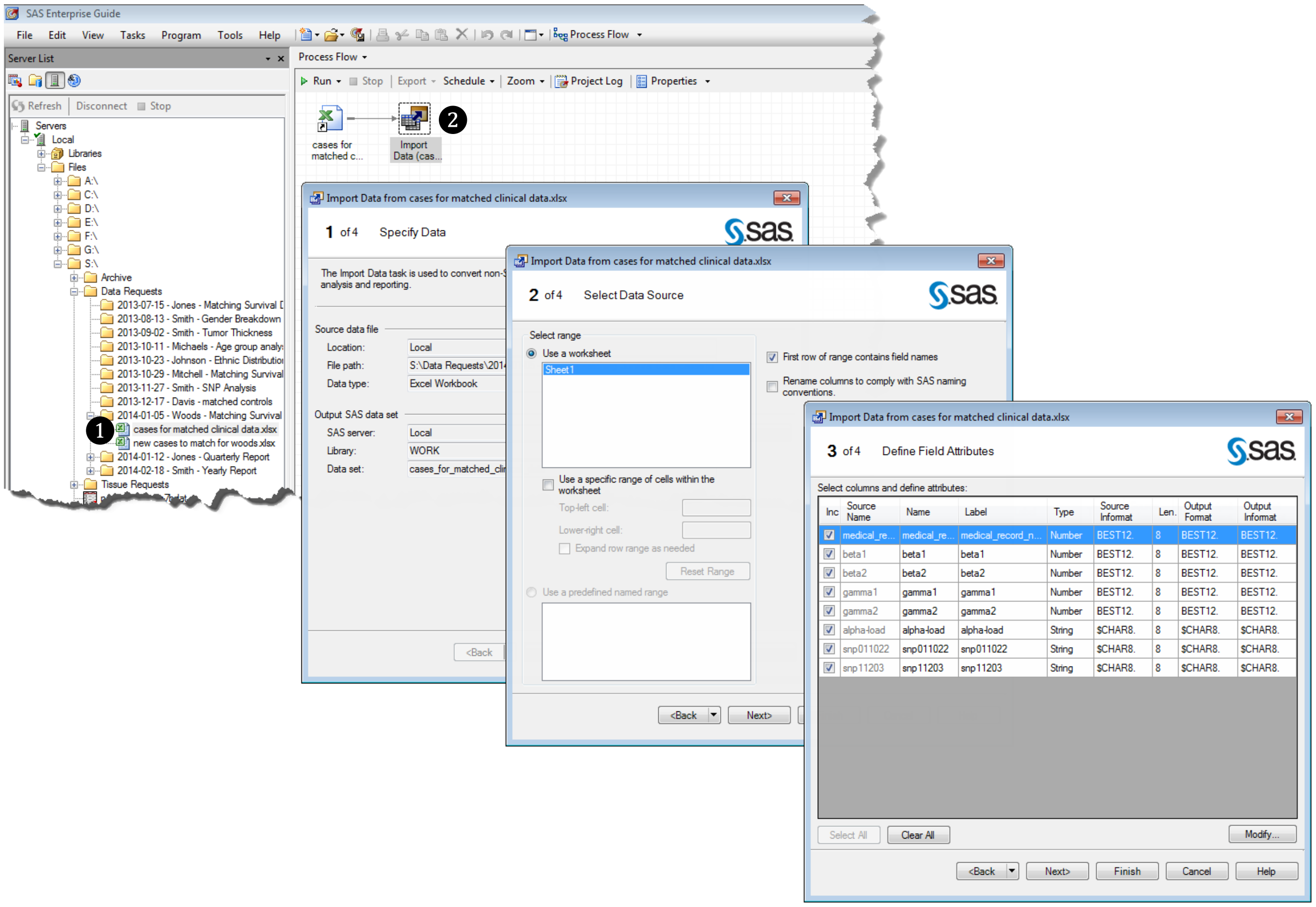
3 of 3 Confirm the selections you have made.

Files To E-mail:
Export File [clinical_data]
Recipient:
labtech1@hospitalxyz.org; labtech2@hospitalxyz.org
Subject:
Clinical Data Request File

[ ] Send e-mail immediately

PROVIDING MATCHING CLINICAL DATA

Another frequently performed task across the entire research team is receiving laboratory results from investigators and providing a canned clinical dataset matched to the data from the excel file. In this case, the data coordinators normally receive the dataset as an Excel file attached to an e-mail. They detach the file into the appropriate "Data Requests" folder on the "S:\" drive ❶ and then use an Import Data task to bring the data into their Enterprise Guide project ❷.



Servers
Local
Libraries
Files
A:\
C:\
D:\
E:\
F:\
G:\
S:\
Archive
Data Requests
2013-07-15 - Jones - Matching Survival D
2013-08-13 - Smith - Gender Breakdown
2013-09-02 - Smith - Tumor Thickness
2013-10-11 - Michaels - Age group analy
2013-10-23 - Johnson - Ethnic Distributio
2013-10-29 - Mitchell - Matching Survival
2013-11-27 - Smith - SNP Analysis
2013-12-17 - Davis - matched controls
2014-01-05 - Woods - Matching Survival
cases for matched clinical data.xlsx
new cases to match for woods.xlsx
2014-01-12 - Jones - Quarterly Report
2014-02-18 - Smith - Yearly Report
Tissue Requests

Import Data from cases for matched clinical data.xlsx

1 of 4 Specify Data

The Import Data task is used to convert non-S
analysis and reporting.

Source data file
Location: Local
File path: S:\Data Requests\2014
Data type: Excel Workbook

Output SAS data set
SAS server: Local
Library: WORK
Data set: cases_for_matched_cli

<Back

Import Data from cases for matched clinical data.xlsx

2 of 4 Select Data Source

Select range
(●) Use a worksheet
Sheet1
[ ] Use a specific range of cells within the worksheet
Top-left cell:
Lower-right cell:
[ ] Expand row range as needed
Reset Range
( ) Use a predefined named range

[✓] First row of range contains field names
[ ] Rename columns to comply with SAS naming conventions.

<Back Next>

Import Data from cases for matched clinical data.xlsx

3 of 4 Define Field Attributes

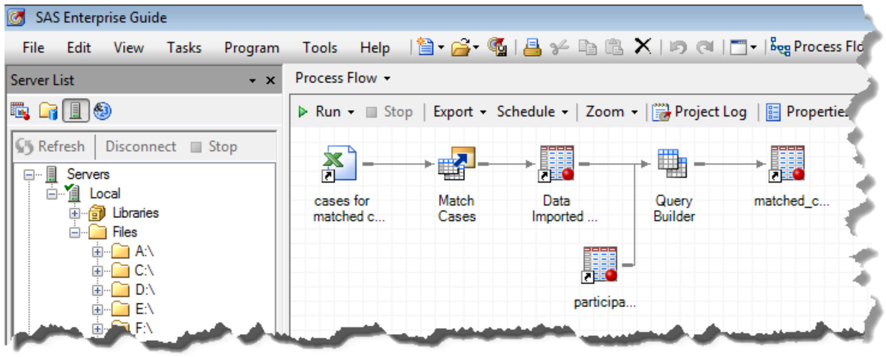
Select columns and define attributes:

Inc	Source Name	Name	Label	Type	Source Informat	Len.	Output Format	Output Informat
✓	medical_re...	medical_re...	medical_record_n...	Number	BEST12.	8	BEST12.	BEST12.
✓	beta1	beta1	beta1	Number	BEST12.	8	BEST12.	BEST12.
✓	beta2	beta2	beta2	Number	BEST12.	8	BEST12.	BEST12.
✓	gamma1	gamma1	gamma1	Number	BEST12.	8	BEST12.	BEST12.
✓	gamma2	gamma2	gamma2	Number	BEST12.	8	BEST12.	BEST12.
✓	alpha-load	alpha-load	alpha-load	String	$CHAR8.	8	$CHAR8.	$CHAR8.
✓	snp011022	snp011022	snp011022	String	$CHAR8.	8	$CHAR8.	$CHAR8.
✓	snp11203	snp11203	snp11203	String	$CHAR8.	8	$CHAR8.	$CHAR8.

Select All Clear All Modify...

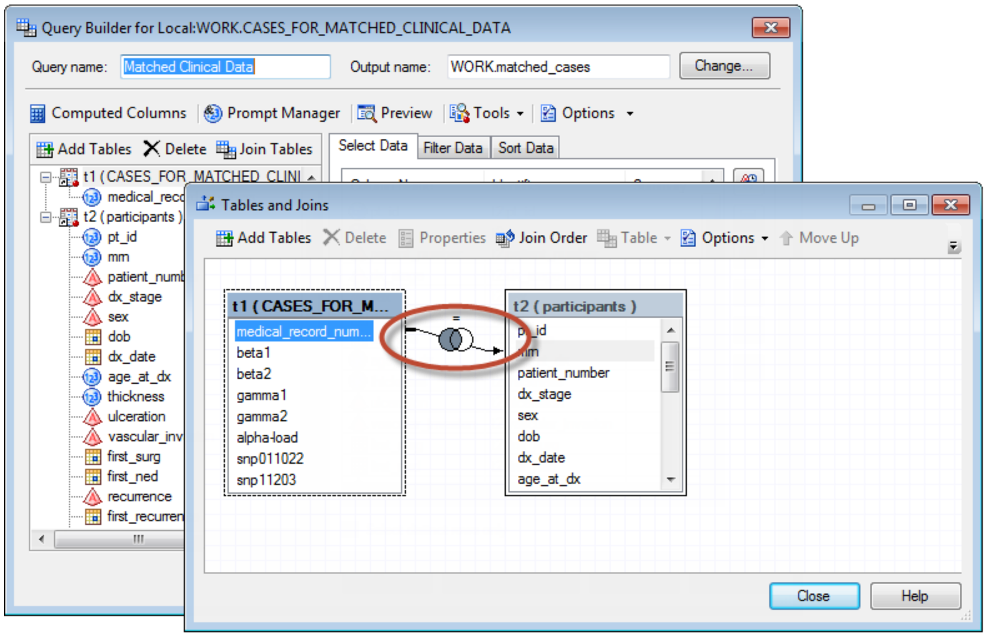
<Back Next> Finish Cancel Help

With these data imported into the Enterprise Guide project, you can easily match the lab data to the clinical data and create a matched clinical dataset using a Query Builder task.

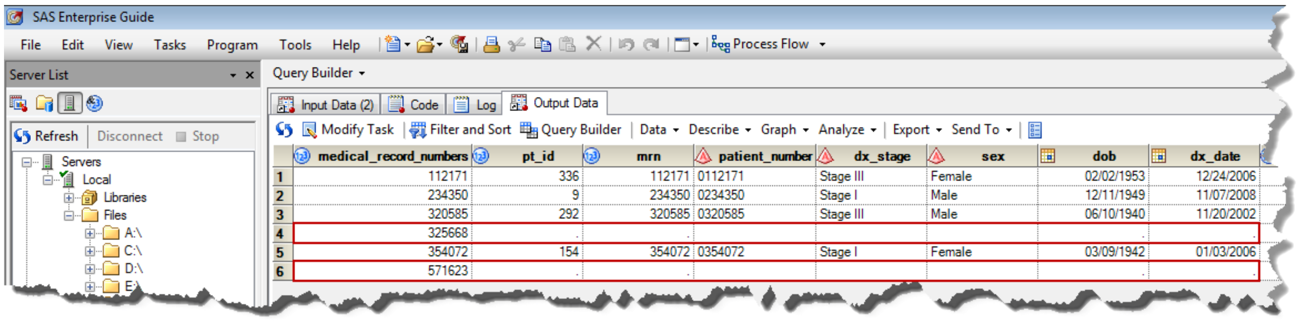


However, unlike the previous Query Builder example, the type of join used to match data from the two datasets will be different in this case. The previous queries all assumed that the resulting dataset should contain only those records from each dataset that had at least one matching record in the other dataset—an equi-join. In producing the matched clinical dataset in the current example, however, the lab investigator is expecting back one row of clinical data for each row of his or her lab data. If you send back only those records for which there is matching clinical data, there is a chance the investigator could lose track of some of their experimental data because they assume the version you sent back had all of their data AND all of the associated clinical data. For this reason, a "left join" is performed between the "participants" dataset and the imported experimental data.

To see how the left join is different, modify the Query Builder task and click on the "Join Tables" button. In looking at this join, you will notice a Venn diagram that symbolizes a "left join". All rows of data from the experimental lab dataset will be returned from the query and only those rows from the "participants" dataset that match a row in the lab dataset will be returned.



In this case, the resulting data shows that there are some individuals in the lab dataset for whom there are not yet clinical data available. This might further evolve into a report to help the clinical data coordinators prioritize their clinical data abstraction work so that data for this project are always complete.



	medical_record_numbers	pt_id	mrn	patient_number	dx_stage	sex	dob	dx_date
1	112171	336	112171	0112171	Stage III	Female	02/02/1953	12/24/2006
2	234350	9	234350	0234350	Stage I	Male	12/11/1949	11/07/2008
3	320585	292	320585	0320585	Stage III	Male	06/10/1940	11/20/2002
4	325668	.	.				.	.
5	354072	154	354072	0354072	Stage I	Female	03/09/1942	01/03/2006
6	571623	.	.				.	.

Provided with the simplified "samples" and "participants" datasets, lab techs, research assistants, data coordinators, study coordinators, and research nurses can all fulfill a number of operational data functions quickly and reliably. Of equal importance, however, is the fact that these research team members find the tool easy to use and are not intimidated in the least about trying to use new Enterprise Guide tasks or using familiar tasks in new ways. In fact, some of the team—realizing that SAS functions are really not very different from Excel functions—are a bit surprised they were ever intimidated by SAS at all. As their comfort level has grown, so has the productivity of the research team as a whole. The current work merely scratches the surface of the types of analytic tasks that the research team can now accomplish on their own; some additional uses of Enterprise Guide in this research team include the following:

Operational Task	SAS Concepts Utilized
Creating randomized lists of samples in the tissue bank inventory as the basis for in-house audits	Computing new variables SAS Functions [RANNOR]
Fulfilling service level agreements regarding the return of tissue samples	Computing new variables System Variables - SYSDATE
Reporting on tissue bank inventory by sample type and disposition	SAS Graph – bar charts/pie charts
Reporting demographics for sponsored research	N-Way Table Analysis (PROC FREQ)
Connecting to the CCL Oracle database and extracting data from other operational tables in order to supplement the "canned" datasets with detailed information regarding tissue transactions, patient consents, and tissue request fulfillment from outside hospitals.	LIBNAMES "Autoexec" Process Flows Advanced Query Builder tasks

The ease-of-use of Enterprise Guide has proven itself in this situation as a true "multiplier" of the research team; using Enterprise Guide, the team is able to facilitate scientists' access to critical data that moves research forward at a faster pace than would be otherwise possible.

REFERENCES

Aanderud, T. & Hall, A. (2012). Building Business Intelligence Using SAS®: Content Development Examples. Cary, NC: SAS Institute, Inc.

Carpenter, A. (1997). Resolving and Using &&var&i Macro Variables. Proceedings of the 22nd Annual SAS Users Group International Meeting. Cary, NC: SAS Institute, Inc.

DeGuire, Y. (2007). The FILENAME Statement Revisited. Proceedings of the SAS Global Forum 2007. Cary, NC: SAS Institute, Inc.

Fecht, M. & Dhillon, R. (2011). SAS Enterprise Guide 4.3: Finally a Programmer's Tool. Proceedings of SAS Global Forum 2011. Cary, NC: SAS Institute, Inc.

Hallahan, C. & Atkinson, L. (2006). Introduction to SAS® Enterprise Guide® 4.1 for Statistical Analysis. Proceedings of the 31st Annual SAS Users Group International Meeting. Cary, NC: SAS Institute, Inc.

Lafler, K.P. (2003). Undocumented and Hard-to-find SQL Features. Proceedings of the 28th Annual SAS Users Group International Meeting. Cary, NC: SAS Institute, Inc.

Lafler, K.P. (2004). PROC SQL: Beyond the Basics Using SAS. Cary, NC: SAS Institute, Inc.

Lafler, K.P. (2005). Manipulating Data with PROC SQL. Proceedings of the 30th Annual SAS Users Group International Meeting. Cary, NC: SAS Institute, Inc.

SAS Institute Inc. (1990). SAS Guide to Macro Processing. Version 6, Second Edition, Cary, NC: SAS Institute, Inc.

SAS Institute Inc. (2011). Base SAS® 9.3 Procedures Guide. Cary, NC: SAS Institute, Inc.

Schacherer, C. (2012a). The FILENAME Statement: Interacting with the World Outside of SAS®. Proceedings of the SAS Global Forum 2012. Cary, NC: SAS Institute, Inc.

Schacherer, C. (2012b). SAS® Data Management Techniques: Cleaning and transforming data for delivery of analytic datasets. Proceedings of the SAS Global Forum 2012. Cary, NC: SAS Institute, Inc.

Schacherer, C. (2013). The Concepts and Practice of Analysis with SAS® Enterprise Guide®. Proceedings of the SAS Global Forum 2013. Cary, NC: SAS Institute, Inc.

Schacherer, C.W. & Detry, M.A. (2010). PROC SQL: from SELECT to Pass-Through SQL. Proceedings of the South Central SAS Users Group. Cary, NC: SAS Institute, Inc.

Schacherer, C.W. & Westra, B.D. (2010). Introduction to SAS® for the Healthcare Analyst. Proceedings of the Midwest SAS Users Group. Cary, NC: SAS Institute, Inc.

Slaughter, S.J. & Delwiche, L.D. (2010). The Little SAS Book for Enterprise Guide 4.2. Cary, NC: SAS Institute, Inc.

Tilanus, E.W. (2008). Sending E-mail from the DATA step. Proceedings of the SAS Global Forum 2008. Cary, NC: SAS Institute, Inc.

CONTACT INFORMATION

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

Christopher W. Schacherer, Ph.D.
Clinical Data Management Systems, LLC
Madison, WI 53711
E-mail: CSchacherer@cdms-llc.com
Web: www.cdms-llc.com