

# The Concepts and Practice of Analysis with SAS® Enterprise Guide®

Chris Schacherer, Clinical Data Management Systems, LLC

## ABSTRACT

SAS Enterprise Guide continues to gain acceptance among SAS programmers as an enterprise solution for a wide range of reporting and analytic applications. However, in order for organizations to realize maximum benefit from their investment in Enterprise Guide, subject-matter experts and a new generation of analysts who have not been trained as SAS programmers need to be trained in the use of this tool. The current work provides a framework for this training by explaining the relationship between Enterprise Guide and traditional SAS programming, introducing the basic Enterprise Guide concepts necessary to function in this environment, and presenting examples of common tasks that will help these users become immediately productive.

## INTRODUCTION

SAS has been a leader in the field of analytics for more than 30 years. For much of that time, however, access to the power of SAS analytics was limited to those intrepid few who were driven to master the SAS programming language. Still today, mastering DATA and PROCEDURE step syntax, acquiring a tool bag of one's most trusted SAS Functions, and learning how to "think like SAS" still involves a significantly long learning curve—at a time when data and analytics are of greater strategic (and operational) importance than ever before. However, thanks to Enterprise Guide, new analysts without SAS programming experience and business users with the subject matter expertise necessary to directly utilize analytics in their day-to-day decision-making can readily gain access to the power of SAS analytics.

In the past, many potential users of SAS software within the business-user ranks were overwhelmed by the prospect of having to "program" in order to extract, manipulate, and analyze data. Many of these same users were very comfortable using GUI tools to sort and filter datasets, but the benefit of having access to the analytic power of SAS did not outweigh their trepidation over the perceived time and effort required to learn both the syntax of the SAS programming language and the concepts necessary to operate efficiently in the SAS system. Similarly, individuals new to the IT analyst role were often intimidated by the expectation that they would quickly become proficient with the SAS language and immediately be an efficient, productive member of the analytic function within their organization. Managers were often tasked with training these new analysts in SAS programming in addition to developing their content knowledge about the data and processes they were analyzing. There were significant risks on all sides of this equation..."will I be successful in this new analyst position?"..."will the new analyst develop all of the skills I need them to master?" Meanwhile, business users were still queuing up with lists of ad hoc requests that needed attention because they had no way of autonomously accessing, manipulating, and reporting on the data.

To address this confluence of challenges, Enterprise Guide was developed to make SAS accessible to a wider variety of users. Until recently, however, SAS programmers have been slow to accept Enterprise Guide as a bona fide SAS development solution. Many felt it was really just a point-and-click end-user tool, not a full-fledged programming tool like SAS Display Manager. More recent versions of Enterprise Guide, however, have made it easier for SAS programmers to leverage their SAS skills against the task of creating powerful, robust, and user-friendly analytic and reporting solutions, and as a result, there is growing acceptance of Enterprise Guide as an analytic solution (Bang, Hemedinger, & Slocum, 2010; Fecht & Dhillon, 2011; Ravenna, 2011; Schacherer, 2012a; Sucher, 2010).

This rejuvenated interest in Enterprise Guide as an enterprise solution is likely to result in a significant (and potentially rapid) culture shift in the SAS community. When successfully implemented, Enterprise Guide holds the potential to greatly facilitate the development of new analysts and to enable a broader group of business users to answer questions themselves rather than waiting in a queue for the next available SAS programmer. For both of these populations, SAS Enterprise Guide can be a tremendously powerful tool, and the organizations that best facilitate the empowerment of these employees will gain a competitive advantage.

The current work, therefore, focuses on the addressing the training needs of new Enterprise Guide users with little or no previous SAS programming experience. First, an overview of traditional SAS programming methods and concepts is presented in order for users to better understand what Enterprise Guide really is and how it does what it does<sup>1</sup>. Following this overview of traditional SAS programming, a description of the Enterprise Guide interface is provided and the relationship between Enterprise Guide tasks and projects and the SAS programming language is explained.

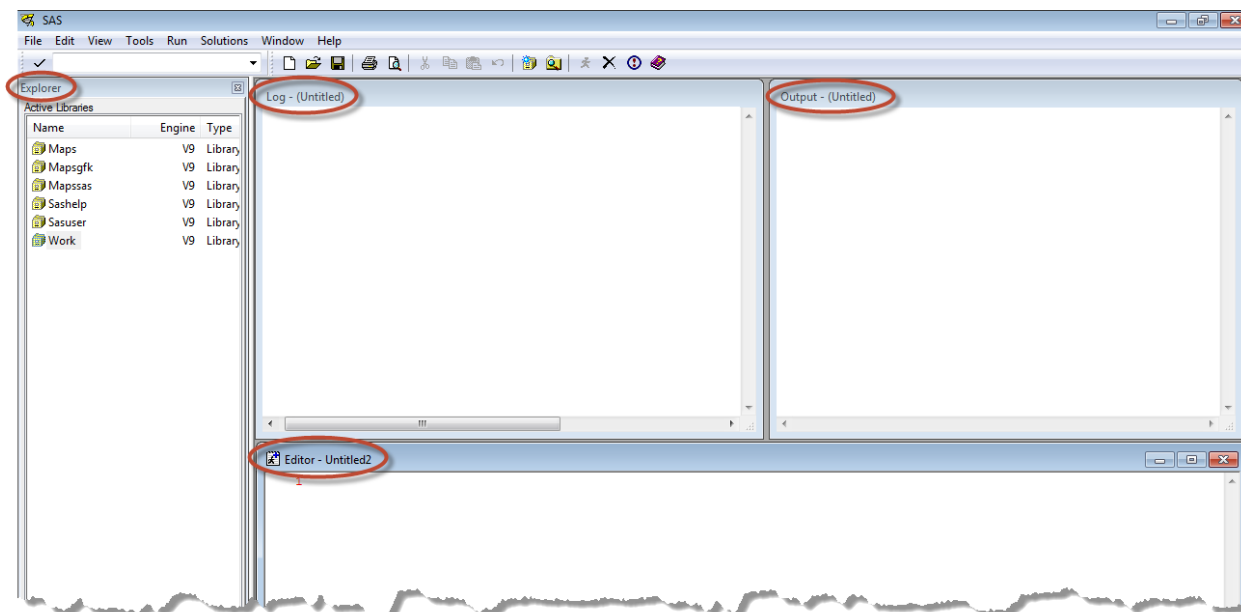
---

<sup>1</sup> Having at least a high-level understanding of the SAS system also helps bridge communication gaps between these new Enterprise Guide users and experienced SAS programmers.

This description includes not only explanations of the interface components over which the user has control, but also the data sources, executable objects, and storage locations that are controlled by the SAS administrator and business intelligence content builders. Finally, a sample project is built from start to finish—demonstrating the use of Enterprise Guide tasks and introducing SAS concepts that users new to the SAS environment will find useful in developing their own SAS Enterprise Guide solutions.

## SAS Programming

Traditionally, SAS programmers manipulate data and generate output by writing and executing programs using SAS language syntax in a development environment known as the Display Manager. This development environment is made up of four main components: the EDITOR, LOG, and OUTPUT windows and the SAS EXPLORER.



The EDITOR window is where the actual syntax of the SAS program is written. The commands written here can be used to import data files, extract data from relational database management systems, filter, sort, and merge datasets, transform/recode variables, and execute analytic procedures. As these programs are run, SAS generates information about the execution of the SAS syntax in the LOG window—identifying syntax errors in the code and describing the datasets being created. As analytic and reporting procedures are executed, the default behavior of SAS is to print the results of those procedures in the OUTPUT window (or, the Results Viewer beginning in SAS 9.2). The SAS EXPLORER allows the programmer to keep track of the data objects with which he or she is working, and, more importantly, allows a means of manually interacting with those datasets by opening them in a data grid view or right-clicking on them to study their properties.

Of course, one of the long-standing stumbling blocks to new users of the SAS system is that SAS is, by its nature, a programming language. This means that there are new concepts to learn and new syntax to master in order to perform even the most basic data transformations and analyses. The SAS program depicted below (see Schacherer, 2012b for the full code) is used to import a monthly data file named with the convention "<year>\_<month>.txt" from an FTP server, create a summary report, and e-mail a .PDF file of the results to the analyst's. When executed, the SAS processing engine begins parsing the program at the first line of the file and continues through the end of the file—executing each processing step as it is encountered in the program file. Performing these same steps manually would require using a number of different software applications to navigate to the FTP server, define how to import the file, and download it, compute new variables once the file is downloaded, create the report output, save the output as a .PDF file, and attach the file to an e-mail message. With this SAS program, however, once the code for these steps is written, all you need to do to perform these steps again is to open the program and execute it again in Display Manager. Especially for recurring reports and analyses, such a programmatic approach has clear advantages over performing the task manually. Moreover, the sophisticated statistical analysis and data manipulation capabilities available in SAS provide functionality available in few other software packages.

```

Paid Claims Report - 12JAN2011A.sas
1 %LET SOURCE_FILE1 = %SYSFUNC(YEAR(%SYSFUNC(INTNX(MONTH,%SYSFUNC(TODAY()),-1))))_;
2 %LET SOURCE_FILE2 = %SYSFUNC(MONTH(%SYSFUNC(INTNX(MONTH,%SYSFUNC(TODAY()),-1))),22.) .txt;
3
4 FILENAME source FTP "&source_file1&source_file2" USER='chris' PASS=&PASSWORD
5 HOST = 'schacherer.com' CD = "ftproot\public\" DEBUG;
6
7 DATA monthly_report_data;
8   INFILE source FIRSTOBS=2 MISOVER LRECL=300;
9   INPUT @1 datepaid mmddyy10. @13 account_no $10. @25 prin_dx $6. @273 paid_amount dollar12.;
10  IF billed_charges ne . THEN discount_rate = .8*billed_charges;
11  ELSE discount_rate = 0.;
12  PUT account_no svc_date :mmddyy10. cpt billed_charges :dollar12.2;
13  RUN;
14
15 ODS PDF FILE='C:\monthly claims report.pdf' STYLE = Journal
16 TITLE 'Claim Payments by Paid Date';
17
18 PROC TABULATE DATA=monthly_report_data;
19 CLASS datepaid;
20 VAR paid_amount;
21 TABLE datepaid='',sum=''*(paid_amount='Total Paid')*FORMAT=DOLLAR20.
22 /BOX='Paid Date';
23 RUN;

```

So, how does one write a SAS program? The answer provided below is not meant to train the reader in all of the necessary concepts and techniques, but to introduce you to these concepts so that you can identify and understand their parallels in the Enterprise Guide environment.

**Program Headers & Comments.** The first step in writing a good SAS program is to write a "header" to help future users of the program understand why the program was written as well as to identify who wrote it and when it was written. Header information is written at the top of the program as a comment (or series of comments)—using "/" to signify the beginning of the comment and "\*" to signify the end of the comment. Using the commenting syntax is important because when you run your program the SAS processing engine will attempt to execute all uncommented text. If your comments are not enclosed in appropriate commenting marks, the SAS engine will attempt to execute them as if they were SAS programming commands—which will result in errors being generated in the LOG.

```

Example SAS Program.sas
1 /*
2 2013-02-13A - Author: Chris Schacherer
3           Initial version of SAS Program to integrate Wellness Program participation
4           data with medical claims and study impact of program participation on total
5           claims cost for XYZ Co.
6 */

```

Whether using a more or less formal method of documenting the purpose of your programs, the key to writing effective header information is recording sufficient information to enable future users of the program to understand the original purpose of the program and any major revisions it has undergone. Combined with comments in the body of the program that explain the program logic, programmers that inherit the code will be able to successfully maintain, debug, and modify the program throughout its lifecycle. After completing or updating the header section of a program, a common first executable statement found in a SAS program is a LIBNAME statement—which defines the location(s) (or, "libraries") where SAS will find the source data being manipulated/analyzed and where the SAS datasets created by the program will be stored.

**LIBRARIES.** Libraries in SAS are logical pointers used to specify the location of the SAS objects with which the program interacts. These locations can be directories on a local hard drives, mapped network drives, or database schemas in a relational database management system such as Oracle®, Microsoft® SQL Server®, etc. Within a SAS program, libraries are defined using the LIBNAME statement. The following LIBNAME statement defines the "Production Datasets" directory on the mapped "S:" drive as the library "LOCAL".

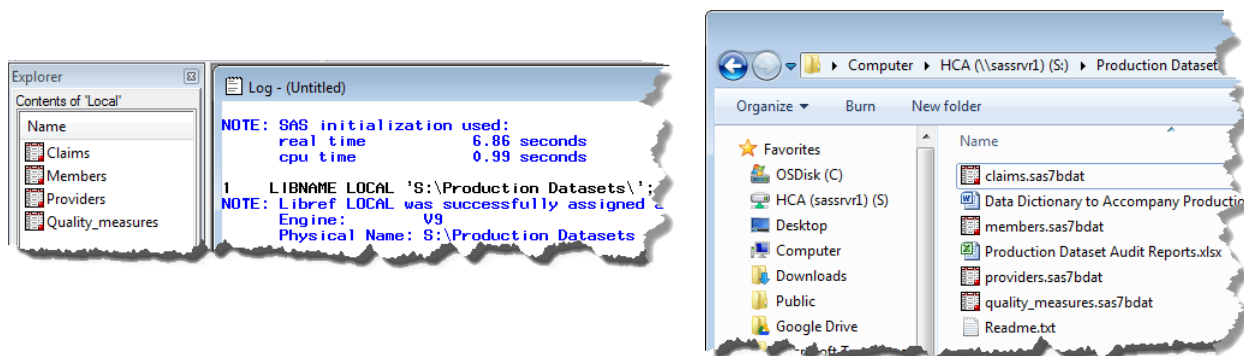
```

Example SAS Program.sas
1 /*
2 2013-02-13A - Author: Chris Schacherer
3           Initial version of SAS Program to integrate Wellness Program participation
4           data with medical claims and study impact of program participation on total
5           claims cost for XYZ Co.
6 */
7
8 LIBNAME LOCAL 'S:\Production Datasets\';
9

```

After the LIBNAME statement is executed, SAS will recognize the library name (or "libref") "LOCAL" as a reference to the location "S:\Production Datasets". This relationship between the libref and the specified location will remain intact for the rest of the SAS session or until the libref "LOCAL" is reassigned to a different location using another LIBNAME statement. Whenever a reference to the library LOCAL is made in your SAS program, "S:\Production Datasets" will be searched for the specified object. In Display Manager, when you open "LOCAL" in the SAS Explorer, you will see the SAS data objects that are stored in this directory.

One characteristic of libraries that is sometimes confusing to novice SAS programmers is that only SAS data objects appear in the library when it is viewed through the SAS Explorer. Even though the specified directory may contain other files that can be viewed in Windows Explorer (e.g., Readme.txt, etc.), only objects that SAS recognizes as natively accessible by SAS are seen in the SAS Explorer view of the library.



For example, in the Windows Explorer® window on the right, we see that, in addition to the SAS datasets there are also Microsoft Word® and Excel® files as well as a .TXT file. In the SAS Explorer window showing the contents of "Local" the only files that are displayed are the SAS datasets.

Although the "S:" drive was successfully used to specify the location of the library "LOCAL" in the previous example, it should be noted that if this program is going to be shared by different analysts, everyone that uses the program would need to have their "S:" drive mapped to the same location as the creator of the SAS program. Therefore, for the sake of portability of your code from one computer to another, it is recommended that libraries referencing network locations be defined using their universal naming convention (UNC) name. In the current example, you would specify LOCAL as follows to specify the full server and directory path to the "Production Datasets."

```

Example SAS Program.sas
1  /*
2  2013-02-13A - Author: Chris Schacherer
3                Initial version of SAS Program to integrate
4                data with medical claims and study impact
5                claims cost for XYZ Co.
6  */
7
8  LIBNAME LOCAL '\\sassrvr1\HCA\Production Datasets\';
9

```

Finally, source data are not just shared as SAS data files on local or network drives, but often resides in relational database management systems. In order to facilitate access to these systems, SAS allows users (via SAS/ACCESS) to connect directly to the databases to which they have access and read the associated database tables and views much as they would read SAS datasets on local and network drives.

In the following example, the SAS Library "FINANCE" is defined as a connection to the Microsoft SQL Server database "billing". This library definition specifies that the database type is an Object Linking and Embedding (OLE) connection to the local system interface SQLOLEDB.1. The library specification further defines the "initial catalog" as the database "billing" running on the SQL Server named "SQLSRVR1" and the database schema to which the SAS library "FINANCE" will attach is "DBO".

```

LIBNAME FINANCE OLEDB PROVIDER=SQLOLEDB.1 REQUIRED=Yes
              USER = cschacherer DATASOURCE = "SQLSRVR1"
              PROPERTIES = ('initial catalog'=billing
                            'Persist Security Info'=True)
              SCHEMA = 'DBO';

```

Finally, in addition to libraries defined by LIBNAME statements in the SAS program, there is one library that is always available when a SAS session is initiated—WORK. The WORK library refers to temporary work space used by SAS programs. If your program creates interim datasets that you do not intend to persist after the execution of the program, you can specify their storage location as the WORK library and they will be purged when your SAS session ends. A common mistake made by novice SAS programmers is creating a final, analytic dataset in the WORK library with the idea that they will use that dataset in a future analysis. Much to their surprise, however, the next time they start SAS, the WORK directory is empty—because, of course, it was purged when they ended their previous SAS session.

**THE PROGRAM BODY – DATA AND PROC STEPS.** The body of a SAS program is where the code is written to perform data transformations, analyses, and report generation. This work is accomplished using two types of executable syntax: DATA steps and PROC steps. DATA steps process the rows of data in a source dataset in a serial, sequential fashion—beginning with the first record in the dataset and continuing (by default) until the end of the source dataset is reached. The code written within the DATA step is used to create new variables, assign values to variables, apply conditional logic to the processing of records in the dataset, and identify which records and variables are output to the resulting dataset. You can think of the DATA step as a general purpose, user-defined manipulation of the source dataset—a DATA (management) step, if you will.

In its simplest form, the DATA step involves specifying the target dataset you wish to create and using the SET command to specify the source dataset(s) that you wish to use in creating the target dataset. The DATA step is terminated by the "RUN" command which indicates to SAS the end of an executable block of code. The following DATA step creates a copy (in the WORK library) of the "claims" dataset read from the "local" library. Each record in "local.claims" (the "fully qualified" name of the dataset) is read into memory and then written out (in this case with no additional transformations) to the dataset "work.claims01".

```
DATA work.claims01;
  SET local.claims;
RUN;
```

What happens between the DATA statement and RUN determines how the source dataset is transformed to create the target dataset. For example, if you wanted to subset a dataset of healthcare claims to create a dataset consisting of only "PAID" claims for a specific client company (for example, "XYZ Co.") you could use a "WHERE" clause to specify that only records with a "client\_code" value of 25 (the code used to identify client company "XYZ Co.") and a "claim\_status" of "PAID" should be included in "xyz\_claims". Within this same DATA step, you could compute the new variable "netpay" for each record in "xyz\_claims" by declaring the variable name and assigning it a value equivalent to the resolution of the arithmetic expression "allowed\_amount – copay". The result of this DATA step is a dataset of paid claims generated by employees of the client company "XYZ Co."

```
DATA work.xyz_claims;
  SET local.claims;
  WHERE client_code = 25 and claim_status = 'PAID';
  netpay = allowed_amount - copay;
RUN;
```

The resulting dataset, in turn, could be merged with other datasets in subsequent steps in the program to yield a dataset with all of the variables necessary for your report or analysis<sup>2</sup>.

Whereas the DATA step is a very powerful tool for manipulating the records and variables that make up a dataset, much of the power of SAS is derived from PROCEDURE (or PROC) steps. According to the SAS (2010), "SAS procedures analyze data in SAS data sets to produce statistics, tables, reports, charts, and plots, to create SQL queries, and to perform other analyses and operations on your data. They also provide ways to manage and print SAS files". Put simply, PROCs are specialized program units provided by SAS to perform specific types of operations on datasets. Through utilization of syntax extensions and options available for each PROC, the SAS programmer can modify the PROC's behavior, but its core functionality is confined to the type of operation for which it was designed.

One SAS procedure commonly used to "profile" (or, provide an overview of) a dataset is the FREQUENCY PROCEDURE (PROC FREQ) which produces frequency distributions and performs inferential statistical tests on categorical data. As demonstrated in the following example, like many other PROCs, PROC FREQ begins with specification of the dataset on which the PROC will be performed—in this case, the "xyz\_claims" dataset. This is

<sup>2</sup> For excellent, expanded explanations of the DATA step programming, the reader is referred to Aster & Seidman (1997), Cody (2007), and Whitlock (2007).

followed by a TABLE statement that specifies the dataset variables (or combination of variables) that will be analyzed. In this example, the distribution of health plan claims for "XYZ Co." is analyzed for each of two categorical variables—"gender" and "wellness" (an indicator of whether the health plan member with which the claim is associated is a participant in the company's Wellness program).

```
TITLE 'Claim Frequencies Across Gender and Wellness Program Participation';
PROC FREQ DATA=work.xyz_claims;
  TABLE wellness gender;
RUN;
```

The screenshot shows a window titled "Results Viewer - sashtml.htm" with the following content:

**Claim Frequencies Across Gender and Wellness Program Participation**

The FREQ Procedure

wellness	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Non-Participant	2569	39.66	2569	39.66
Participant	3909	60.34	6478	100.00

gender	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	3771	58.21	3771	58.21
M	2707	41.79	6478	100.00

In the preceding PROC FREQ example, individual tables are produced to look at the distribution of healthcare claims generated by "XYZ Co." employees across "gender" and "wellness" program participation but with a simple adjustment to the TABLE statement's syntax, you can produce the cross-tab of the claim frequencies across both gender and wellness.

```
TITLE 'Crosstab of Frequencies (Gender x Wellness Program Participation)';
PROC FREQ DATA=work.xyz_claims;
  TABLE wellness*gender;
RUN;
```

The screenshot shows a window titled "Results Viewer - SAS Output" with the following content:

**Crosstab of Frequencies (Gender x Wellness Program Participation)**

The FREQ Procedure

Frequency Percent Row Pct Col Pct	Table of wellness by gender			
	wellness	gender		Total
		F	M	
Non-Participant	1487	1082	2569	
	22.95	16.70	39.66	
	57.88	42.12		
	39.43	39.97		
Participant	2284	1625	3909	
	35.26	25.08	60.34	
	58.43	41.57		
	60.57	60.03		
Total	3771	2707	6478	
	58.21	41.79	100.00	

In addition to this tabular output, PROC FREQ can also be used to produce inferential statistics (e.g., Chi-Square tests) and to produce a dataset containing the PROC FREQ output that can be used as source data for subsequent steps in the SAS program.

```
PROC FREQ DATA=work.xyz_claims;
  TABLE wellness*gender / chisq out = wellness_gender_freq;
RUN;
```

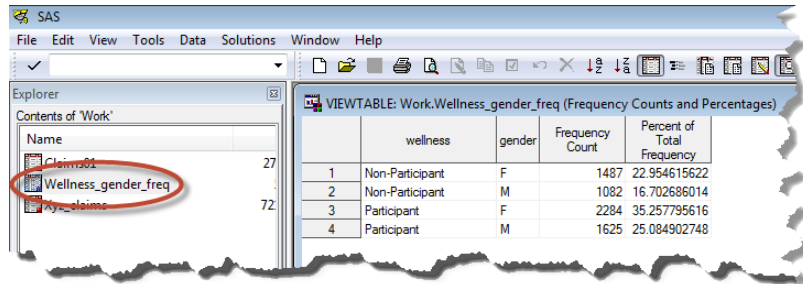
Statistics for Table of wellness by gender

Statistic	DF	Value	Prob
Chi-Square	1	0.1906	0.6625
Likelihood Ratio Chi-Square	1	0.1905	0.6625
Continuity Adj. Chi-Square	1	0.1687	0.6812
Mantel-Haenszel Chi-Square	1	0.1905	0.6625
Phi Coefficient		-0.0054	
Contingency Coefficient		0.0054	
Cramer's V		-0.0054	

Fisher's Exact Test	
Cell (1,1) Frequency (F)	1487
Left-sided Pr <= F	0.3405
Right-sided Pr >= F	0.6781
Table Probability (P)	0.0187
Two-sided Pr <= P	0.6804

Sample Size = 6478



Just as PROC FREQ can analyze an entire dataset and produce a frequency distribution in seconds, the MEANS procedure (PROC MEANS) can be used to provide the mean, standard deviation, and minimum/maximum values for one or more numeric variables in your analytic dataset<sup>3</sup>. In the following example, these statistics are generated for the "netpay" variable produced an earlier example.

```
TITLE 'Mean Value of Netpay Across all XYZ Co. Claims';
PROC MEANS DATA=work.xyz_claims;
  VAR netpay;
RUN;
```

Again, by altering the syntax of the PROC slightly, a different analytic question is answered. In the following example, the mean, standard deviation, and minimum/maximum values for "netpay" are produced separately for the claims associated with health plan members who are "Participants" and "Non-Participants" in the Wellness program.

```
TITLE 'Average Netpay for Wellness "Participants" and "Non_Participants"';
PROC MEANS DATA=work.xyz_claims;
  VAR netpay;
  BY wellness;
RUN;
```

<sup>3</sup> In addition to being useful for standard reporting of descriptive statistics, however, PROC MEANS is also valuable as a data cleaning tool—as it can reveal out-of-range values (e.g., identifying negative values for "Patient Age", revealing variables like "Cost", and "Billed Amount" that may have too little or too much variance based on historical data, etc.).



Results Viewer - SAS Output

**Average Netpay for Wellness "Participants" and "Non\_Participants"**

The MEANS Procedure

wellness=Non-Participant

Analysis Variable : netpay				
N	Mean	Std Dev	Minimum	Maximum
2569	459.4987699	415.0800709	51.0500000	2296.00

wellness=Participant

Analysis Variable : netpay				
N	Mean	Std Dev	Minimum	Maximum
3909	97.6290151	59.4494984	0	274.9700000

In order to analyze a dataset "BY" one or more variables, however, the dataset first needs to be sorted by those variables. This is accomplished using the SORT PROCEDURE (PROC SORT). In the following example, the "xyz\_claims" dataset is sorted by "wellness" so that the PROC MEANS analysis can be executed by each group. Unlike the preceding PROC steps however, PROC SORT does not necessarily produce any new output by default but rewrites the existing dataset in the sort order specified in the BY statement.

```
PROC SORT DATA=work.xyz_claims;
  BY wellness;
RUN;
```

Like all code executed in the SAS program, log entries are generated by execution of the PROC SORT step. The log entries generated by SAS as it executes a program provide valuable feedback to SAS programmers—allowing them to know whether or not there were errors encountered in the execution of their program. In this case, the log entry indicates that the sort was successfully executed—sorting the 6478 records in the "xyz\_claims" dataset in .07 seconds.

```
Log - (Untitled)
954
955 proc sort data=work.xyz_claims;
956 by wellness;
957 run;

NOTE: There were 6478 observations read from the data set WORK.XYZ_CLAIMS.
NOTE: The data set WORK.XYZ_CLAIMS has 6478 observations and 13 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.07 seconds
      cpu time            0.00 seconds
```

SAS programs are generally written as a series of DATA and PROC steps used to manipulate data and accomplish a discrete piece of work such as the production of a monthly report, the statistical analyses associated with a scientific manuscript, or root cause analysis of an aberrant performance indicator. Using a combination of DATA and PROC steps, the analyst is able to start from the raw source data, manipulate those data to create the appropriate analytic datasets, and report on or analyze those datasets to achieve the objective of the program. In the adjacent program, the DATA and PROC steps in the previous examples are shown as part of the SAS program "XYZ Claims & Wellness Analysis.sas". When run in the Display Manager, all of the output shown in the previous examples is generated to the Results Viewer and descriptions of the execution of each step is written to the LOG window.

```
XYZ Claims & Wellness Analysis.sas
1 /*
2 2013-02-13A - Author: Chris Schacherer
3 Initial version of SAS Program to integrate Wellness Program participation
4 data with medical claims and study impact of program participation on total
5 claims cost for XYZ Co.
6 */
7
8 LIBNAME LOCAL '\\sasser\HCA\Production Datasets\';
9
10 LIBNAME FINANCE OLEDB PROVIDER=SQLOLEDB.1 REQUIRED=Yes
11 USER = cschacherer DATASOURCE = "SQLSRV1"
12 PROPERTIES = ('initial catalog'=billing 'Persist Security Info'=True)
13 SCHEMA = 'DBO';
14
15 DATA work.xyz_claims;
16 SET local.claims;
17 WHERE client_code = 25 and claim_status = 'PAID';
18 netpay = allowed_amount - copay;
19 RUN;
20
21 TITLE 'Claim Frequencies Across Gender and Wellness Program Participation';
22 PROC FREQ DATA=work.xyz_claims;
23 TABLE wellness gender;
24 RUN;
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40 PROC SORT DATA=work.xyz_claims;
41 BY wellness;
42 RUN;
43
44 TITLE 'Average Netpay for Wellness "Participants" and "Non_Participants"';
45 PROC MEANS DATA=work.xyz_claims;
46 VAR netpay;
47 BY wellness;
48 RUN;
```



Learning to write the SAS code necessary to access, transform, and analyze or report on the data utilized by most business users on a day-to-day basis is entirely accessible to most anyone who is interested in learning. Historically, however, many analysts learned SAS programming only as a means to an end—to answer the analytic questions they face in their day-to-day work. Whereas many of these analysts became very proficient (even great) SAS programmers, what they probably would have preferred was the ability to derive the benefits of SAS software without having to learn the syntax of necessary to program PROC and DATA steps. This is precisely where Enterprise Guide comes into play.

## INTRODUCTION TO SAS ENTERPRISE GUIDE

SAS Enterprise Guide is a graphical user interface that allows novice (and experienced) SAS users to generate and execute SAS code without knowing all of the syntax involved in writing SAS code. For example, say you wanted to copy data from a database table into a local dataset, but you only want to include a specific subset of records. As a SAS programmer with many years of experience and significant knowledge gleaned from the likes of Aster and Seidman (1997), Carpenter (2012), Cody (2007), and Lafler (2004) you might subset these data using:

### PROC SQL

```
PROC SQL;
CREATE TABLE work.xyz_claims AS
SELECT a.*,allowed_amount - copay AS netpay
FROM local.claims
WHERE client_code = 25 and claim_status = 'PAID';
QUIT;
```

or a DATA step with a where clause

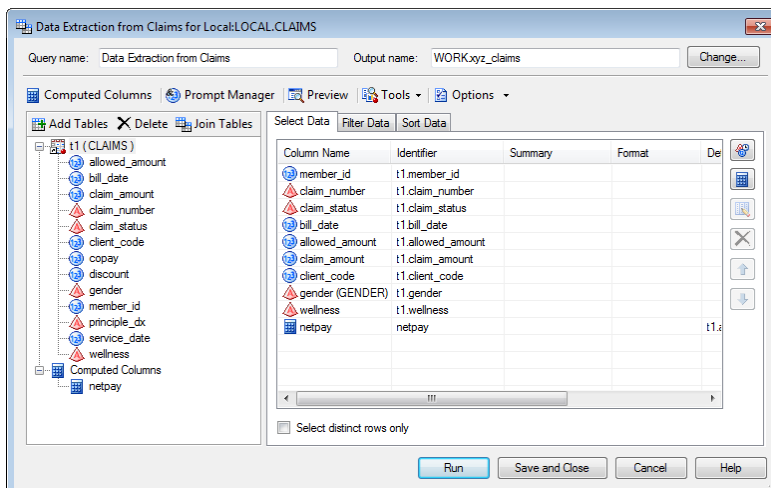
```
DATA work.xyz_claims;
SET local.claims;
WHERE client_code = 25 and claim_status = 'PAID';
netpay = allowed_amount - copay;
RUN;
```

or a DATA step with a conditional output statement

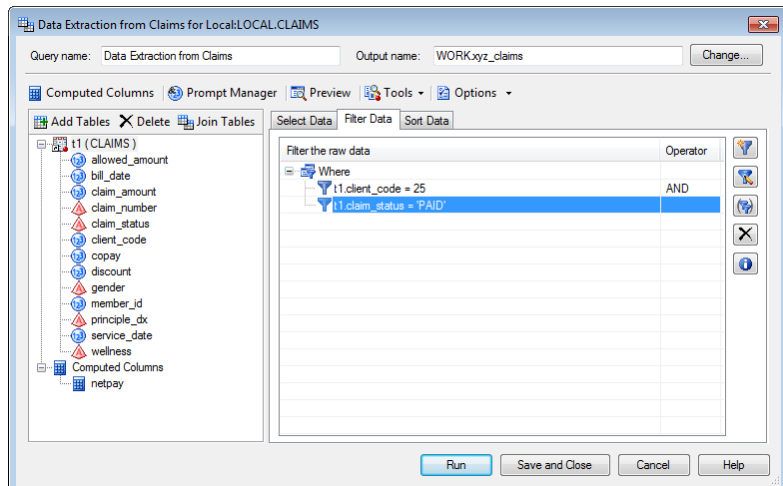
```
DATA work.xyz_claims;
SET local.claims;
netpay = allowed_amount - copay;
IF client_code = 25 and claim_status = 'PAID' THEN OUTPUT;
RUN;
```

Which way is "right"? Which way is "best"? Several factors come into play. Moreover, you need to know which lines of code need to end with a semi-colon, the difference between "QUIT" and "RUN", and why one would ever be used instead of the other. Again, all these things can be learned (and in the opinion of the author are highly valuable and inherently interesting things to learn), but you just want to produce your analytic dataset as quickly and easily as possible.

With SAS Enterprise Guide, accomplishing this same task is almost entirely intuitive. As is discussed in detail later in the paper, after selecting the source dataset and adding a Query Builder task to your Enterprise Guide project, you simply drag and drop the variables you want to include in the resulting dataset into the "Select Data" tab. Next, by clicking on "Computed Columns", you enter Enterprise Guide's point-and-click interface for creating new variables like "netpay" which are then added to the "Select Data" tab by default.



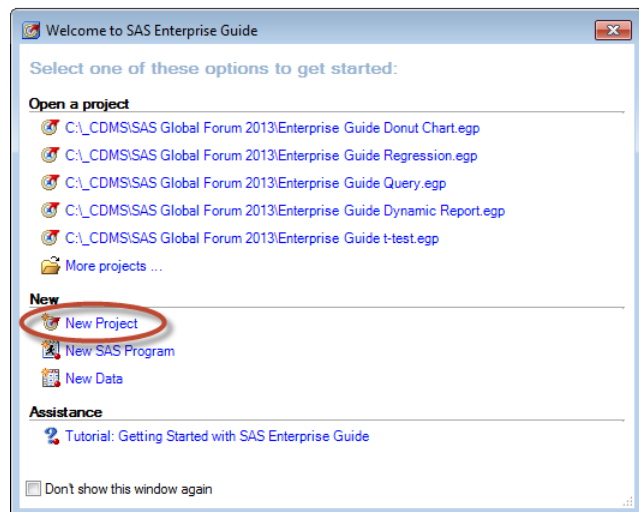
Next, simply select the "Filter Data" tab, and click on the "create filter" button to create the filtering condition(s) you wish to apply to the source data. As with computing new variables, Enterprise Guide provides a highly intuitive user interface for creating these filter conditions. When the variables have been selected and the filter conditions specified, simply click "Run" to produce the dataset "xyz\_claims" in the WORK library.



With Enterprise Guide, instead of writing the code necessary to extract, manipulate, and analyze data, you can point-and-click your way through the available data and task resources to create your desired analytic output.

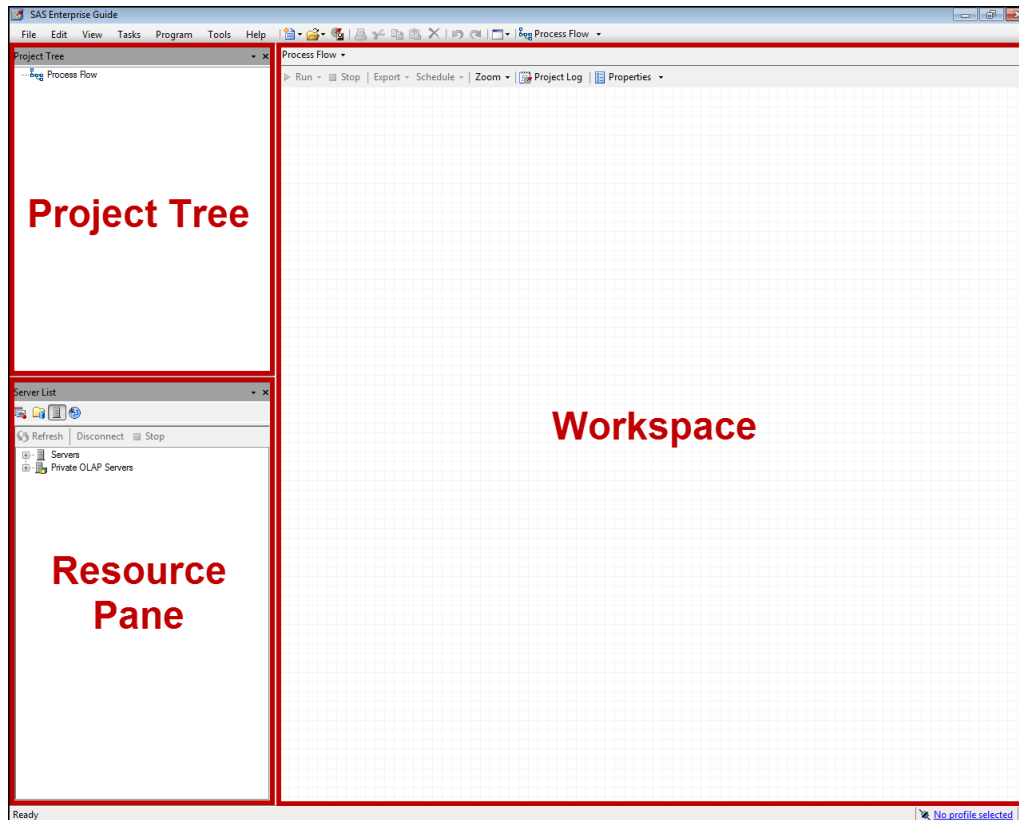
Of course, like any software package, there is a learning curve to overcome, but once you understand the basic functionality of Enterprise Guide, you can expand your analytic capabilities at a very rapid pace. A natural starting point for learning Enterprise Guide is to explore the major components of 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 launching Enterprise Guide, you are presented with some new nomenclature. Do you want to open an existing project or start a new project? As discussed previously, the work done using the SAS programming language is organized into "programs", so you might ask yourself "Is a Project the same thing as a Program?" Not exactly; Slaughter and Delwiche (2010) define an Enterprise Guide **Project** as "a collection of related data, tasks, results, programs, and notes". These items are logically related in that they are all included in the project file because they contribute to the goal of the project—to produce a specific report, generate analytic results, clean and merge datasets, etc. In short, a project combines all of the things needed to perform a discrete unit of work. Of course, these are the same things that a SAS "program" does, so whereas there are technical differences between a "program" and a "project", you could think of a "project" as being the same as a "program". The biggest difference, of course, is the manner in which you develop an Enterprise Guide project.



After choosing "New Project" to navigate past the splash-screen<sup>4</sup>, you are presented with a screen comprised of two docked windows (the **Resource Pane** and the **Project Tree**) and the **Workspace**—which technically is not a "window" since it cannot be closed, minimized, etc. independently from the application.

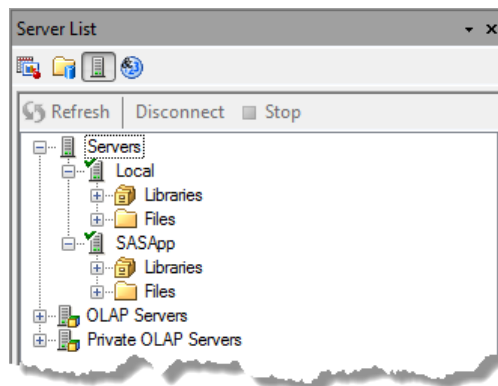
<sup>4</sup>Of course, if at some point you checked "Do not show this window again", you will no longer see the splash-screen at start-up, but will instead be taken directly to a new project.



## RESOURCE PANE.

The Resource Pane is located in the lower left corner of the Enterprise Guide interface. This window, as its name suggests, organizes the many "resources" available to the user. The four main types of resources available to you are: (1) Servers, (2) SAS Folders, (3) Tasks, and (4) Prompts.

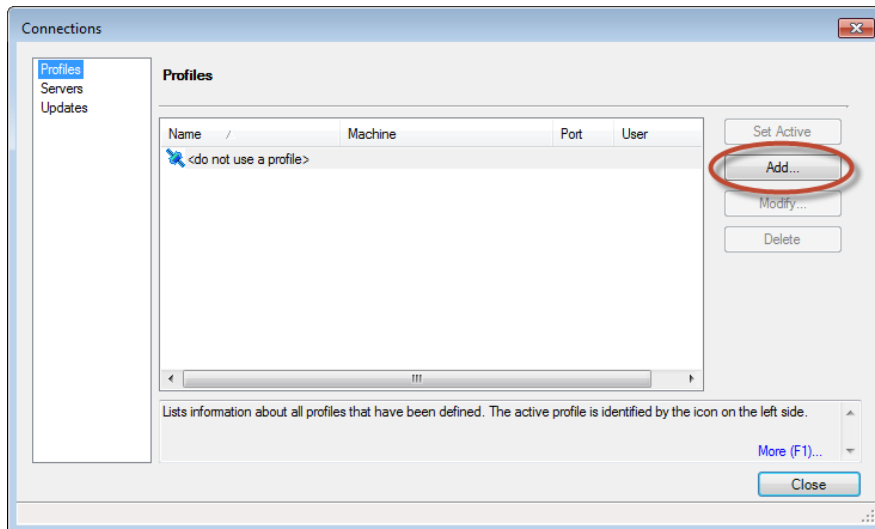
**1. Servers.** As mentioned previously, Enterprise Guide is an incredibly flexible tool that allows you to generate and execute SAS code. In order to execute the SAS code you generate, you need access to a SAS engine; there are two ways in which you can get the necessary access. First, if you have the PC SAS installed on your workstation, you will have access to a "server" named "Local". Local simply refers to your local installation of the SAS software with the Display Manager interface. If "Local" is the only server to which you have access, then when you run the code generated by your Enterprise Guide project, it will be executed on your local installation of PC SAS. However, as Enterprise Guide is part of the SAS Business Intelligence Platform, you may not have a local installation of PC SAS and will be executing your SAS code against a SAS Workspace Server (or Pooled, OLAP, or Stored Process Server—depending on the type of object you are accessing). The workspace server (SASApp, in the current example) is a centrally-administered SAS server that enables multiple users to simultaneously access data content and execute SAS code. Some of the following discussion of Enterprise Guide assumes that you are working in a SAS BI environment and are utilizing a workspace server, but the manipulation of data with Enterprise Guide "tasks" is the same whether you are working on a workspace server or a Local installation of SAS.



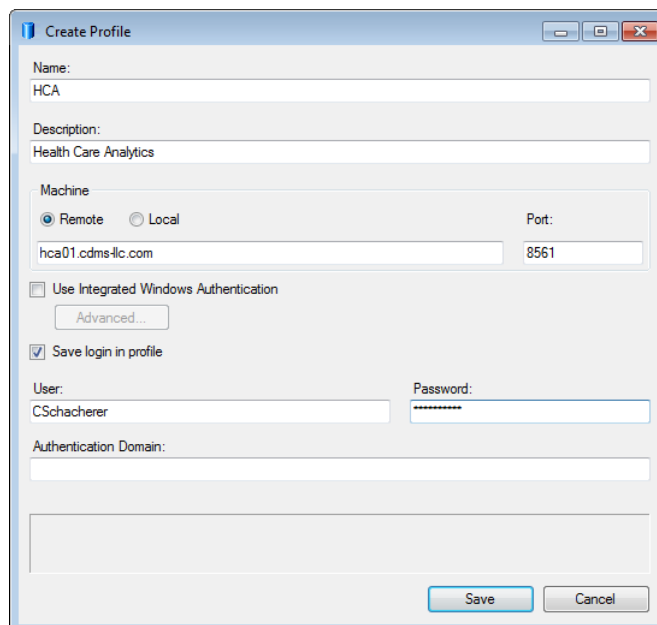
Assuming that you are working in a SAS BI environment, the first thing you will need to do in order to use Enterprise Guide is ask your SAS Administrator to create a SAS Identity for you within that environment. There is probably an authorization process within your organization that specifies which "Roles" you will be assigned within the environment (e.g., Marketing Analyst, Finance Report Writer, etc.). Typically, roles are used to specify the data content to which you have access and they can also (depending on how your environment is administered) limit the

functionality available to you within Enterprise Guide. Once you have your SAS Identity assigned by the SAS Server Administrator, you will hopefully have Enterprise Guide installed from a custom deployment that contains all of the details necessary to connect your installation of Enterprise Guide to the appropriate metadata server (see Smith, 2012 for a discussion of this topic). If this information is not part of your organization's deployment package, when you first launch Enterprise Guide, you will need to create a "Connection Profile" to identify the SAS Metadata Server to which you will be connecting. As described by Hemedinger (2012), the connection profile is "like a telephone number that connects you to your SAS environment".

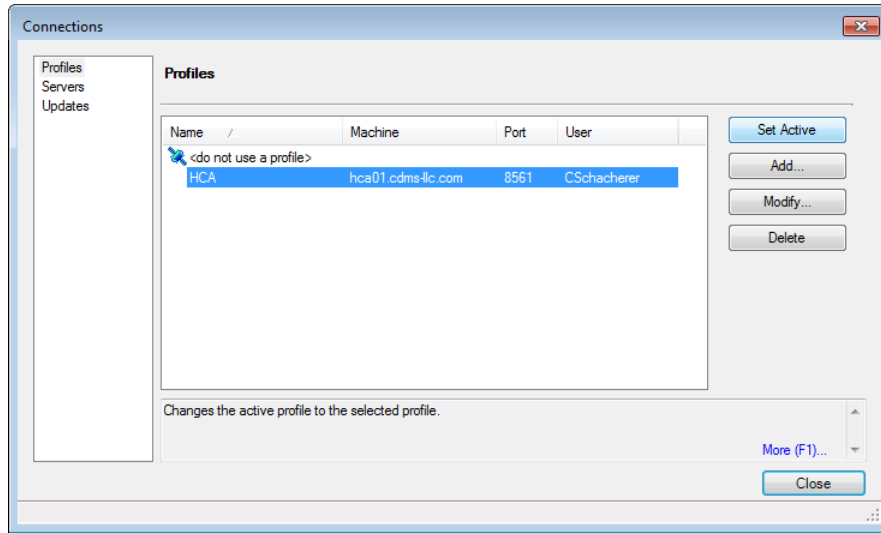
To create a new connection profile in Enterprise Guide, navigate to the Connections Dialog by navigating to Tools► SAS Enterprise Guide Explorer► File► Manage Profiles. If your installation of Enterprise Guide does not have any profiles created, click "Add" to create a profile.



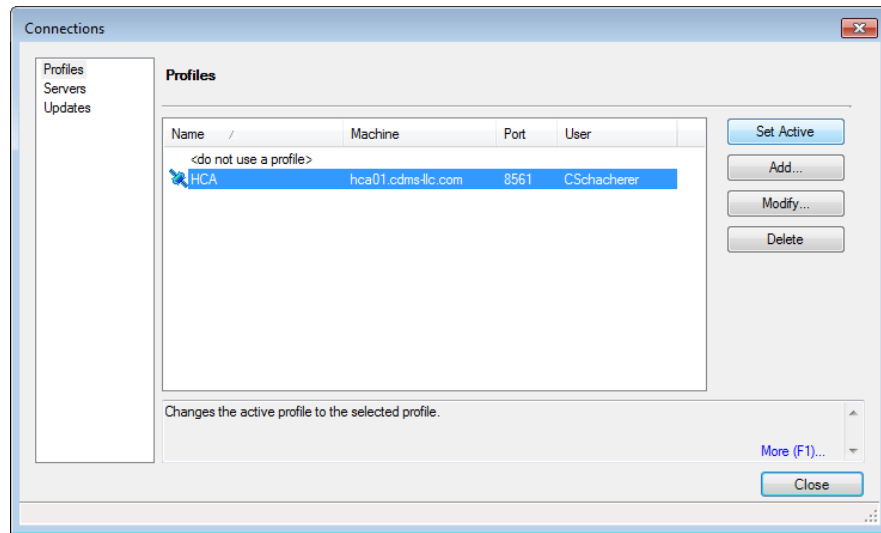
In the "Create Profile" dialog, provide a Name and Description for your connection profile. Next, specify the fully qualified name of the SAS Metadata Server to which you are connecting along with the port number on which the server is listening for connection requests (your SAS administrator can provide this information for you). If your SAS BI Platform environment utilizes Integrated Windows Authentication you will be able to connect to the server with your Windows network logon, otherwise, provide the user ID and password provided by your Server Administrator and decide whether or not to save your login information in your profile. Again, depending on your SAS environment, the "Save login in profile" option may be disabled, forcing you to provide your password each time you log on to the SAS Server.



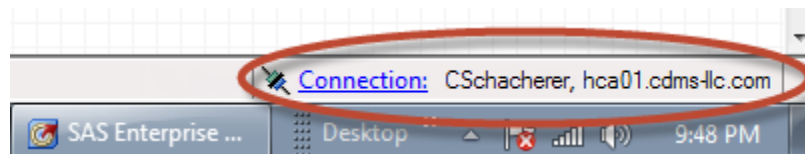
With the information for your profile entered, save the profile, highlight the new profile in the Connections dialog, and click "Set Active".



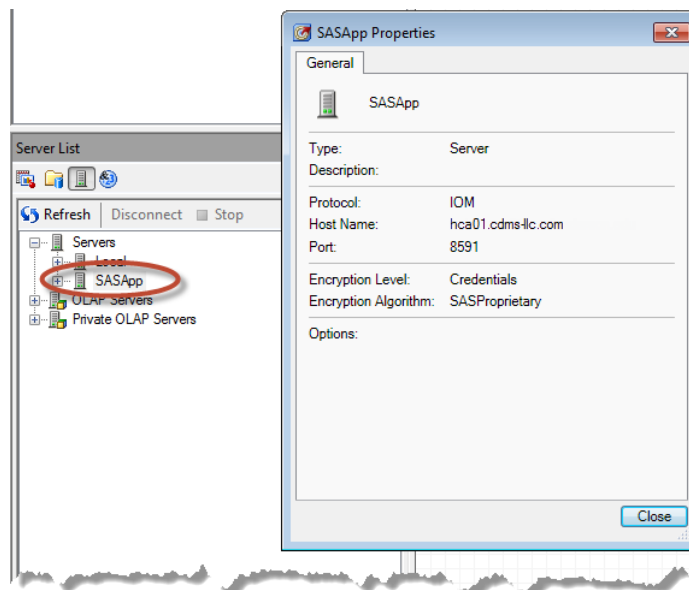
Once a profile is set to be the active (or default) profile within Enterprise Guide, Enterprise Guide attempts to make a connection to the specified server, and, when successfully connected, indicates this profile as the current connection with the blue connection icon.



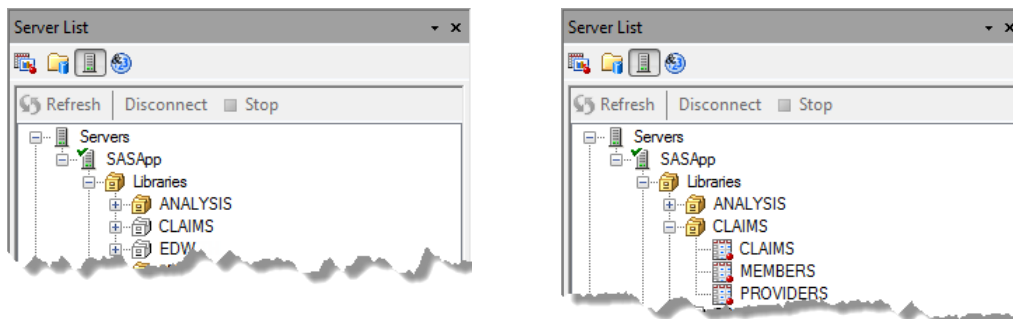
Once the Connections dialog is closed, you will also notice the current connection information in the lower right hand corner of the Enterprise Guide application interface. If you want to return to the Connections dialog to switch connections or create a new connection, simply click the "Connection" hyperlink.



Similarly, you can determine the identity of a server (in this case, the default workspace server) by right-clicking on it in the server resource and selecting "Properties".



In the SAS Intelligence Platform the connection to the SAS Metadata Server is the key to accessing the data assets with which you will be working. Behind the scenes, your connection to the SAS Metadata Server gives you access to metadata—or, data about data—which is used to define data objects, organize them into logical groupings and define the groups of users who can access them. One of the object types with which you may interact is the **SAS Library**. Conceptually, the SAS Libraries defined on the SAS Server are the same as those defined via PC SAS; that is, they are shortcuts to data storage locations. By default, you will have access to the temporary library WORK—just as you would if you were using PC SAS. In addition, depending on the metadata objects to which you have been granted access, you may see other SAS libraries as well. In the following example, the connected user has access to the "Analysis", "Claims", and "EDW" libraries. As in the earlier examples of SAS libraries defined in the Display Manager, these libraries might be directories containing SAS datasets, connections to Oracle database schemas, connections to SQL Server databases, etc. To the Enterprise Guide user who has access to these disparate data sources, the data will all appear as datasets within the individual libraries. Notice, however, that the Analysis library is yellow whereas the Claims and EDW libraries are grayed-out and appear to be inactive. The Analysis library in this case has been defined in the metadata as being "assigned" by default (or, "PreAssigned"), meaning that the user is actively connected to these data automatically when they are authenticated by the Metadata Server. The Claims and EDW libraries, on the other hand, are defined as "Unassigned" by default—meaning that the connection to the back-end data sources (e.g., Oracle, SQL Server) are not "Assigned" automatically, but are made, instead, when the user first attempts to access them. As is depicted below, once data within the "CLAIMS" library is accessed by the Enterprise Guide project, it changes color to indicate that the library is now "Assigned".<sup>5</sup>

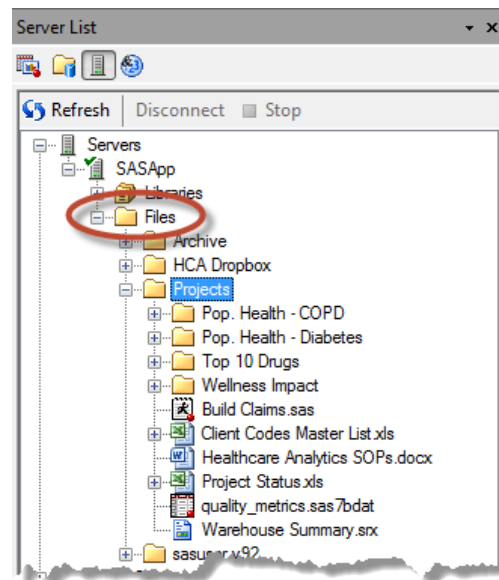


<sup>5</sup> In addition to libraries managed by the SAS Server Administrator, Enterprise Guide users can also (when not restricted from doing so) create their own librefs by issuing libname statements within SAS programs embedded in their Enterprise Guide projects (see, Schacherer, 2012a for examples).

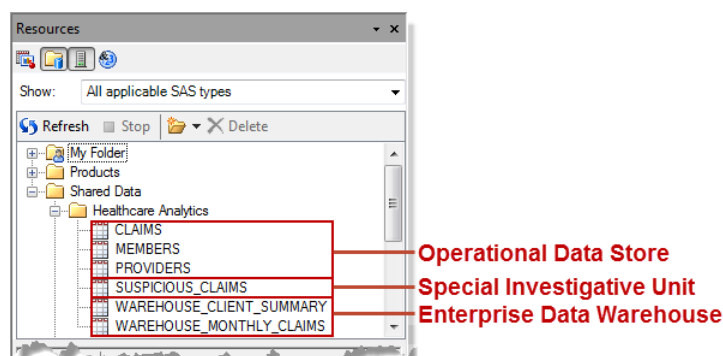


In addition to providing users access to SAS libraries and their associated datasets, the SAS Server also provides users with access to file types not accessible through SAS libraries. Via the **Files** directory, you can access Excel files, Access databases, and text files, for example, and, as you will see in a later example, you can easily include these data files in your Enterprise Guide project. This is a significant advantage over SAS Display Manager because it allows you to simply drag and drop these files into your project and rapidly integrate the data into your analysis or report with just a few mouse clicks. In a SAS program, a separate DATA or PROC step must be written to import the data and the procedures for importing different data types vary—sometimes in significant ways.

The location of the Files directory is established by your SAS Administrator, but is simply an operating system directory structure to which you have access via the SAS server. By default, the Files directory is pointed to an individual user's home directory on the server, but could be assigned at a higher level of the server directory structure to facilitate collaborative work among different groups in the organization (Overton, 2012). In addition to using this directory for storing and organizing datasets saved in other file types, you can also use this directory to store and organize your Enterprise Guide projects, analytic output, etc.



**2. SAS Folders.** In addition to Libraries and Files, the SAS Intelligence Platform (of which Enterprise Guide is one of the client applications) also makes data assets available through a construct called a **SAS Folder**. Like a file folder on your local operating system, SAS Folders are used to logically organize your data assets. Unlike an operating system folder structure, however, the organized assets are not physical files written to disk, but rather **metadata objects**. Metadata, or data about data, is central to the management of the SAS Intelligence Platform. Suppose, for example, that a group of analysts working in the Healthcare Analytics Department of a third-party processor of healthcare claims all need access to the same data assets distributed across several systems—say, the operational data store for healthcare claims data stored in SQL Server, a subset of tables in the Oracle-based enterprise data warehouse, and a SAS dataset of suspicious claims maintained by the Special Investigative Unit. These datasets and relational database tables could be accessed through individual SAS libraries established for each of these data sources, but they could also be "surfaced" to the end user through a single SAS Folder created specifically to serve the needs of that functional area. As shown in the following figure, using this latter approach, the SAS Folder "Healthcare Analytics" (a metadata object) was created and a metadata entry for each of the data objects was added to it.



The metadata objects that refer to individual datasets or relational database tables (e.g., claims, members, etc.) each store attributes about the physical table they represent—its name, metadata library through which it is accessed, etc.—as well as attributes that describe the data columns (e.g., name, data type, length, etc.). As a result, when looking at the attributes of these metadata objects, the end user is not seeing information "live" from the actual source data, but rather the attributes stored in the metadata for that object. When the data are selected for use in an analysis, however, the metadata associated with that object is used to access the data and bring them into the Enterprise Guide project.

In addition to organizing raw data sources like SAS datasets and relational database tables, SAS Folders can also be used to organize other types of metadata objects to which you might have access in your organization's SAS Intelligence Platform. Although the current work does not discuss specific examples of each data object type, they are described briefly here.

**SAS Reports** are XML files produced by tools such as SAS Web Report Studio and Enterprise Guide to provide a specific, organized view of data that facilitates an understanding of a defined business process (e.g., number of claims processed per client organization, total revenue by region, etc.). The layout of the output is designed by a report writer and is meant to reliably convey information from the underlying data in a consistent format. Reports are generally utilized by end-users who are interesting in understanding trends within an organization by reviewing fluctuations in specific metrics (e.g. revenue from specific lines of business over time, costs incurred by specific business units, etc.). Although reports can be built to enable end-users to specify parameters for their analysis (e.g., revenue for a specific region), the layout of the content remains fixed. Reports are intended for users who want to consume a consistent, fixed view of the data in which they are interested.

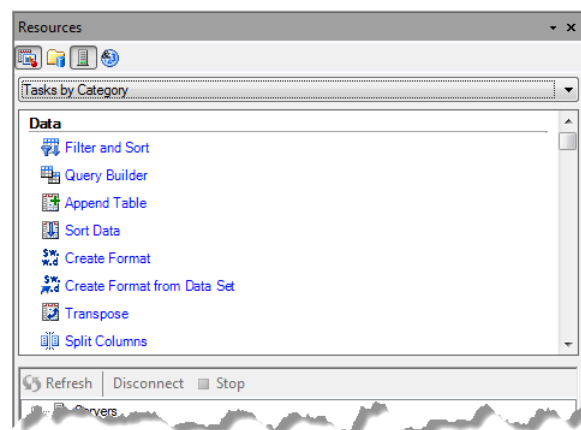
**Information Maps** are SAS content objects that pre-define both (a) relationships among data tables and (b) dataset filters and may rename data columns to labels more familiar to the business user. As a result, they facilitate end-user queries of complex business data. For example, in trying to include data from a health plan's raw membership and claims files in an analysis, an Enterprise Guide user would need to know the columns defining the foreign key relationship between these datasets, the type of join appropriate for the analysis they want to perform, and possibly subtle differences in meaning between several similarly named database columns. With an Information Map of these same data, however, someone has already performed the appropriate join, selected the appropriate variables, and made the data available to the end user in a manner that allows them to simply select the data they want to include in their analysis. To the end-user, interacting with the Information Map is similar to interacting with a single dataset; knowledge about the underlying relationships between the source data objects and the subtleties of different operational dataset variables is not necessary—freeing the analyst to focus on the business question.

Another type of SAS metadata object that facilitates user interaction with business data is the **OLAP Cube**. OLAP Cubes provide access to aggregate data across multiple "dimensions" (or facets) of data within a performance (or, measurement) domain. Instead of performing time- and resource-intensive queries against raw data in order to sum, count, or otherwise aggregate data associated with a specific measurement, OLAP cubes contain aggregate data summarized across several "dimensions" (and within the intersections of those dimensions)—allowing cube users to rapidly access summary data for any combination of values along those dimensions. Similar queries to aggregate the data from the raw source data could take hours, but if an OLAP cube has been built for the performance domain of interest, the summary statistic you need could be just a few mouse clicks away.

A **Stored Process** "is a SAS program that is stored on a server and can be executed as required by requesting applications" (SAS, 2011, p. 1). Stored processes, for example, enable users of web reports to execute server-side SAS programs that extract data and generate refreshed report content. Separating the SAS programming logic from web application in this way facilitates change management and makes the stored process a more modular component that can be utilized by multiple applications (Aanderud & Hall, 2012). Within Enterprise Guide, stored processes can be used to repeat data manipulations required to refresh analytic datasets, generate daily, monthly, or quarterly reports based on complex data manipulations, or just about any other task that an Enterprise Guide project can accomplish. After developing the Stored Process and registering it with the metadata server, it can be made available through a SAS Folder to other users so that common processes are executed in precisely the same way across projects.

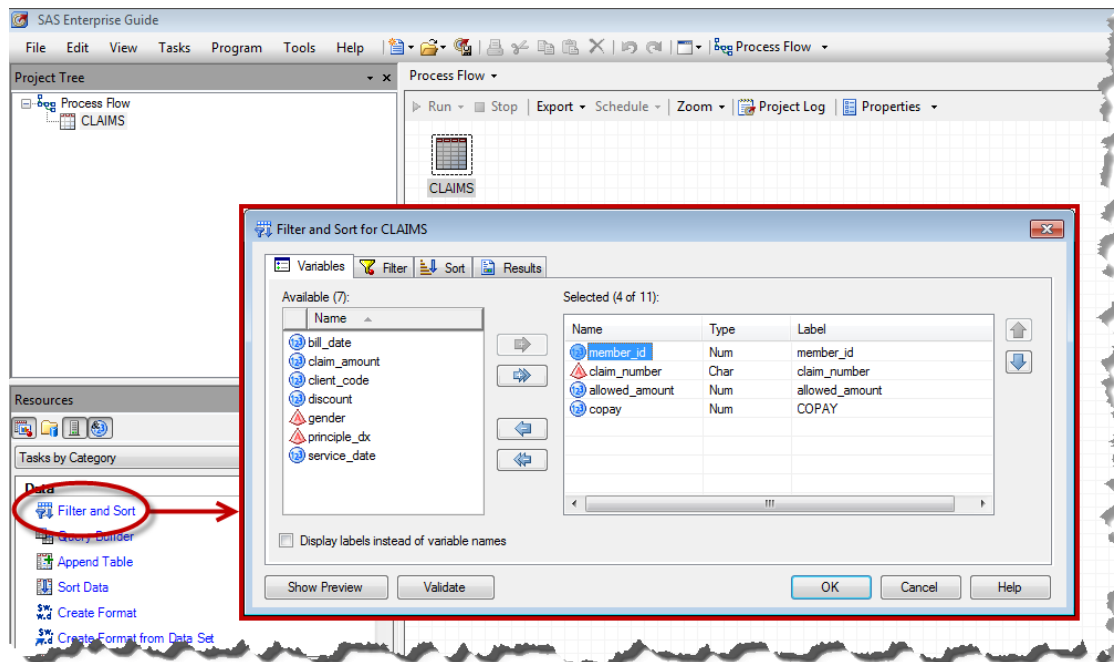
Whereas these metadata objects may facilitate your work, however, as an analyst you will likely spend a significant amount of your time manipulating and analyzing data using SAS Enterprise Guide **tasks**.

**3. SAS Enterprise Guide Tasks.** Like PROCs in the SAS programming language, **Enterprise Guide Tasks** are specialized to perform specific types of operations—filter and sort data, create bar-charts, perform t-tests, etc. Unlike PROCs, however, 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 PROC. 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. Upon discovering this relationship between Enterprise Guide and the SAS programming language, many experienced SAS programmers dismiss Enterprise Guide as not adding any significant value beyond what the SAS programming language already does.

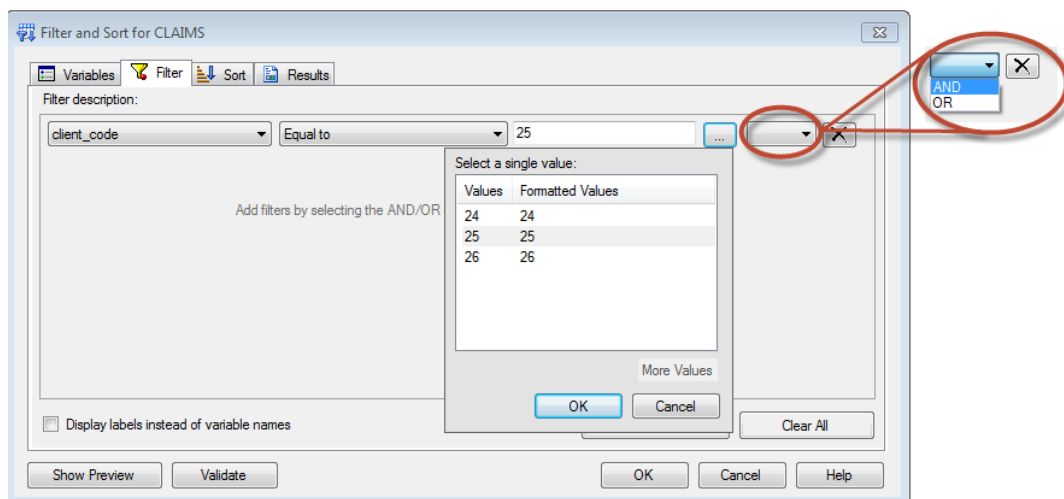


However, whereas there are arguably many reasons experienced SAS programmers should still learn how to utilize Enterprise Guide (Schacherer, 2012a), this is exactly the point. Executing an Enterprise Guide task is the same as executing a PROC step. The only significant difference between the two is the amount of time required to become proficient in their use.

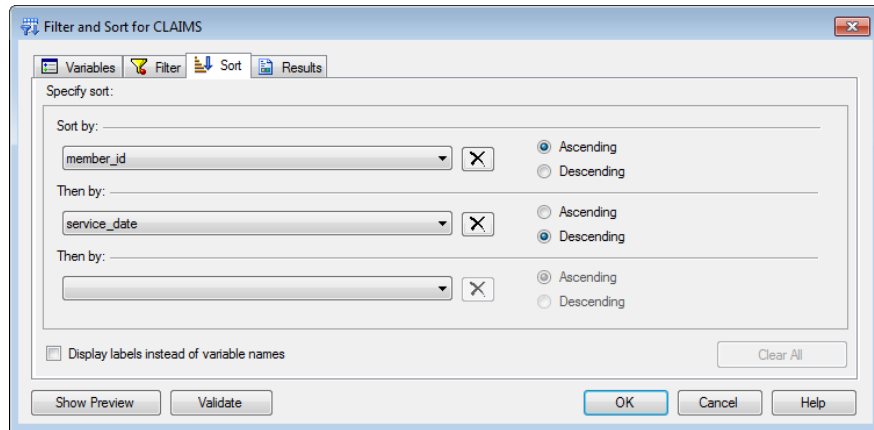
Aligned with the goal of making the power of SAS analytics accessible to users without SAS programming experience, utilizing tasks in Enterprise Guide is very simple. For example, suppose you want to filter the healthcare claims dataset "claims" to include only those claims generated by employees of your client company "XYZ Co." (client code "25" in your claims dataset). As depicted in the following figure, after selecting the dataset and clicking on the "Filter & Sort" task, you enter the Filter and Sort task dialog. The "Variables" tab allows you to select the variables that you wish to include in the filtered dataset.



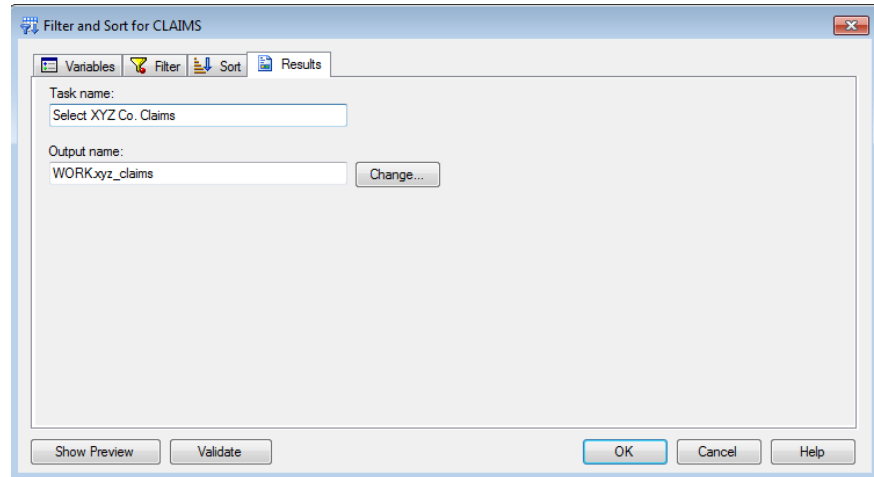
Next, navigate to the "Filter" tab and specify the filtering condition(s) you wish to apply to the source dataset. In this case, the output dataset will contain only those records containing a "client\_code" value of "25", but notice how the interface facilitates the specification of the filter criteria. In addition to providing a drop-down list of every variable in the source dataset and a complete list of potential operators (Equal to, Not Equal to, etc.), once a variable is selected for the filter condition, clicking on the ellipsis button [...] brings up a list of the unique values of that variable in the dataset. If you wish to add another condition to your filter, there is a drop-down list that allows you to specify whether both conditions must be true in order for a record to be written to the output dataset or if either criterion is sufficient for entry into the dataset.



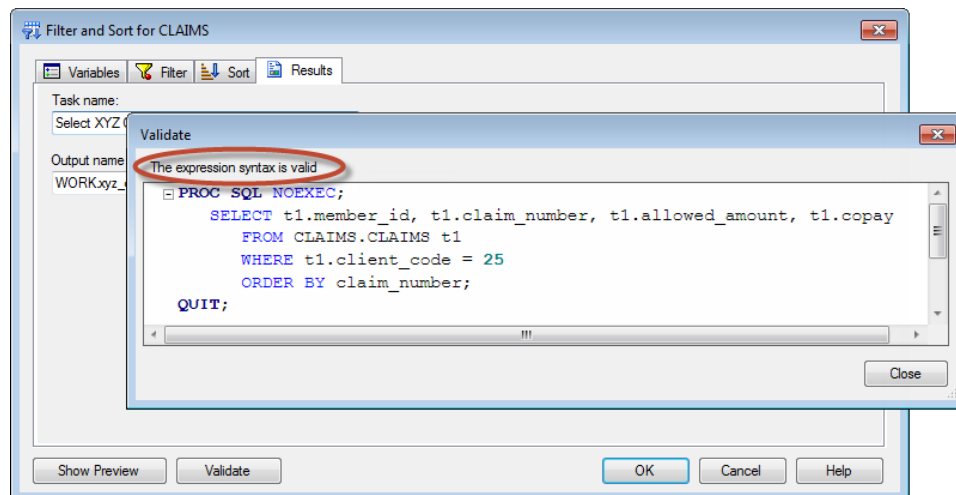
If you choose to sort your output, you can navigate to the "Sort" tab and choose one or more variables on which to sort the output dataset—as well as the order in which to sort the records. In this example, the records will be sorted in ascending order of "member\_id", and within a given member\_id value records will be sorted in descending order of "service\_date".



Finally, on the "Results" tab, you can give your task a name to identify it in the Workspace and specify the name of the output dataset along with the library to which it will be written. At this point, you can click "OK" to execute the task and create your output dataset of claims generated by XYZ Co. employees, sorted member\_id and service\_date. However, if you first click the "Validate" button, the nature of the relationship between Enterprise Guide tasks and SAS PROCs is illuminated.



As demonstrated below, clicking on the "Validate" button shows the SAS code that will be executed as a result of the selections made in the Filter and Sort task dialog. This example reveals the essence of Enterprise Guide tasks as user interfaces that help you use SAS PROCs by soliciting from you information about how you want a given SAS PROC to be executed and then generating the appropriate SAS code to execute the PROC in that manner.



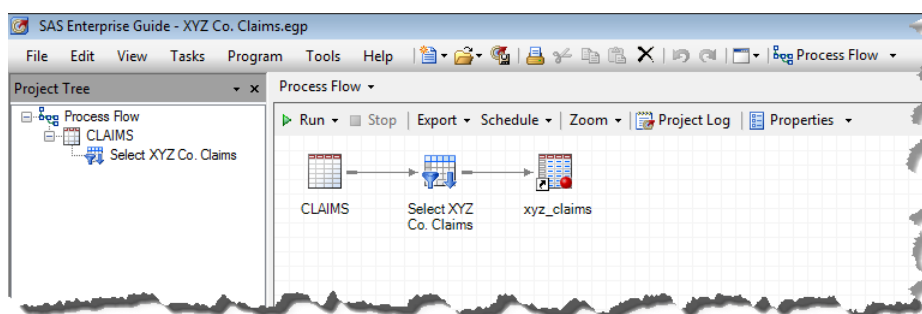
As a result of these user-interfaces built on top of the PROCs, the time it takes to become proficient with EG is a fraction of the time it would take you to learn the same techniques using the SAS programming language!

**4. Prompts.** The final type of resource to be introduced is the "Prompt". Prompts play an important role in creating dynamic solutions for analysts and report writers. This resource provides a flexible way to present users of the project with a means of specifying parameters that drive its execution. For example, the values provided through prompts can be used as criteria for subsetting datasets, as passwords enabling access to source data systems, or to specify which of a number of graphs or charts are to be generated as part of a report. An example of the use of prompts is provided later in the paper, but now that you have an understanding of Server, Folder, Task, and Prompt resources, you can start to put them together in a project in the SAS Enterprise Guide **Workspace**.

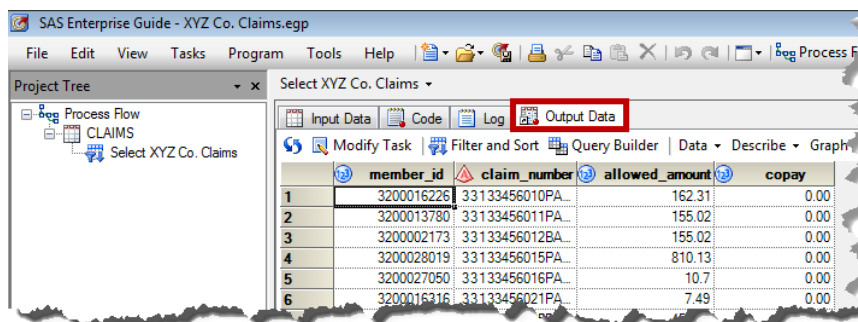
**WORKSPACE.**

The workspace contains **Process Flows** and **Document Windows**. A process flow is a construct that helps organize the resources included in a project; it is the part of the Enterprise Guide interface where you will perform the development work associated with your projects.

Process flows get their name from the fact that they graphically depict the sequence of program logic execution. For example, upon completion of the Filter and Sort task in the previous example, the process flow in which the task was specified would look something like the following—depicting the relationships between the source dataset "claims", the Filter and Sort task "Select XYZ Co. Claims" and the output dataset "xyz\_claims".

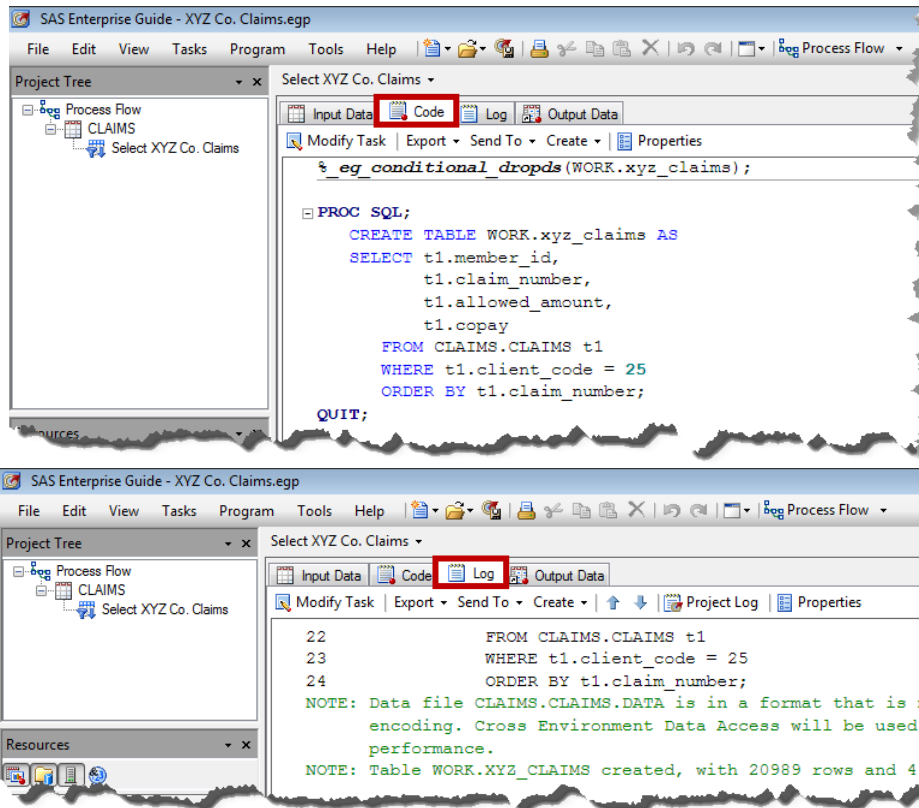


One thing that was not discussed in that previous example, however, is what happens when you execute the Filter and Sort task. Yes, the "xyz\_claims" dataset is created, but the user experience (as encountered with the default Enterprise Guide configuration) can be a bit disconcerting to new users. When you execute the task by clicking "OK" in the task dialog, a **Document Window** containing the resulting dataset appears in the workspace—replacing the process flow that contains the task you just executed.

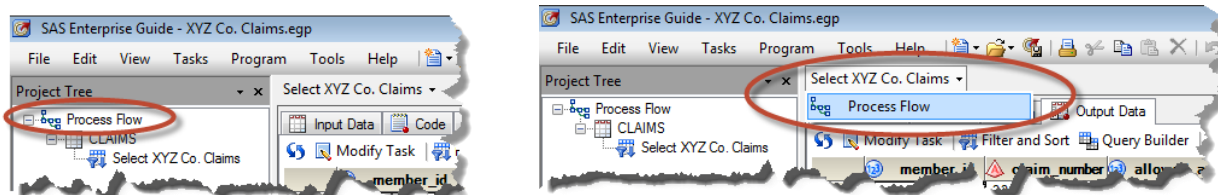


Like the "Validate" function in the previous example, this document window also demonstrates the link between Enterprise Guide and the SAS programming language. In the following two figures, you can see that in addition to the output dataset, the document window also contains the SAS code generated by the task, and the log entries generated by execution of that SAS code.

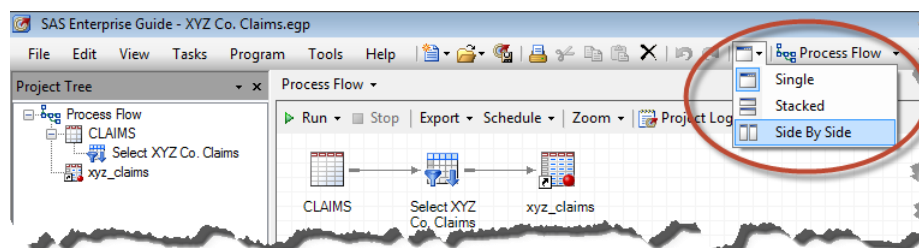




But where is the process flow with which you were just working? The process flow from which you executed the task is still in the Workspace, but it has been relegated to a layer "behind" the current document window. To navigate back to the process flow (or, bring it back to the "front" layer of the Workspace), you can either double-click on the process flow in the Project Tree or select the process flow from the drop-down that lists all process flows and document windows active in the project<sup>6</sup>.



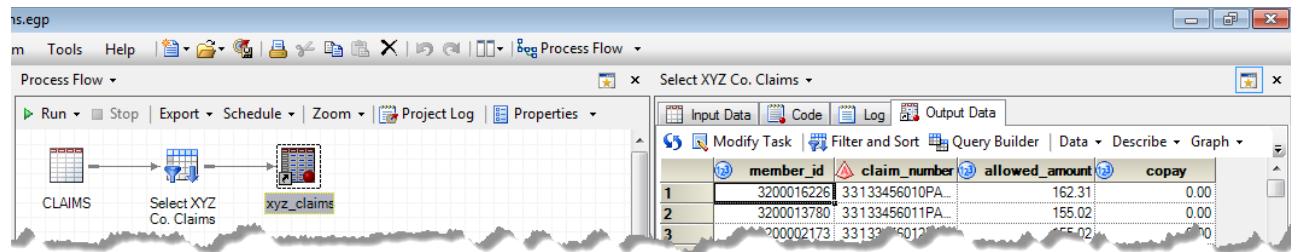
One alternative to your Enterprise Guide configuration that may make working in the Workspace a bit easier is to change the Workspace Layout. Using the Workspace Layout button on the Workspace toolbar, you can choose to change your single-view layout of the Workspace to either two stacked views or two side-by-side views.



In the current example, the "Side by Side" option is selected and now as more process flows and document windows are added to your project, you have the capability to view any two of these objects simultaneously.

<sup>6</sup> A third option is to simply close the document window by clicking the "X" in the upper right corner of the document window.

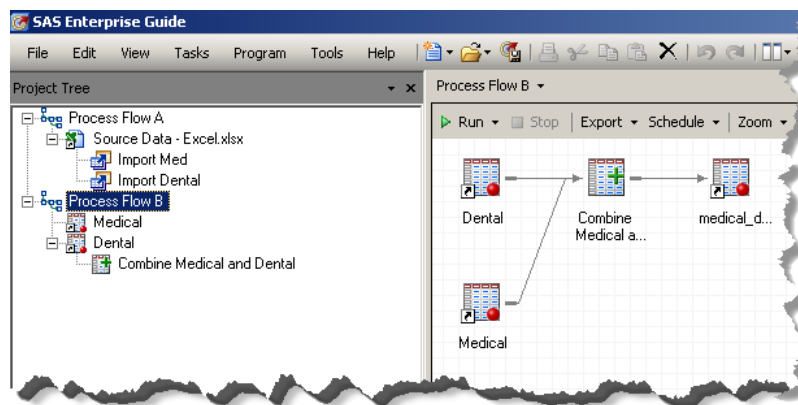




To summarize the purpose of the Workspace, it serves as a container for **Process Flows** that are used to organize project resources and **Document Windows** that are used to view task results and other documents associated with the project.

### PROJECT TREE.

The last part of the Enterprise Guide interface is the **Project Tree**. The Project Tree serves to organize the project by providing quick reference to the lineage of datasets, the tasks that use and generate those datasets, and the process flows used to organize the tasks in logical groups within the project. In the following project tree, we see that the tasks "Import Med" and "Import Dental" are both part of "Process Flow A". The output datasets "Medical" and "Dental" are used in "Process Flow B", and you can see from the depiction of "Process Flow B" in the Workspace, that these datasets are both inputs to the task "Combine Medical and Dental". Through careful naming of tasks and datasets and the logical grouping of tasks within process flows, the Project Tree can be a useful tool for quickly navigating a complex project.



## EXAMPLE PROJECT – WELLNESS PROGRAM IMPACTS ON HEALTHCARE COSTS

As demonstrated in the previous examples, the work performed in a process flow involves applying tasks to source data. By sequentially specifying the dataset transformations, joins, and analytic tasks applied to these datasets—with the outputs of one task acting as inputs to subsequent tasks—the desired analytic outcome can be achieved in a tractable fashion. These process flows, in turn, are saved as an Enterprise Guide project that achieves some specific analytic goal. The following Enterprise Guide example project demonstrates a number of steps that you might find useful in your own projects. In this example healthcare claims from health plan members who participate in a wellness program are compared to the claims of members who did not participate in the program.

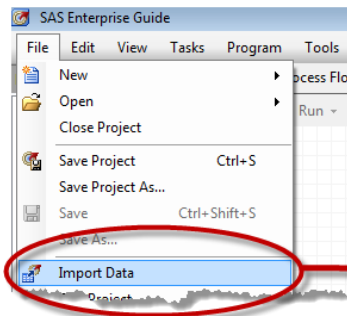
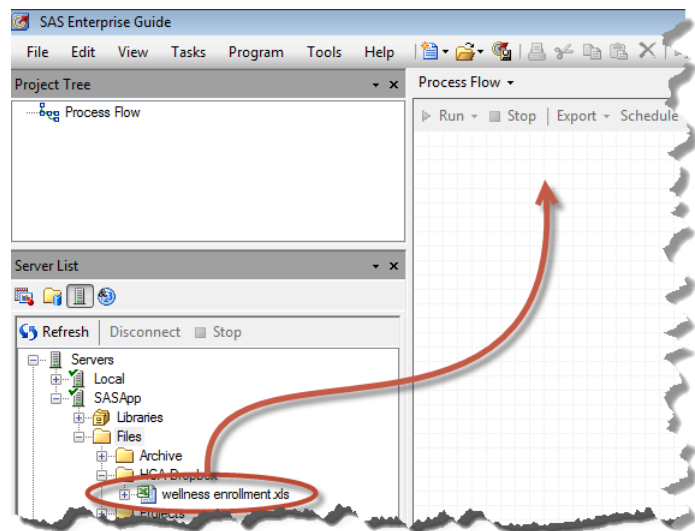
The project begins with importing an Excel file containing data that can be used to identify wellness program participants. Next the healthcare claims data are brought into the project from a table surfaced through the SAS Folder "Healthcare Analytics". These two datasets are joined together using a Query Builder task which limits the resulting dataset to paid claims generated by employees of XYZ Co. The resulting dataset is then used to generate a series of graphs and analyses that provide information about the impact of wellness programs on healthcare costs at XYZ Co.<sup>7</sup>

<sup>7</sup> Please note that these are fictitious data generated programmatically solely for the purpose of this demonstration. They are not intended to inform the reader about the costs of healthcare or the impact of wellness programs on those costs.

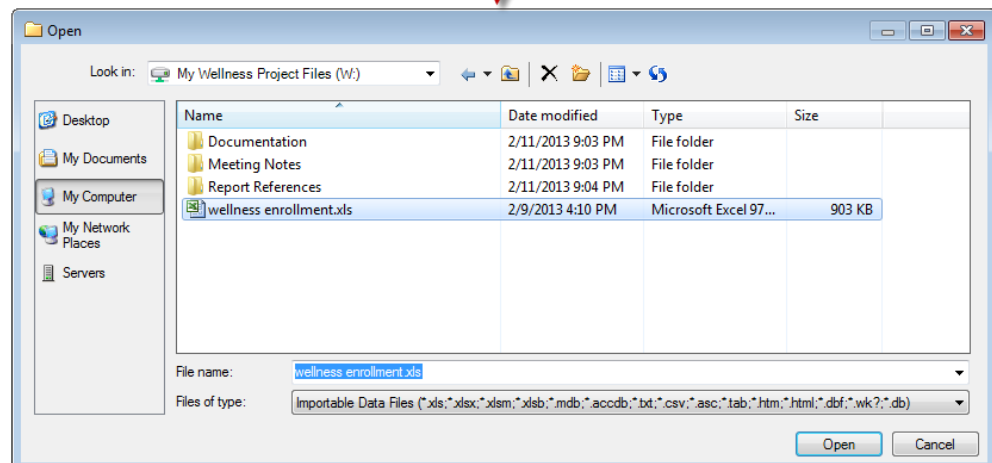
## IMPORTING DATA FROM FILETYPES OTHER THAN SAS

As in the real world, the data we want to analyze are not always made available to us as nice tidy SAS datasets created expressly for the purpose of the analyses we want to perform. In this particular case some of the data you need for your analysis might be provided by another department and made available to you as an Excel file containing the membership IDs of the health plan's wellness program participants from each client company.

Importing the Excel file (as well as other file types such as text, CSV, MS Access, etc.) is very simple with Enterprise Guide. If your workspace server is configured to support collaborative work across your organization, it might be possible to navigate to the file through the "Files" resource on your workspace server (as in the example on the right) and simply drag the Excel file onto your process flow and

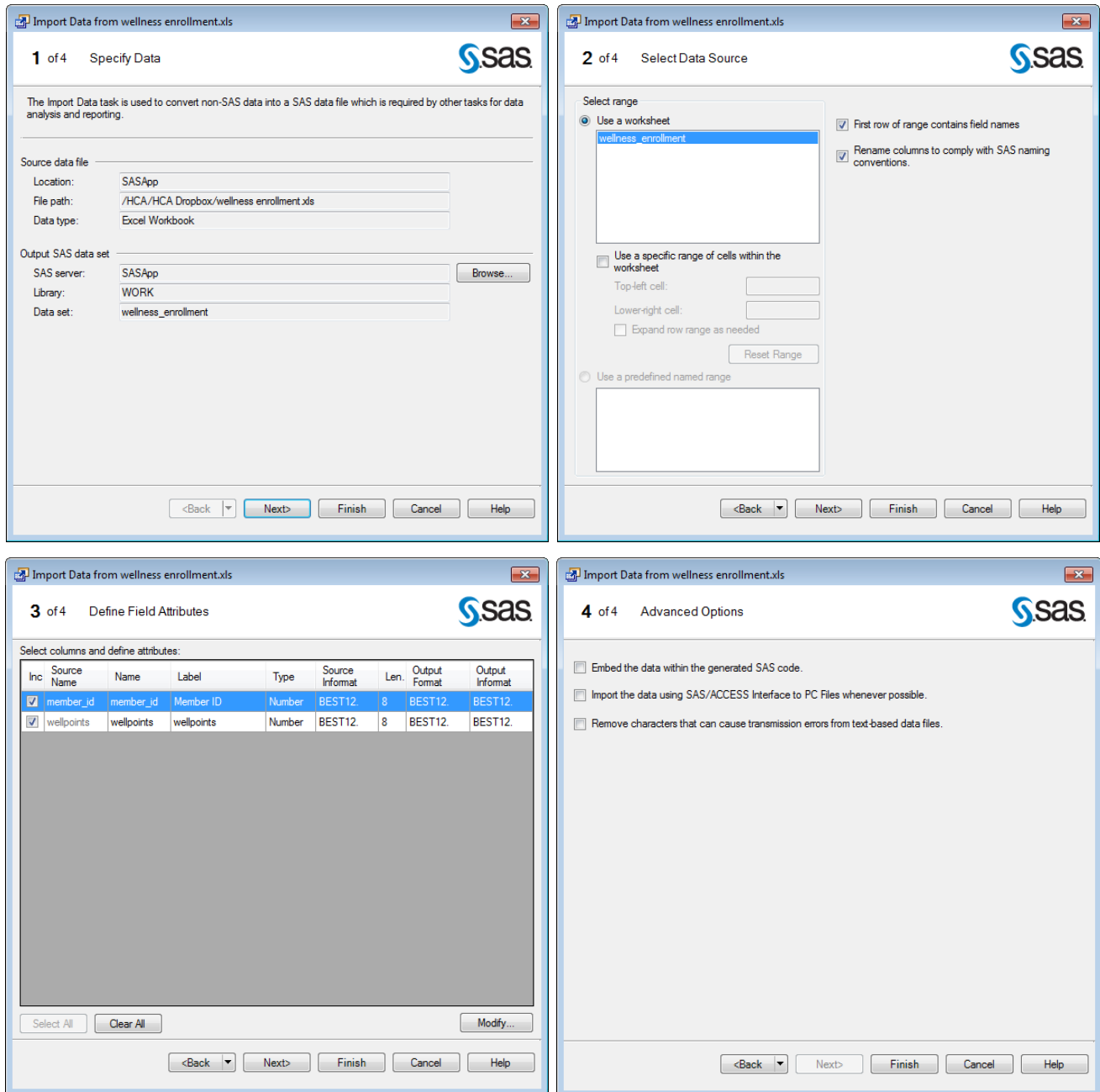


Alternatively, if you have the file on a local disk or LAN directory mapped to your local machine, simply use the "Import Data" task to navigate to the file and open it.

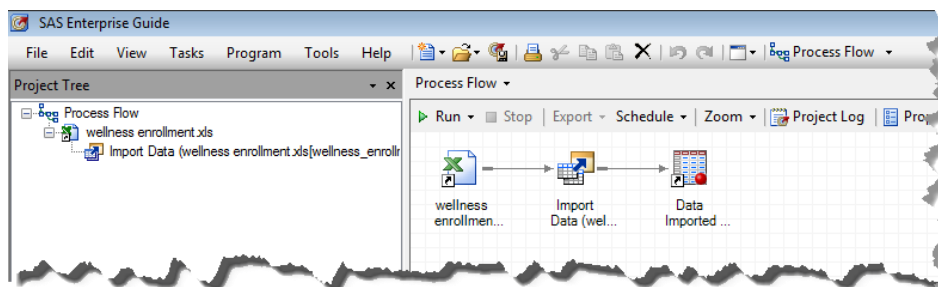


These two approaches to importing data from an external file highlight an important characteristic of the "Files" resource; it is a physical directory on the workspace server. By default, when the workspace server is installed, each user is assigned a directory of their own in which they can save output, projects, etc. Some installations are customized, however, to facilitate workgroup collaboration by moving the default "Files" directory higher in the directory structure hierarchy to allow for both shared and private directories (Overton, 2012).

Regardless of how you locate the external file, once the external file is opened or dragged onto the process flow, Enterprise Guide automatically detects the type of file being imported and launches the appropriate Import Data task. This task walks the user through a four-step wizard to (1) specify the source data file and output dataset, (2) select the Excel worksheet to import from the file, (3) specify data types, SAS variable names, informat, and formats for the imported data, and (4) specify advanced options associated with the current task.



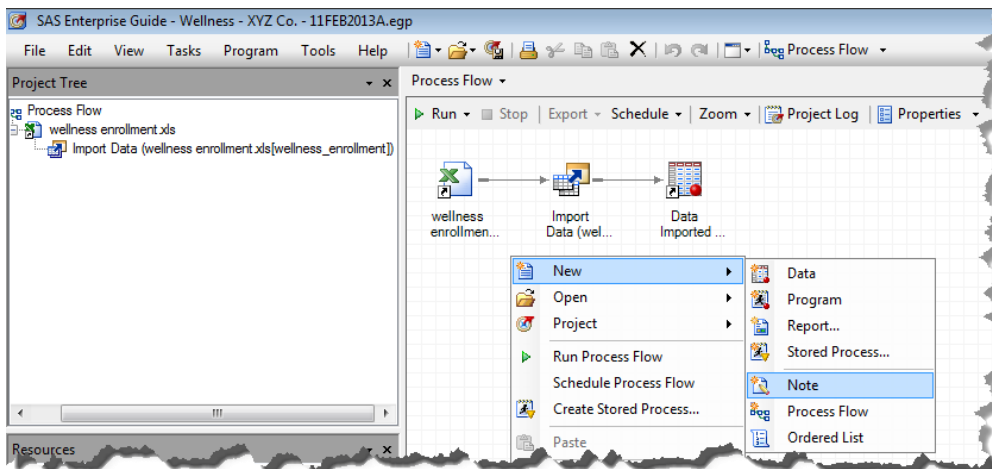
Having executed the Import Data task, your project now contains a logical reference to the Excel file, the Import Data task, and the SAS dataset "wellness\_enrollment".



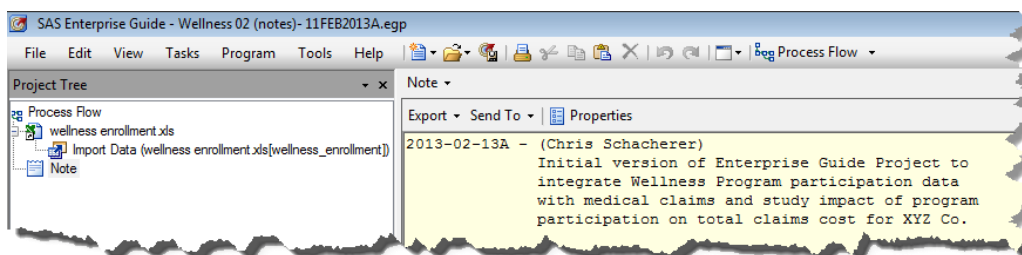
## PROJECT COMMENTING.

Despite the usefulness of process flows in understanding the linear sequence of events within an Enterprise Guide project, one of the keys to successfully managing a project over time and making it possible for others to use it in the future is documenting what the project does, how it does it, and why you are doing it in the first place. As in Base SAS programming, this is one of the most over-looked aspects of developing an analytic solution, and the consequences of ignoring it can be significant in terms of lost time and irreproducible reports and analyses. The rationalizations given are numerous and frequently repeated—"This is just a one-time thing I need to do quickly; I'll never need to do this particular analysis/report again", "There is no way I will forget why we use that set of criteria to define the dataset", etc. Despite these seeming truisms, however, ad-hoc requests become standard requests, memories fade, and people change jobs—leaving others in charge of their previous analytic responsibilities.

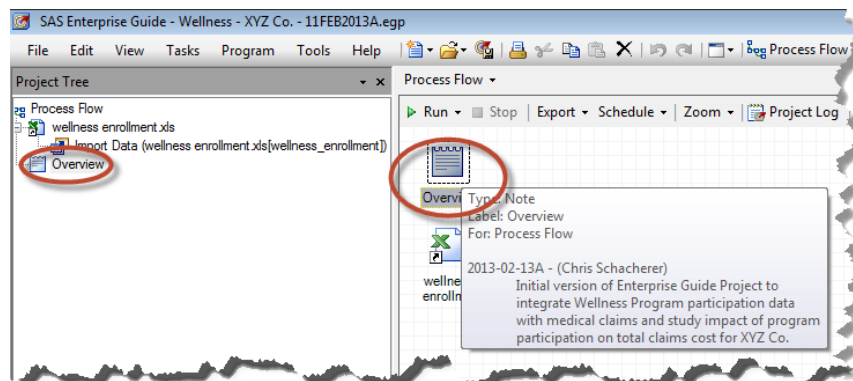
In order to facilitate the documenting of your project, Enterprise Guide allows you to attach **Notes** to your project. To add a note to a process flow, you simply right-click in the process flow and choose **New ► Note**.



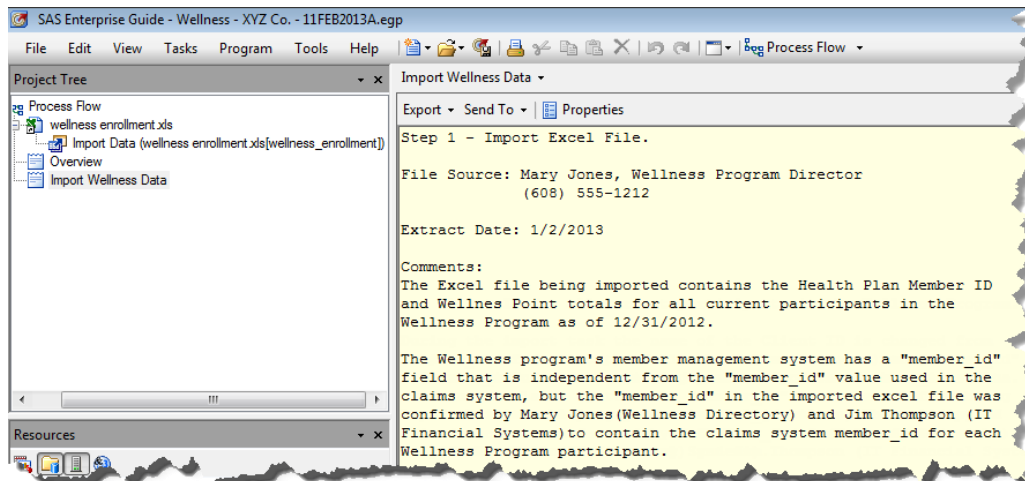
A blank note opens in a document window, and you can record the relevant information about the current process flow or project. Although each organization has its own unique requirements for what constitutes an acceptable level of documentation of SAS programs and Enterprise Guide projects, at the very least such comments should include the date a project was created, its purpose, the effective dates and descriptions of changes to the project, the name of the person who is responsible for the project.



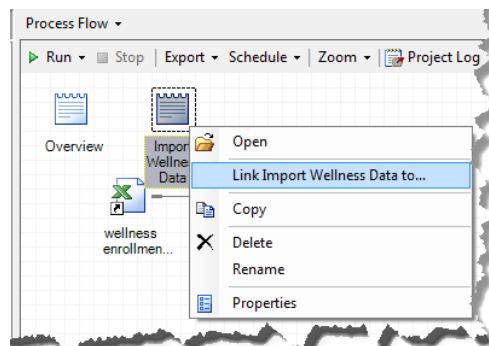
Once you complete the note and close the document window, the note appears in your process flow (as well as in the Project Tree), and as you hover over the note, its contents are displayed in a tooltip.



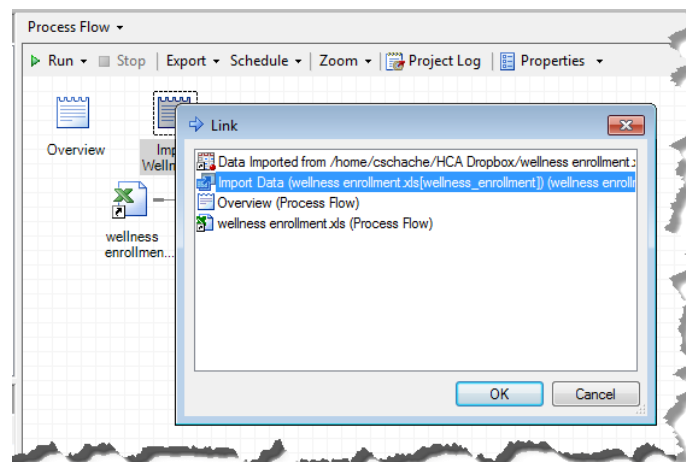
Beyond creating an Overview to describe the original purpose of an Enterprise Guide project, however, it is also a good idea to provide background information on individual tasks within the project. This not only helps future users of the program to understand how you approached the project, but can also provide you with valuable information when you want to modify the program in the future. In this project, for example, it might be helpful to describe the source of the Excel file and transformations performed on it as it was imported. Even though this information is fresh in your mind now, a year from now (or even a few weeks from now) you may need to search for it again if it is not documented. More importantly, if someone else is trying to modify or run this project in the future, they would have no idea where these data came from or if the assumption regarding the equality of the "member\_id" column in wellness participation and claims data had been validated. For both of these reasons, the following note is added.



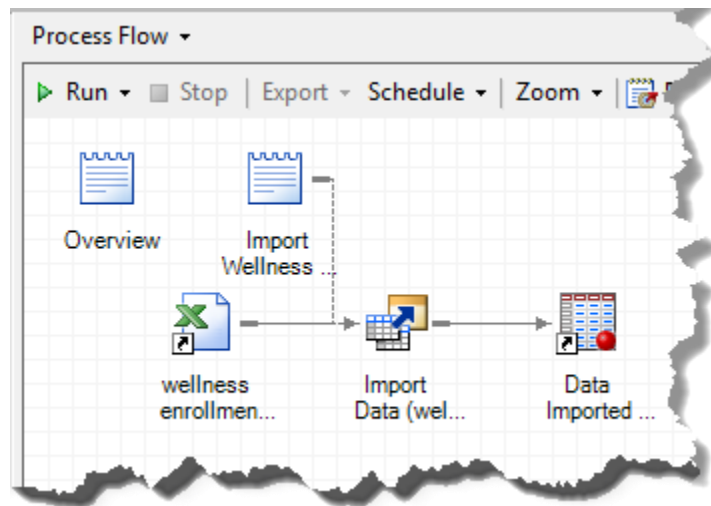
After the note is written, it can be linked directly to the Import Data task. First, right-click on the note and select "Link Import Wellness Data to...".



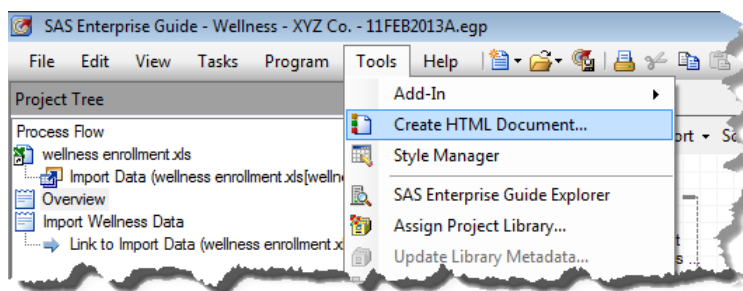
Then specify the process flow widget to which you want to link the note (in this case, the Import Data task).



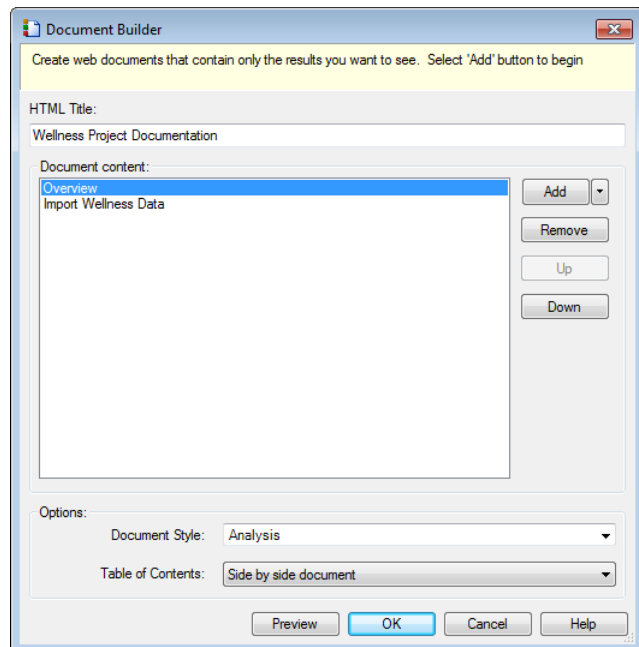
Once linked, the note is clearly associated with the Import Data task.



These notes can also be used to generate an HTML document that combines all of the notes in your project<sup>8</sup>. First, select **Tools ► Create HTML Document**.



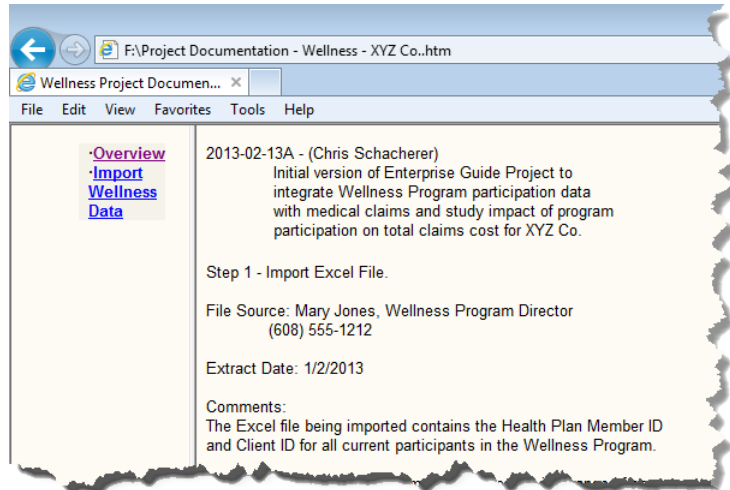
Next, in the Document Builder dialog, click "Add" to specify the type of items you want to add to the HTML document (in this case "Note"), and after selecting the items for inclusion, use the "Up" and "Down" buttons to arrange the notes in the order you want them to appear in the document. When you have arranged the notes in the correct order, click "OK" to generate the HTML document.



<sup>8</sup> The reader is also referred to Hallahan & Atkinson (2006) for examples of adding analytic output to HTML documents using the "Create HTML Document" option.



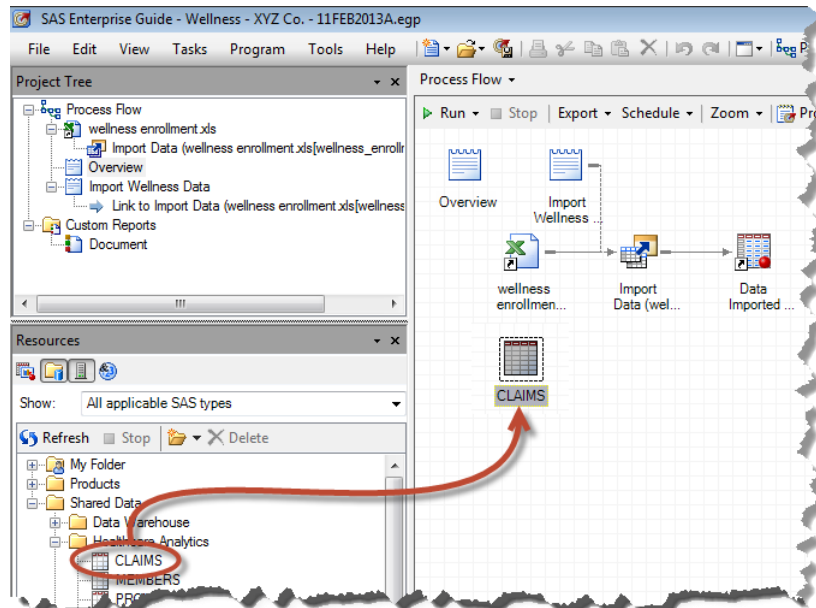
By default, the document will be output to a process flow in your project, but you can also choose to export the document from your project to an external HTML file.



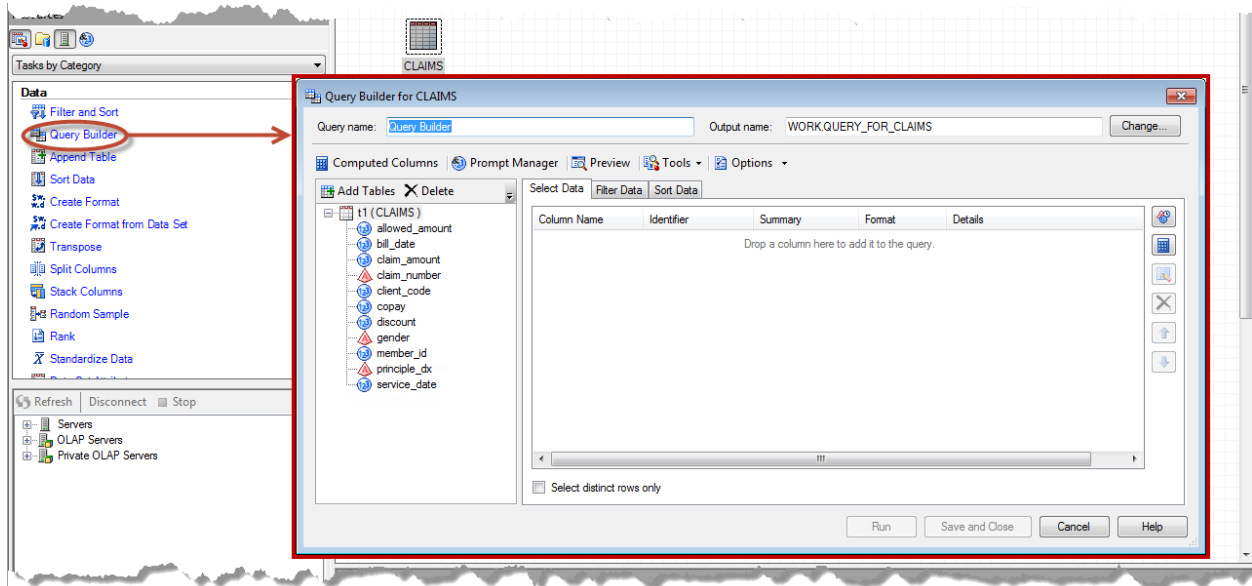
Now that the Wellness data has been imported and the project documentation has been started, the next step is to bring the claims data into the project so it can be joined with the Wellness Program participation data.

### JOINING SAS DATASETS, FUNCTIONS, AND FORMATS.

Joining two (or more) datasets is one of the most common data management tasks you are likely to perform in preparation for analysis and report generation in Enterprise Guide. In the current example, of course, what you need to do is join the healthcare claims data to the wellness participation data so that you can determine which claims are associated with a health plan member who participated in the wellness program and which claims are associated with a non-participant. As with importing the wellness data, to get the claims data into your project, you simply drag and drop the dataset onto the process flow. Unlike the wellness participation data, however, the "claims" dataset is a SAS dataset surfaced through the SAS Folder "Healthcare Analytics", so there is no task to execute to convert the data to SAS format; it is ready to manipulate, analyze, and report.

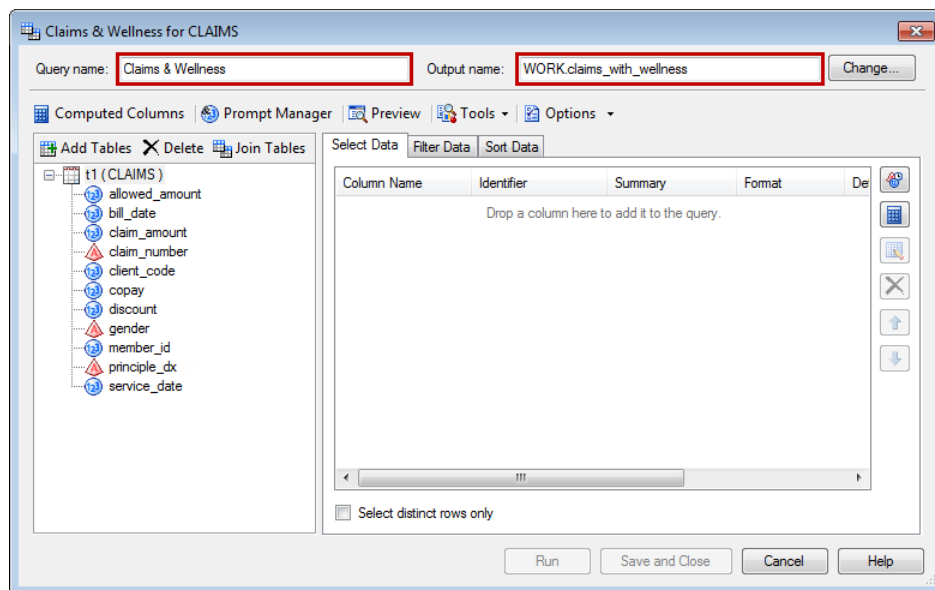


With the claims data in the process flow, a Query Builder task is added to the process flow by clicking "Query Builder" in the task list (or, by right-clicking the "claims" dataset in the process flow and choosing "Query Builder")<sup>9</sup>.



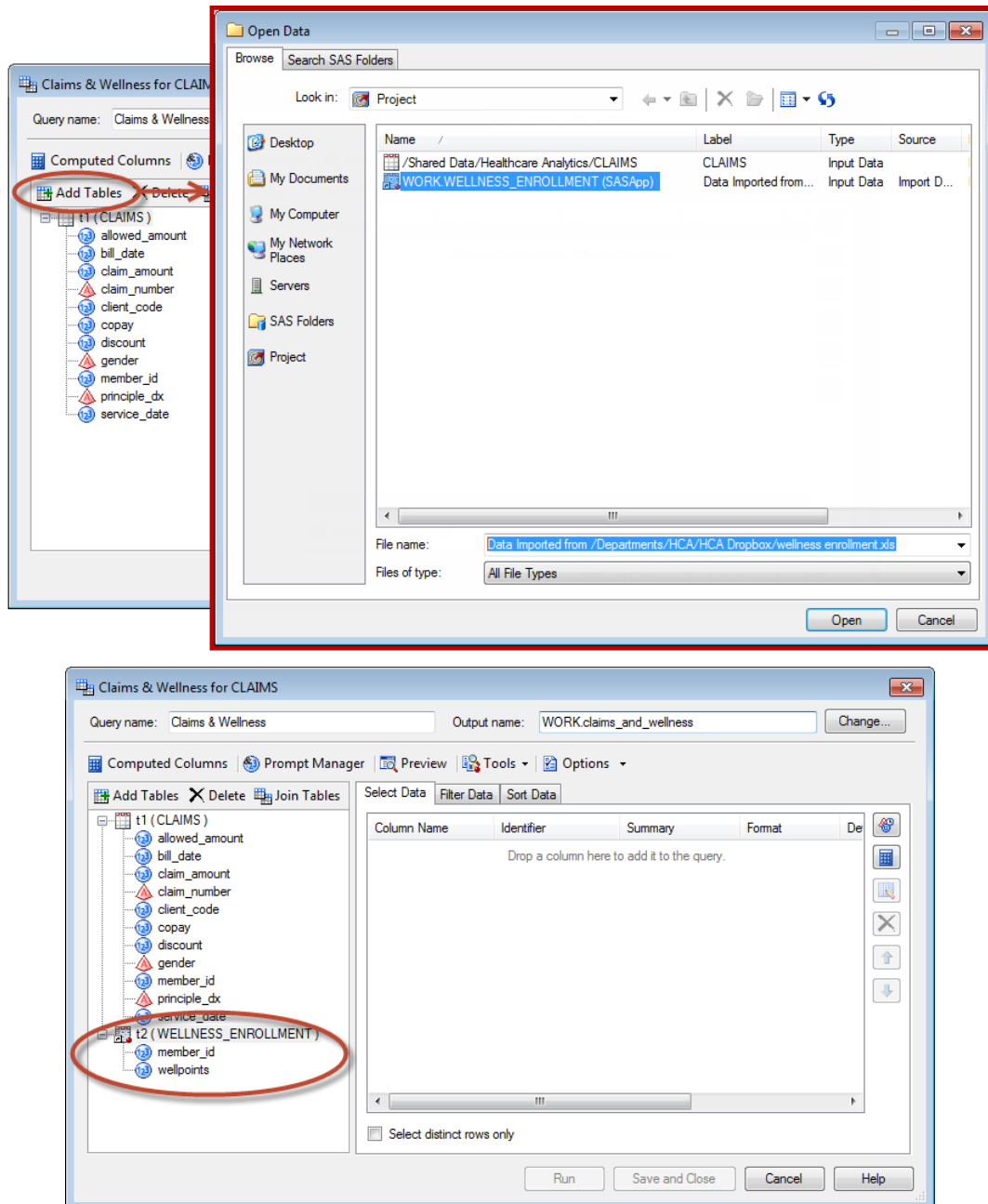
Unlike the Import Data task, the Query Builder task does not walk you through the associated process in a step-by-step (wizard) fashion, but instead presents you with a rich interface to enable a variety of data transformations. Remember, the job of an Enterprise Guide task is to gather from the user the information necessary to generate and execute the SAS code that will perform a data transformation, report, or analysis precisely as you intend.

As shown in the following figure, upon entry into the Query Builder task, you can assign the query a name that will be used to identify the task in the process flow and assign the name and location (SAS Library) of the output dataset.



<sup>9</sup> Interestingly, by default, when clicking on a task Enterprise Guide automatically assumes that you want to apply it to the last dataset highlighted. This is true even if a dataset is not currently selected. For example, if you currently have a note highlighted, but the dataset on which you last clicked was "claims", the task will start with the claims dataset.

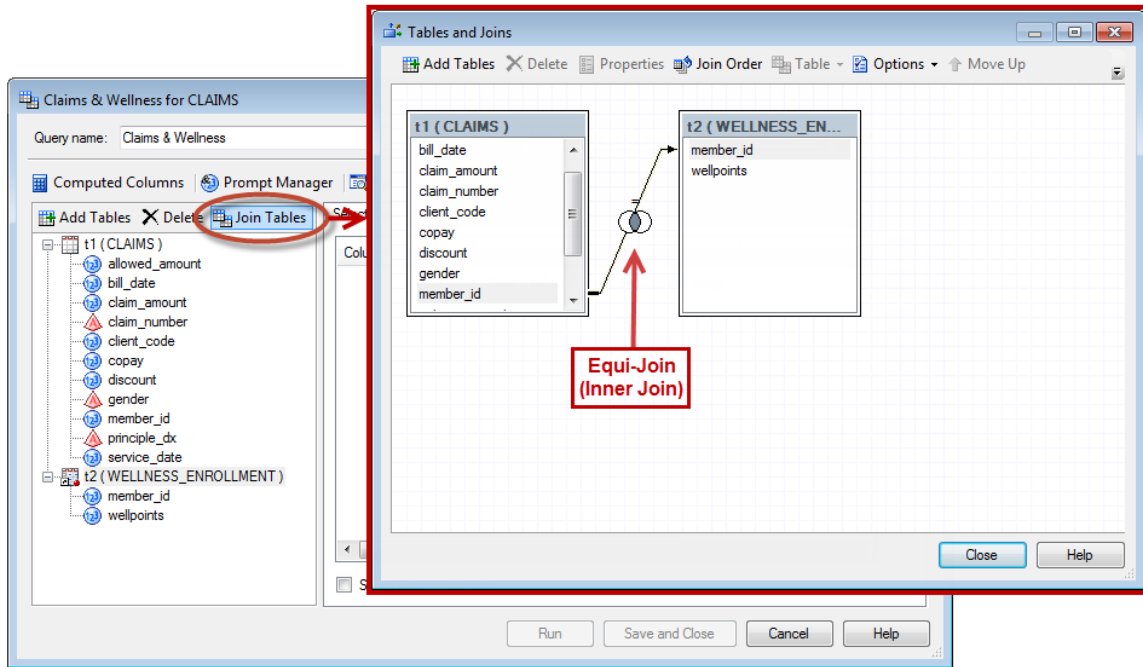
Next, the wellness data are added to the query by clicking on "Add Tables" and selecting the "wellness\_enrollment" dataset in the "Open Data" dialog.



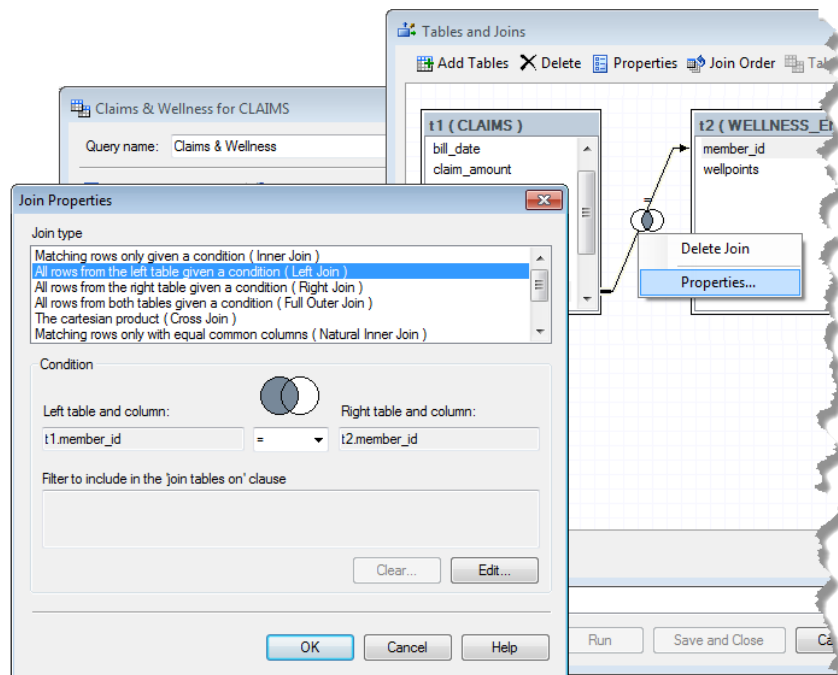
At this point SAS knows that you want to include records from both datasets in the output dataset. However, if you were to run the task now SAS would not know how you want to join data from the claims dataset to data from the wellness dataset. It seems obvious to you that you want claims records to be joined to wellness participation records based on the member\_id (that is, to match a claim record to the wellness record with the same member\_id), but SAS has no way of knowing that unless you somehow specify this condition. Without this specification, SAS will assume that you want to create every possible combination of matched records in the two datasets—a so-called "Cartesian product". For two datasets, each containing 1,000 records, the dataset resulting from such a join would contain 1,000,000 records (1000 x 1000, or, every possible combination of matched records from the two datasets).

In order to specify the condition by which the two datasets are to be joined, click on "Join Tables". By default, when adding a join, Query Builder will assume that you want to perform an equi-join—wherein the resulting dataset contains only records for which both of the datasets contribute data. In this case, the dataset resulting from such a

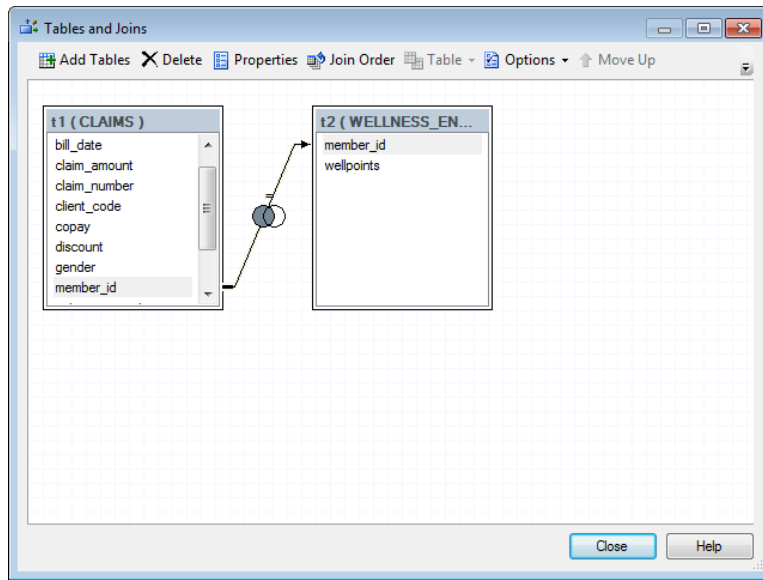
join would represent only those members with records in both the claims and wellness datasets—or, wellness plan participants who generated claims. Claims from non-participants would be excluded.



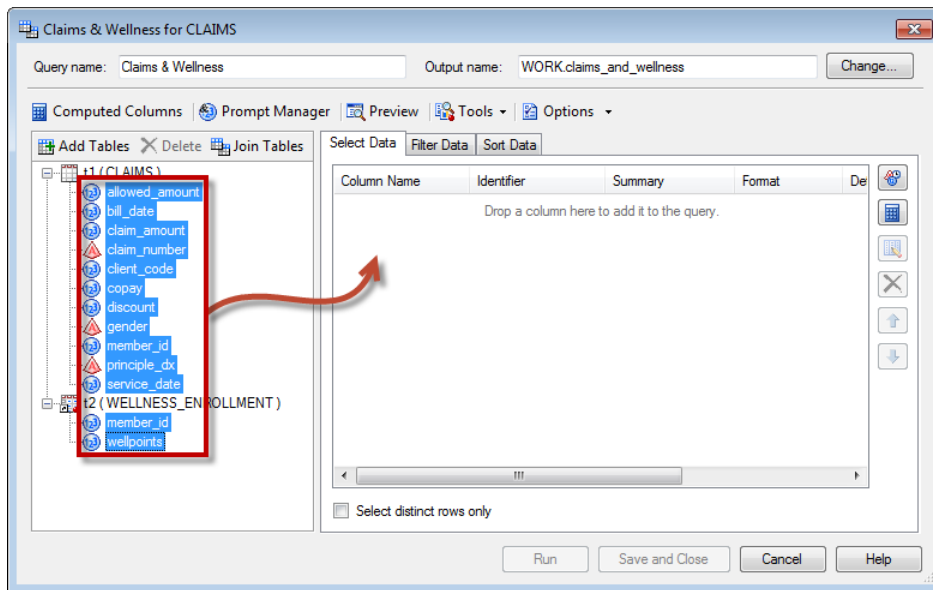
However, because the goal of this analysis is to compare those claims generated by health plan members who did not participate in the wellness program to those generated by members who did participate, a "LEFT-JOIN" will be specified to ensure that the resulting dataset contains all records from the claims dataset and only those records from the wellness dataset that have a matching record in the claims dataset. To change the join condition, right-click on the join and choose "Properties". Next, in the Join Properties dialog, select "Left Join" from the Join Types and click "OK"



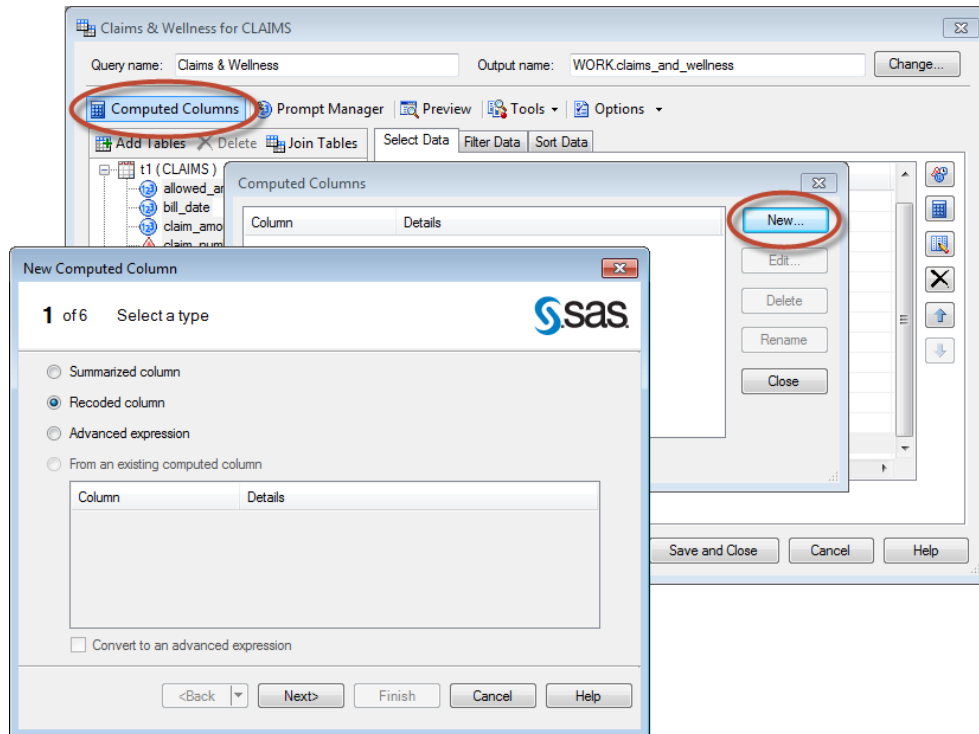
With the join type set to LEFT JOIN, close the Tables and Joins window.



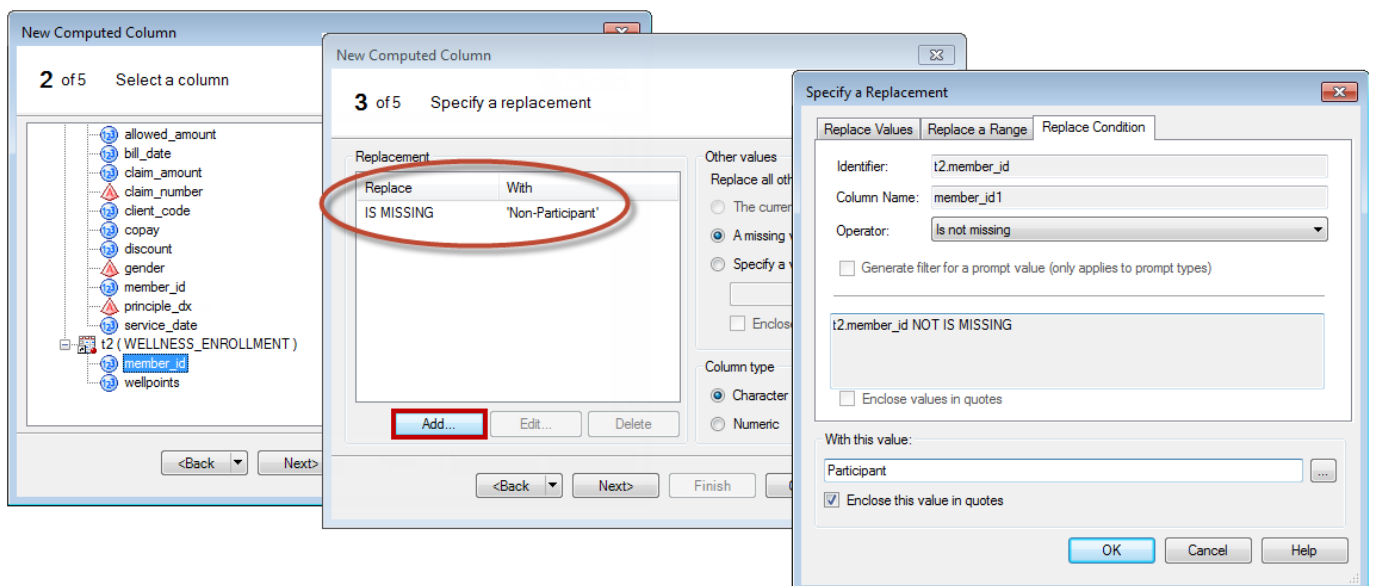
Next, the dataset variables to be included in the output dataset are selected from the list of available variables in the left pane and dragged to the "Select Variables" list.



In addition to variables available directly from the datasets, the Query Builder task also allows you to create "Computed" variables by summarizing, transforming, and recoding dataset variables. The first variable computed in this example is the "wellness" variable that will be used to indicate whether a given claim record is associated with a wellness program participant or non-participant. After clicking on "Compute Variables", you have a number of choices regarding the type of computed variable you wish to create. In this case, a "Recoded" variable will be created.

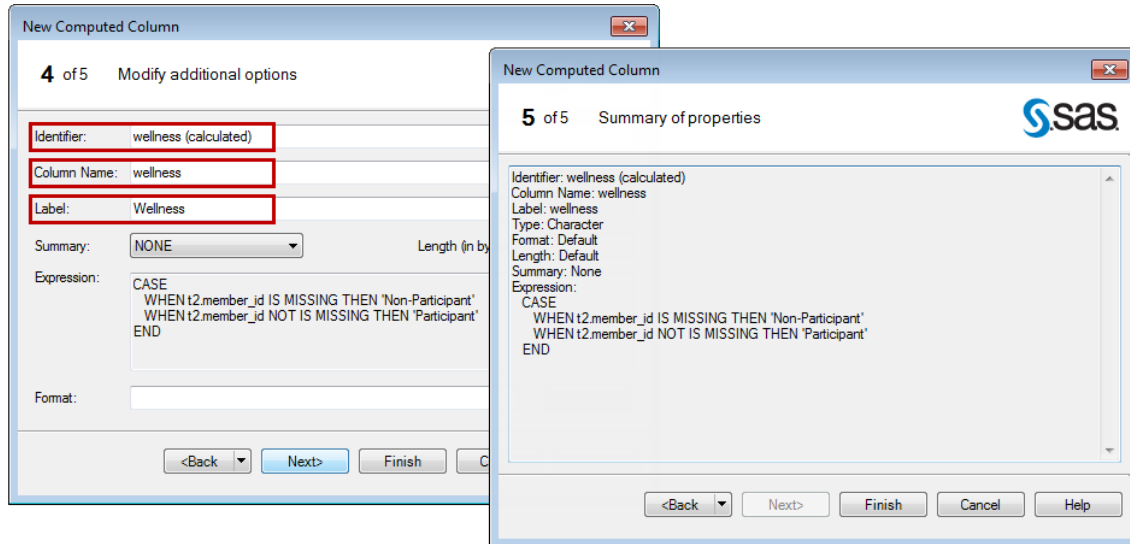


The member\_id from the wellness dataset will be selected as the basis for the recoding (Step 2). Remember that the join condition was specified such that all records from the claims dataset will contribute to the resulting dataset regardless of whether there is a matching member record in the wellness dataset. Therefore, in cases where a claim record is not associated with a member in the wellness dataset, the "member\_id" contributed from the wellness dataset will be "missing"—allowing you to recode missing values of this variable as "Non-Participant". Conversely, records in which the member\_id from the wellness dataset is not missing will be recoded as "Participant" in the new, computed variable.





After specifying the replacement conditions, you can specify an identifier for the new variable (to identify it within the Enterprise Guide task) as well as a column name to identify the variable in the resulting dataset and a label that will be used to identify the variable in analytic and reporting output. Finally, Step 5 of computing a new variable simply provides you an opportunity to review the information you have provided regarding the new variable computation. When you click "Finish", the definition for the new variable will be added to the Query Builder's list of available variables and will automatically be added to the list of variables selected for output to the resulting dataset.

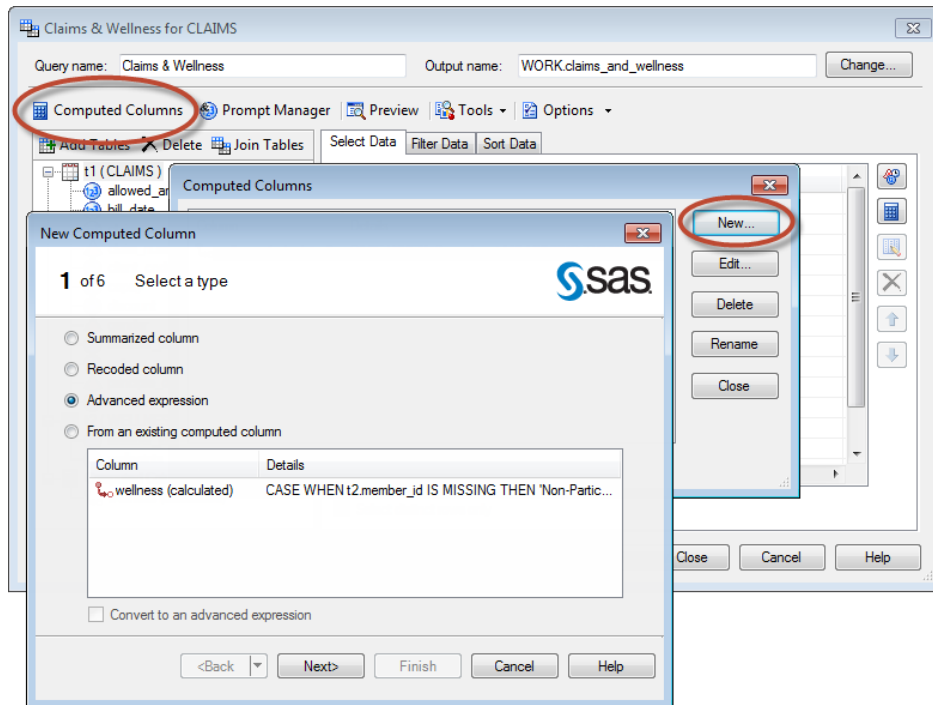


Before moving on to generating the dataset with this new "wellness" variable, however, there are two other variables that need to be created. As is sometimes the case in real-world datasets, the current claims dataset has some unfortunate data aberrations. One of these is the fact that the "claim number" variable contains both the claim number and an indicator of the claim status (e.g., "PAID", "PEND", "BAD").

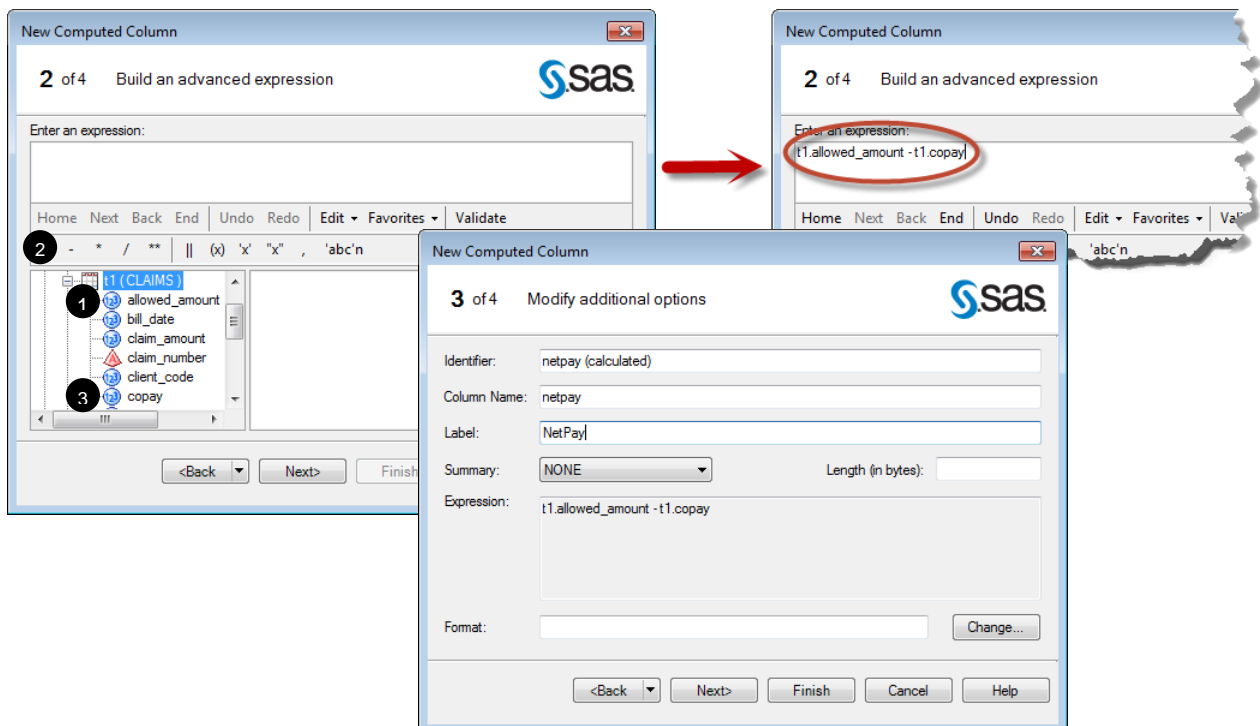
	client_code	member_id	gender	claim_number	service_date	bill_date	claim_amount	allowed_amount	copay
1	25	3200016226	F	33133456010PAID	18658	18658	852.24	162.31	0.0
2	25	3200013780	F	33133456011PAID	18628	18666	840	155.02	0.0
3	25	3200002173	F	33133456012BAD	18628	18666	840	155.02	0.0
4	24	3200019135	M	33133456013PAID	18629	18667	810.13	810.13	0.0
5	26	3200008703	M	33133456014BAD	18629	18633	10.7	10.7	0.0
6	25	3200028019	M	33133456015PAID	18631	18633	7.49	7.49	0.0
7	25	3200027050	M	33133456016PAID	18630	18633	12.62	12.62	0.0
8	24	3200024688	M	33133456017PAID	18629	18633	21.21	21.21	0.0
9	26	3200031254	M	33133456018PAID	18630	18633	12.62	12.62	0.0
10	24	3200002041	M	33133456019BAD	18630	18633	38.43	38.43	0.0
11	24	3200021691	F	33133456020PAID	18631	18633	52.9	52.9	0.0

For some of the planned analyses, you may want to have the actual claim number as it exists in other datasets throughout your organization (e.g., for the purpose of being able to join these data to other claim-centric datasets). Similarly, for the purpose of the present example, suppose the valid analysis of these data depends on including only "PAID" claims in the final dataset. For these reasons, two additional computed variables (claim\_num and claim\_status) will be created using the Advanced Expression Builder<sup>10</sup>. Unlike the previous computed variable, creating these two new variables will require use of **SAS Functions**. As in the previous example, to compute a new variable, click "Computed Columns" ► "New"

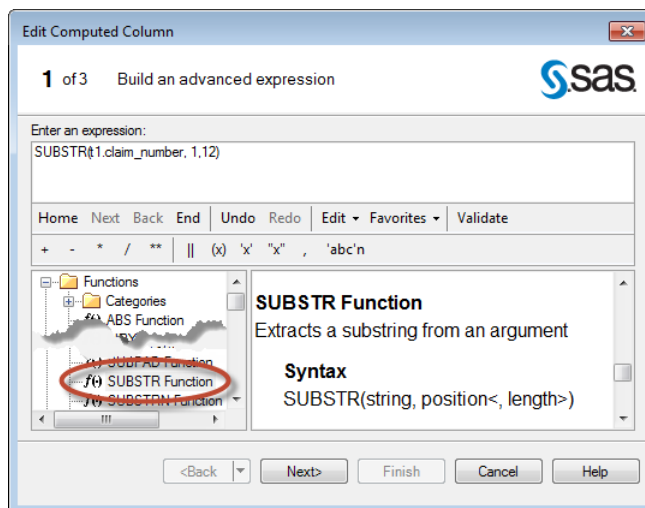
<sup>10</sup> It should also be noted that the Expression Builder is available in several Enterprise Guide tasks. It can also be used, for example, in the Filter and Sort task to create customized criterion for subsetting (i.e., filtering) a dataset.



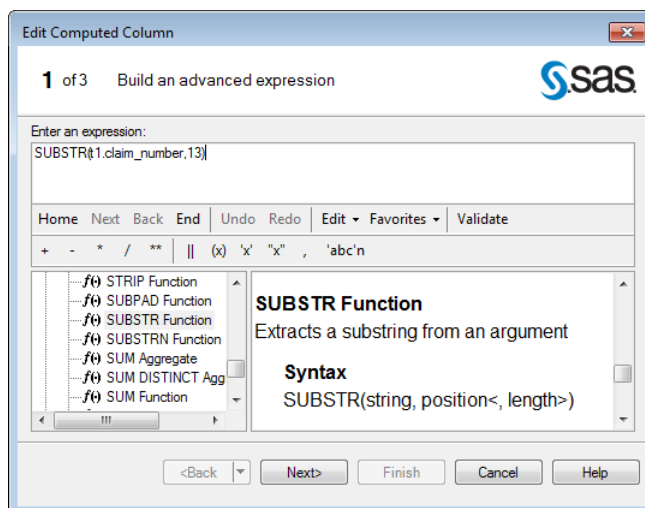
The Advanced Expression Builder allows you to write custom programming logic for the purpose of creating new variables in a dataset. This is not necessarily a skill that beginning Enterprise Guide users have to acquire to be immediately successful, but you should be aware that the capability exists to create complex calculated variables. As you become more comfortable with Enterprise Guide, it is likely that you will discover many ways in which you can add value to your projects through creation of such customized variables. As a quick demonstration of the basic functionality of the Expression Builder, you could create a "netpay" variable by subtracting the "copay" amount from the "allowed\_amount". Within the Advanced Expression Builder, ① double-click on "allowed\_amount", ② click on the minus sign (or type " - " in the expression), and ③ double-click on copay. This simple algebraic expression will create a new variable in the dataset and compute its value for every record in the output dataset. Click "Next" to name the new variable in Step 3, and then click either "Next" or "Finish" to complete the computation of "netpay".



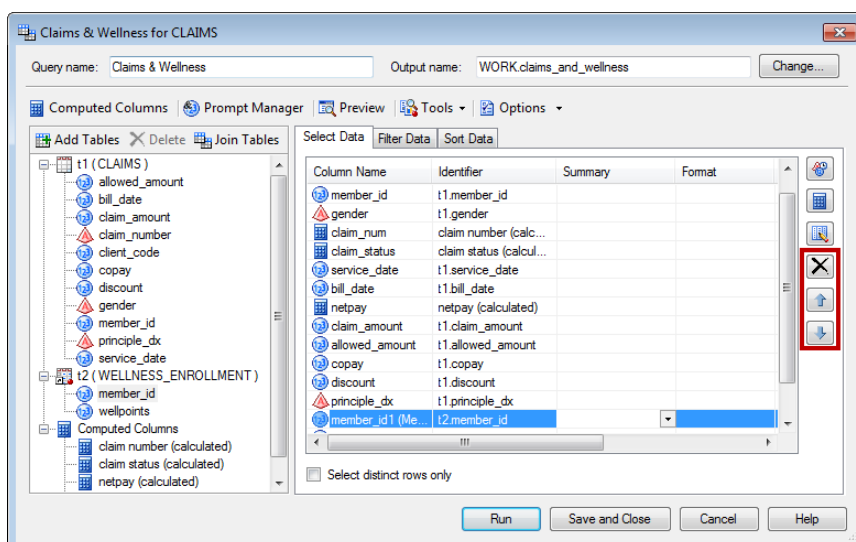
In order to transform the ill-formed "claim\_number" variable into acceptable "claim\_num" and "claim status" variables, the expression written in the Advanced Expression Builder will utilize a **SAS Function**. SAS Functions are specialized program units that take 0, 1, or more input parameters and return a single, distinct, predictable output based on those inputs. SAS Functions are generally categorized by the types of data on which they act (e.g., character, numeric, date). In the following example, the character function SUBSTRING (SUBSTR) is used to parse the text string found in the "claim\_number" variable to produce the new "claim\_num" variable. Specifically, this application of the SUBSTR function will read the contents of "claim\_number" from the beginning of the value (i.e., position 1) and read the contents for 12 characters—returning just the "claim number" portion of the existing variable. The result returned by the substring function is assigned as the value of the new "claim\_num" variable. Note also, that as the function is selected from the list of functions in the expression builder, a description of the function, its behavior, and appropriate syntax examples are presented in the right-hand pane. Again, you do not necessarily need to learn the behaviors of all the different SAS Functions in order to quickly become proficient in the use of Enterprise Guide, but as you become more experienced with the basic use of tasks, adding knowledge of SAS Functions can help you add more value to your projects.



Next, the variable computed variable "claim\_status" is calculated using the SUBSTR function—this time, substringing the claim number variable from the position after the claim number (i.e., beginning at position 13) through the end of the value.



With the computation of these variables complete, you now have all of the variables needed for the output dataset. You can use the controls on the right-hand side of the Select Data tab to rearrange variables by moving them up or down in the list of selected variables or to delete them from the resulting dataset altogether (e.g., the redundant "member\_id" variable from the wellness data).



If you ran this Query Builder task after arranging the variables in the desired order (and with no other changes to the query specification), you might generate an output dataset that looks like the following. This figure shows that the computed variables "claim\_num", "claim\_status", and "netpay" were all generated as expected. However, some of the variables may not have the appearance you expect. For example, although not a huge leap for consumers of your report to make, the variables that represent dollar figures (e.g., netpay, claim\_amount) are not identified as dollar amounts, but simply as numeric values. Perhaps more troubling, however, is the fact that the "service\_date" and "bill\_date" variables do not look like dates at all, but rather integers. This is because SAS stores dates as integers representing the number of days since January 1, 1960. So, for example, the SAS Date "0" is equivalent to 1/1/1960, "1" is equivalent to 1/2/1960, etc. This means that the service date "18658" is really "1/31/2011".

	client_code	member_id	gender	claim_num	claim_status	service_date	bill_date	netpay	claim_amount
1	25	3200016226	F	33133456010	PAID	18658	18658	162.31	852.24
2	25	3200013780	F	33133456011	PAID	18628	18666	155.02	840
3	25	3200002173	F	33133456012	BAD	18628	18666	155.02	840
4	24	3200019135	M	33133456013	PAID	18629	18667	810.13	810.13
5	26	3200008703	M	33133456014	BAD	18629	18633	10.7	10.7
6	25	3200028019	M	33133456015	PAID	18631	18633	7.49	7.49
7	25	320002705	M	33133456016	PAID	18630	18693	12.62	12.62

Especially for presentation of variables containing dates and dollar amounts, it is preferable for reports and other analytic output to provide more user-friendly presentation of their values. The solution is to apply **SAS Formats** to these variables. SAS Formats are instruction sets that tell SAS how to display data, and SAS provides a wide variety of these instruction sets (Formats) that can be applied to data such as Dates, Time, Currency, and other numeric values. To apply a more familiar date format to the "service\_date" variable, for example, first click on "Modify Task" in the document window containing the output dataset (or, right-click the Query Builder task in the process flow and choose "Modify <task name>"). Next, double-click the "service\_date" variable in the Select Data list and change the format to the Date format MMDDYY10. It is important to understand that even though the appearance of the date will change in the output dataset once the format is applied, the values are still stored as integers in the dataset. By applying the format, you have only changed the way the values are presented.

The screenshot shows the SAS Query Builder interface for a task named "Claims & Wellness for CLAIMS". The "Computed Columns" list includes variables like client\_code, member\_id, gender, claim\_num, claim\_status, service\_date, bill\_date, netpay, claim\_amount, allowed\_amount, copay, discount, and principle\_dx. The "service\_date" variable is selected in the list.

Two dialog boxes are open:

- Properties for service\_date:** Shows the column name as "service\_date", label as "Service Date", and format as "None". A red box highlights the "Change..." button.
- Formats:** Shows a list of categories with "Date" selected. The "MMDDYY10" format is highlighted in the list. The "Attributes" section shows "Overall width" set to 10 and "Decimal places" set to 0. The "Example" section shows a value of 14245 (01Jan1999) and its output as 01/01/2011.

By similarly applying the MMDDYY10. format to "bill\_date" and the currency format "DOLLAR12.2" to the columns representing dollar amounts, the dataset begins to lend itself more readily to use in reports that need to convey this information in a clear, concise fashion.

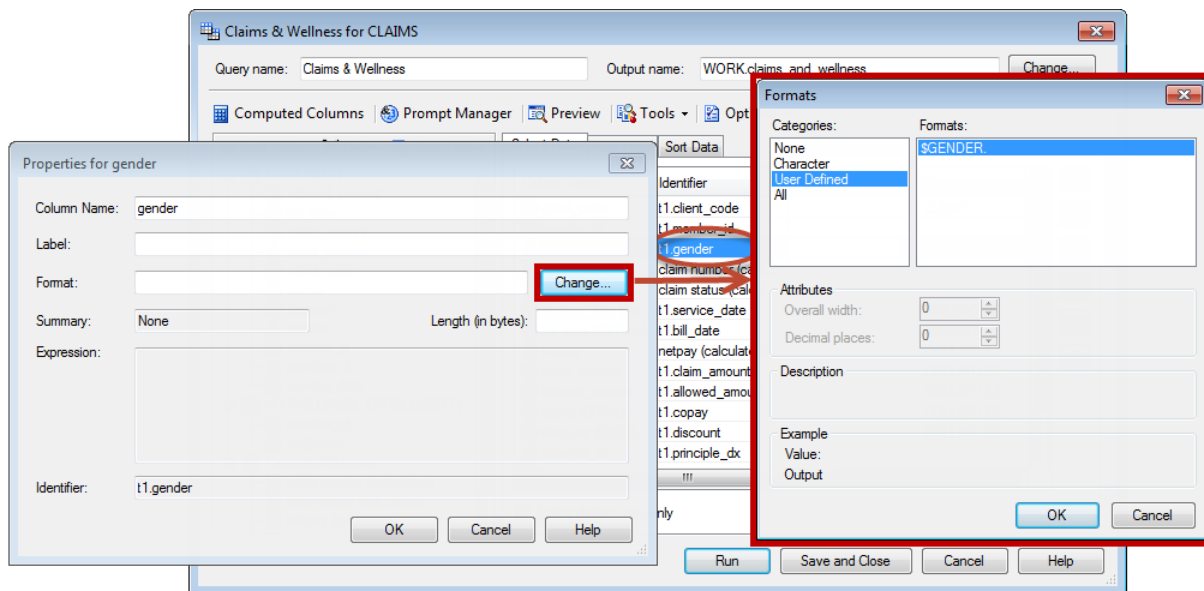
	client_code	member_id	gender	claim_num	claim_status	service_date	bill_date	netpay	claim_amount
1	25	3200016226	F	33133456010	PAID	01/31/2011	01/31/2011	\$162.31	\$852.24
2	25	3200013780	F	33133456011	PAID	01/01/2011	02/08/2011	\$155.02	\$840.00
3	25	3200002173	F	33133456012	BAD	01/01/2011	02/08/2011	\$155.02	\$840.00
4	24	3200019135	M	33133456013	PAID	01/02/2011	02/09/2011	\$810.13	\$810.13
5	26	3200008703	M	33133456014	BAD	01/02/2011	01/06/2011	\$10.70	\$10.70
6	25	3200028019	M	33133456015	PAID	01/04/2011	01/06/2011	\$7.49	\$7.49
7	25	3200027050	M	33133456016	PAID	01/03/2011	01/06/2011	\$12.62	\$12.62
8	24	3200024688	M	33133456017	PAID	01/02/2011	01/06/2011	\$21.21	\$21.21

In addition to these Formats supplied by SAS, Enterprise Guide also provides a facility to create your own **User-Defined Formats**. Again, because Formats are simply a set of instructions to specify how certain values should be displayed, all the Create Format task does is create the relationship between values encountered in the dataset and the label you would like to assign to that value (or range of values). In the following example, the format "\$GENDER." is created to assign the labels "Male" and "Female" to the values "M" and "F".

After launching the Create Format task from the Resource Pane, assign a name to the new format, and then define the format by adding the code/label pairs that form the rules for associating specific values with their corresponding labels. In this case, the name assigned to the format and the name of the dataset variable to which it will eventually be applied are the same, but this is just a coincidence. You could just as easily name the format "\$SEX.", "\$GROUP", or a host of other names.

The screenshot shows the 'Create Format' dialog box in SAS Enterprise Guide. The 'Format name' is 'GENDER' and the 'Format type' is 'Character'. The 'Define formats' section shows a table with 'Label' and 'Ranges' columns, containing 'Female' with range 'F' and 'Male' with range 'M'. The 'Label definition' section shows 'Label: Male' and 'Type: Discrete' with 'Values: M'. The 'How to define a format' section provides instructions on how to define a format.

After executing the task by clicking "Run", the "\$GENDER." format is now available to be assigned to the "gender" column in the dataset. As in the previous example, you can again modify the Query Builder task used to create "claims\_and\_wellness" dataset, double-click the gender variable in the Select Data list, navigate to the "User Defined" formats and select "\$GENDER."




Again, the data stored in the dataset are still the single-character values ('M' & 'F'), but when displayed in the dataset or in your SAS output, what will be displayed are the gender labels assigned by the format<sup>11</sup>. In the following example, note that in addition to the character format "\$GENDER.", the dataset also assigns a numeric format "CLIENT." to the variable "client\_code"—labeling the values "24, 25, & 26" as "ABC, Inc., XYZ Co., and 123, LLC", respectively.

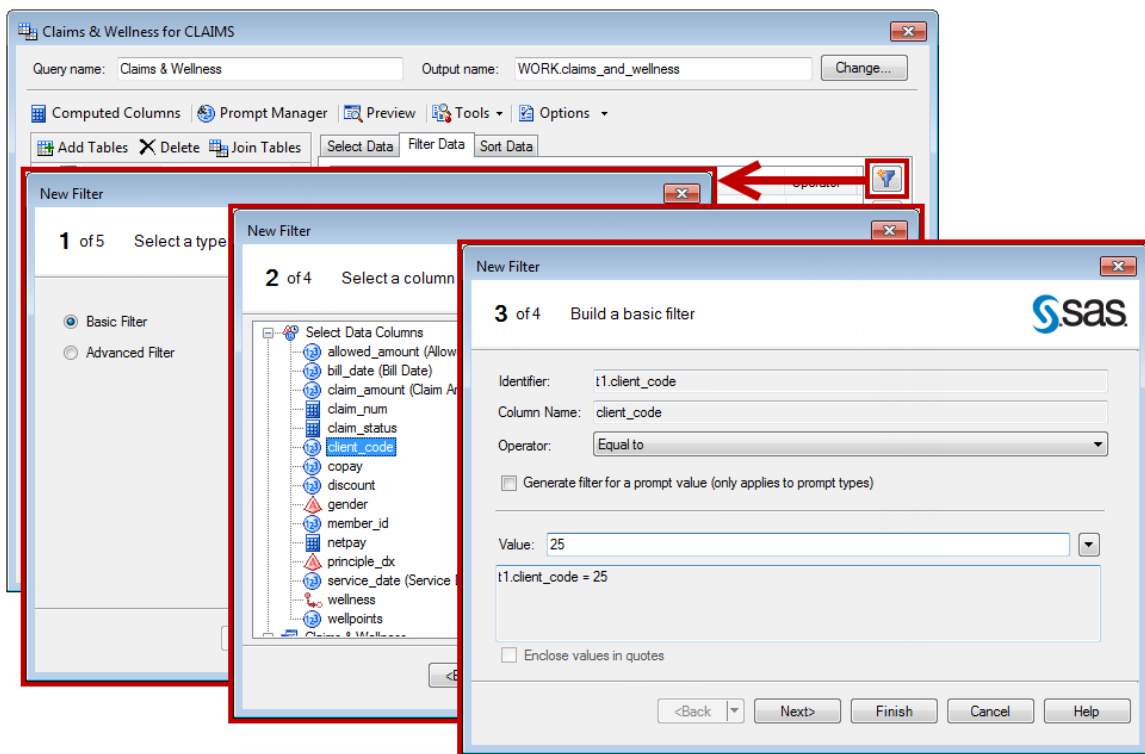
	client_code	member_id	gender	claim_num	claim_status	service_date	bill_date	netpay	claim_amount
1	XYZ Co.	3200016226	Female	33133456010	PAID	01/31/2011	01/31/2011	\$162.31	\$852.24
2	XYZ Co.	3200013780	Female	33133456011	PAID	01/01/2011	02/08/2011	\$155.02	\$840.00
3	XYZ Co.	3200002173	Female	33133456012	BAD	01/01/2011	02/08/2011	\$155.02	\$840.00
4	ABC, Inc.	3200019135	Male	33133456013	PAID	01/02/2011	02/09/2011	\$810.13	\$810.13
5	123, LLC	3200008703	Male	33133456014	BAD	01/02/2011	01/06/2011	\$10.70	\$10.70
6	XYZ Co.	3200028019	Male	33133456015	PAID	01/04/2011	01/06/2011	\$7.49	\$7.49
7	XYZ Co.	3200027050	Male	33133456016	PAID	01/03/2011	01/06/2011	\$12.62	\$12.62

With the columns of data selected, computed, and formatted for presentation, however, you still may need to specify which rows of data will be output by the Query Builder task. As mentioned previously, the analysis being performed is specific to your client "XYZ Co." They likely care less what the "overall" impact of wellness programs is across all of your clients than they do what the impact of their wellness program is on their bottom line. Therefore, the dataset will be reduced (or filtered) to include ONLY those rows of the claims dataset that represent claims generated by the employees of XYZ Co.

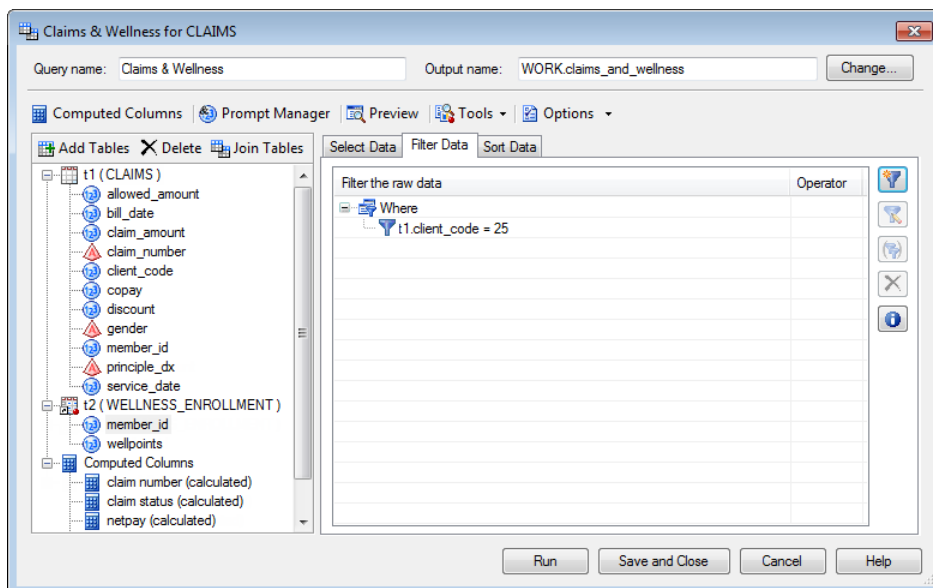
<sup>11</sup> If you work in a SAS BI environment, it is likely that someone has already created a number of user-defined formats based on the various look-up tables in the systems that source your analytic data—common among these are user-defined formats to provide consistent labels for Gender, Age-Group (Child/Teen/Adult/Senior), etc.



To perform this filtering operation, navigate to the Filters tab of the Query Builder interface, click on "New Filter" , and select "Basic Filter". Next, (Step 2) because we want to limit the dataset to claims from client company XYZ Co., select "client\_code" as the variable to evaluate. Then (in Step 3) select "Equal to" as the Operator, and the value 25 (corresponding to the value identifying XYZ Co.<sup>12</sup>) as the criterion.



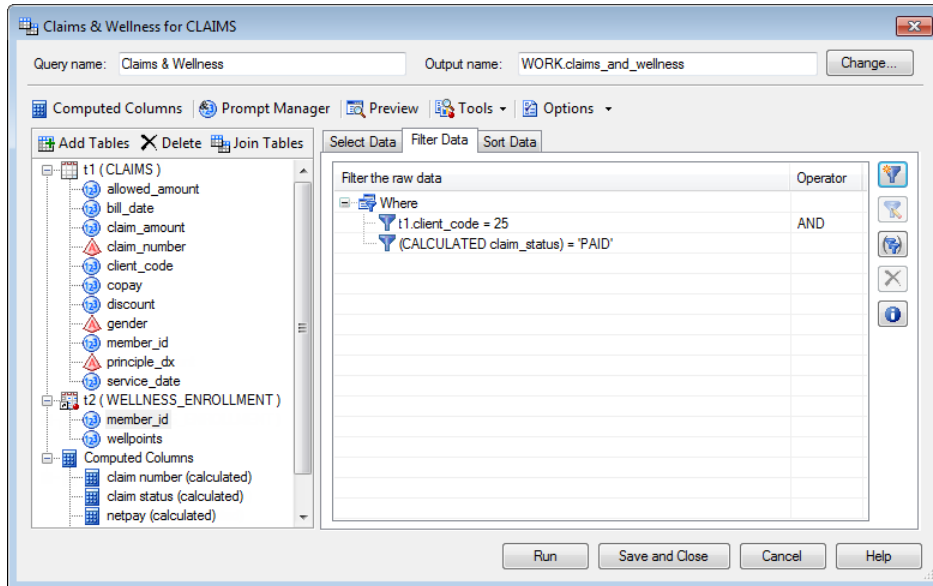
After clicking "Finish", the new filter is added to your Query Builder task and the output dataset will contain only those records with a client\_code of "25".



In addition to limiting the client company, let's assume that the dataset must also be limited to claims that have a status of "PAID". After creating another new Basic Filter, and specifying the "AND" operator to determine the

<sup>12</sup> Because we have not yet created the formatted dataset that would allow "25" to be displayed as "XYZ Co.", the numeric codes are the only available values that can be selected as the criteria.

combination of filters to apply to each record being evaluated for output, the subsetting conditions for the query are complete. A record must both have a client\_code of "25" and a claim\_status of "PAID" in order to be written to the output dataset "claims\_and\_wellness".

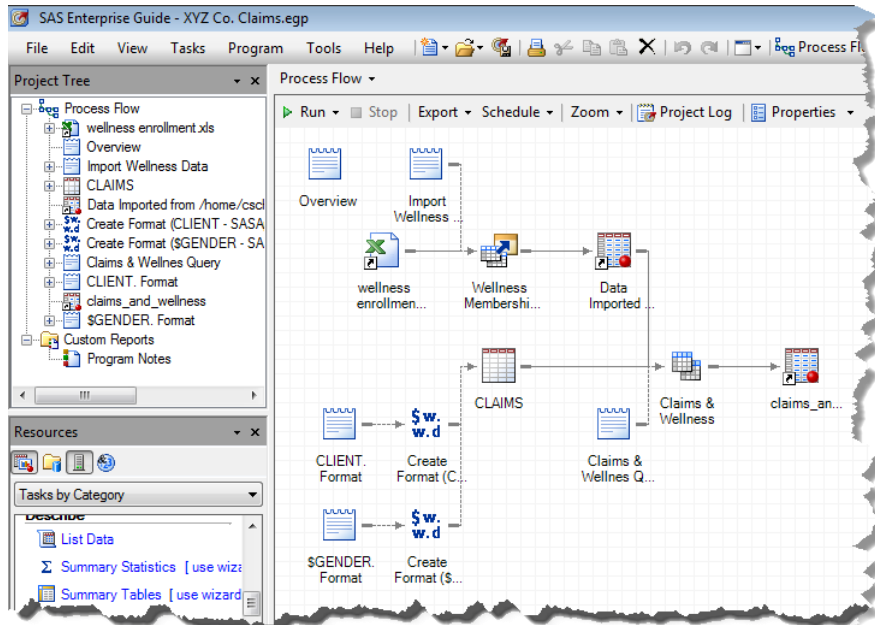


At this point, you can click "Run" to produce a dataset with all of the variables necessary for your analysis calculated and formatted for presentation.

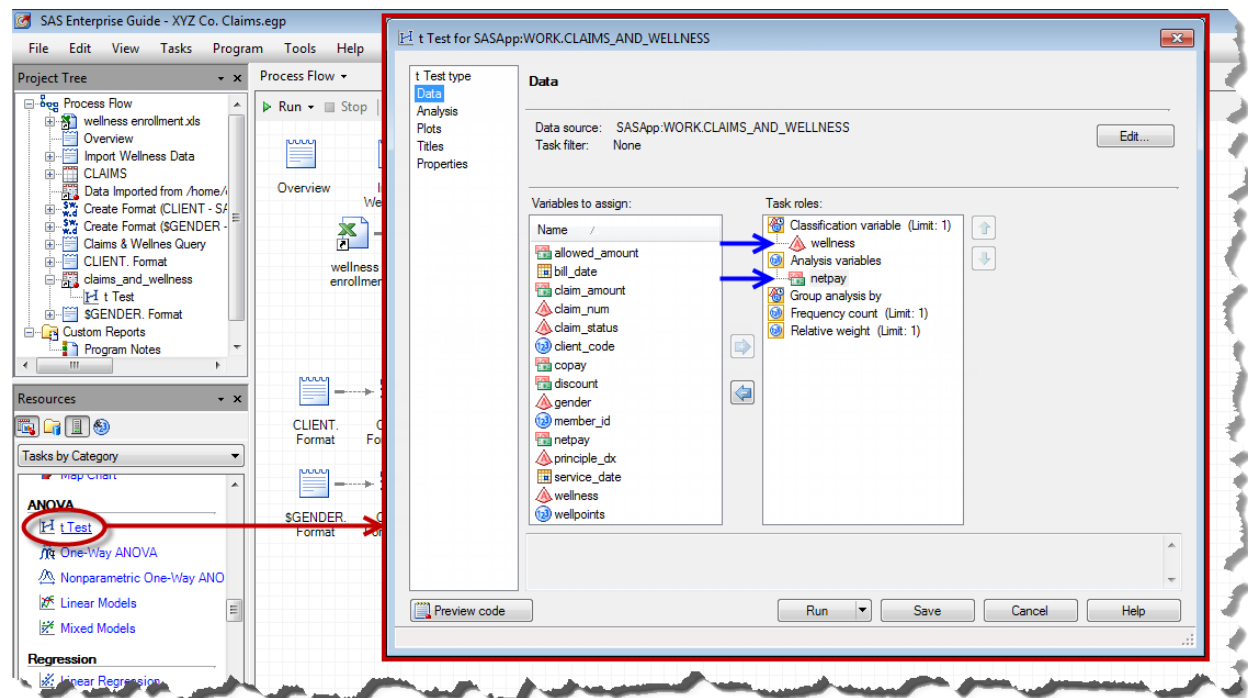
	client_code	member_id	gender	claim_num	claim_status	service_date	bill_date	netpay	claim_amount
1	XYZ Co.	3200016226	Female	33133456010	PAID	01/31/2011	01/31/2011	\$162.31	\$852.24
2	XYZ Co.	3200013780	Female	33133456011	PAID	01/01/2011	02/08/2011	\$155.02	\$840.00
3	XYZ Co.	3200028019	Male	33133456015	PAID	01/04/2011	01/06/2011	\$155.02	\$840.00
4	XYZ Co.	3200027050	Male	33133456016	PAID	01/03/2011	01/06/2011	\$7.49	\$7.49
5	XYZ Co.	3200016316	Female	33133456021	PAID	01/01/2011	01/04/2011	\$12.62	\$12.62
6	XYZ Co.	3200024552	Female	33133456027	PAID	01/01/2011	01/07/2011	\$21.21	\$21.21
7	XYZ Co.	3200020884	Female	33133456028	PAID	01/01/2011	01/07/2011	\$12.62	\$12.62

## PRODUCING GRAPHS WITH ENTERPRISE GUIDE.

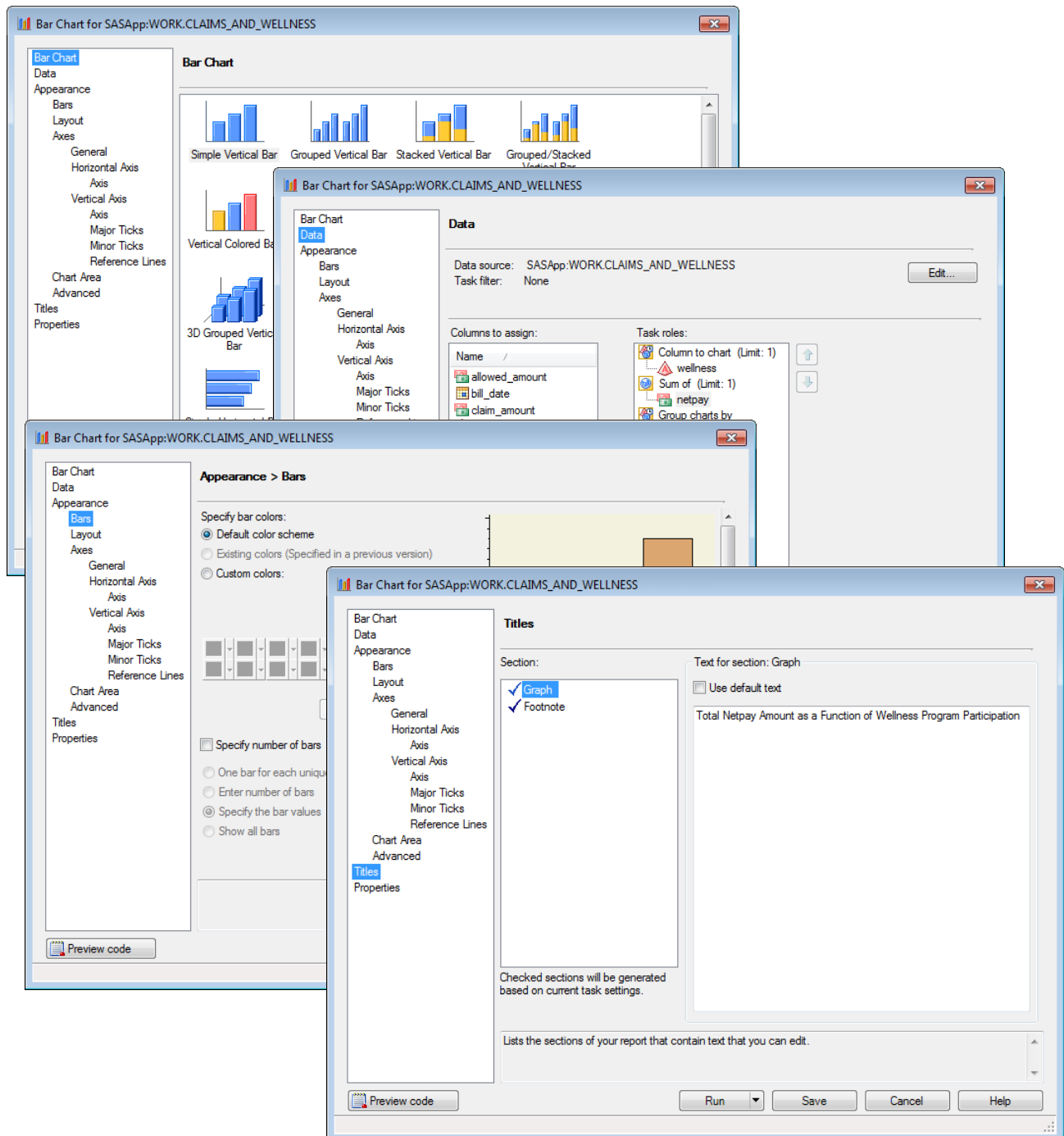
At this point in the project you have imported the Wellness Program enrollment data from an Excel file, dragged SAS dataset "claims" into the project from the "Healthcare Analytics" SAS Folder, joined the datasets together with the Query Builder task, computed new variables for use in your analysis and even created custom formats to make the data a bit easier to understand in your analytic output and reports. You have a SAS dataset containing precisely the rows and columns needed for your analysis.

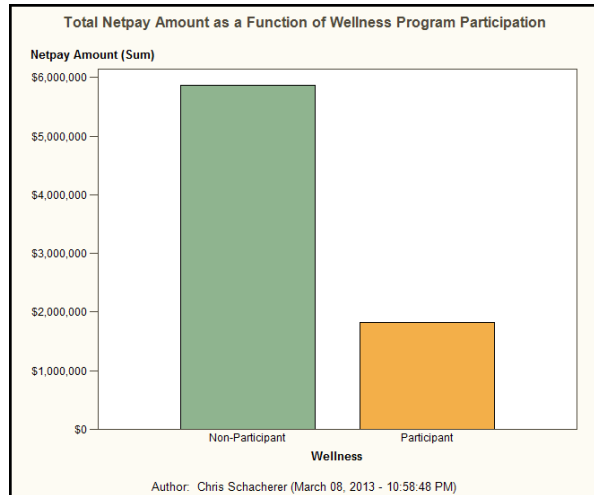


Next, you might want to test the statistical significance of the difference in mean payment amounts for claims generated by Wellness Program participants and non-participants. Like the other steps in the project, performing the analysis begins by bringing another task into the project. This time, the t-test task is selected, and by specifying the groups you want to compare (wellness participants and non-participants) and the quantitative variable on which to compare them (netpay), you can perform the statistical test comparing the mean netpay values.

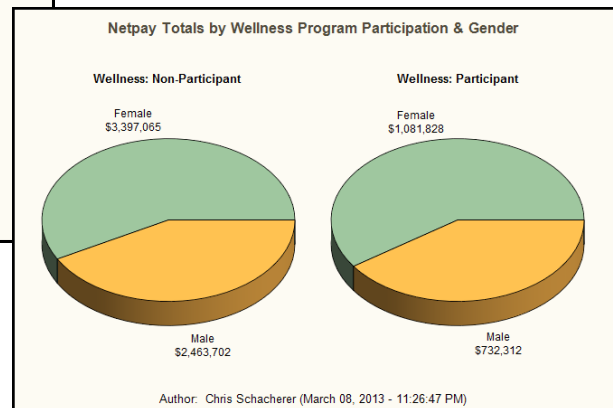
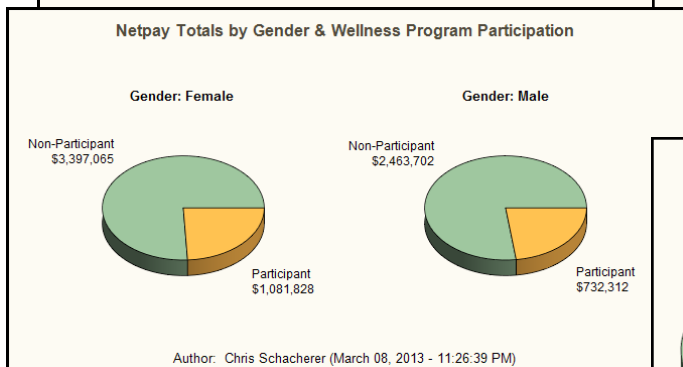
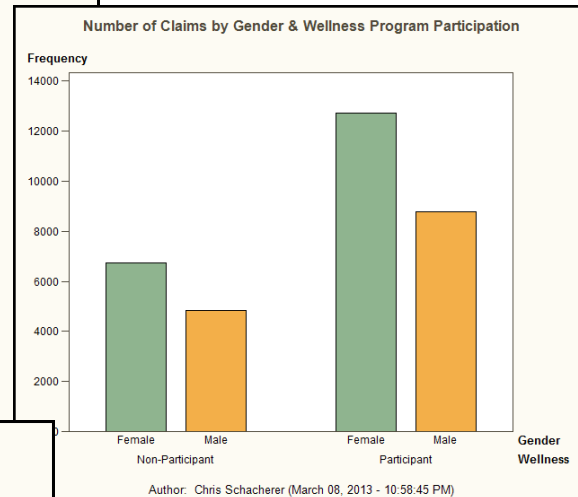
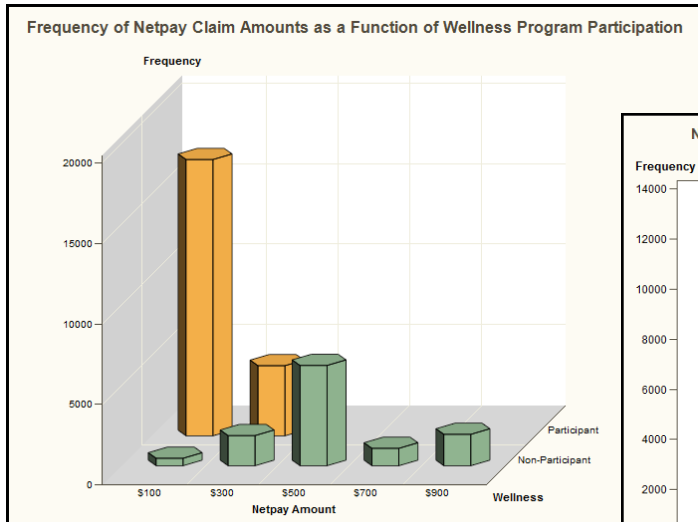


Depending on the type of analysis you are conducting and the audience to which it is presented, however, you might want to also provide some charts and graphs to help visually demonstrate the impacts of the Wellness Program. For new SAS users just starting out with Enterprise Guide, the Graph tasks also represent a skill area in which you can quickly begin to produce output that rivals that created by more experienced SAS programmers. Except for SAS programmers that spend the majority of their time producing graphs and charts, creating print-ready charts and graphs is a typical weakness even for experienced SAS programmers. With Enterprise Guide's wide array of Graph tasks, however, anyone can quickly produce very polished charts and graphs. In the Wellness Program analysis, one of the graphs you might want to generate is a bar chart comparing the total dollars spent on claims generated by Participants and Non-Participants. To do this, just click the Bar Chart task and, again, walk through the specification of the options for the task—including, the bar chart type, the variables to chart (in this case, the sum of "netpay" as a function of wellness category), the appearance of the bars, background, tick marks, etc., and title and footer text.

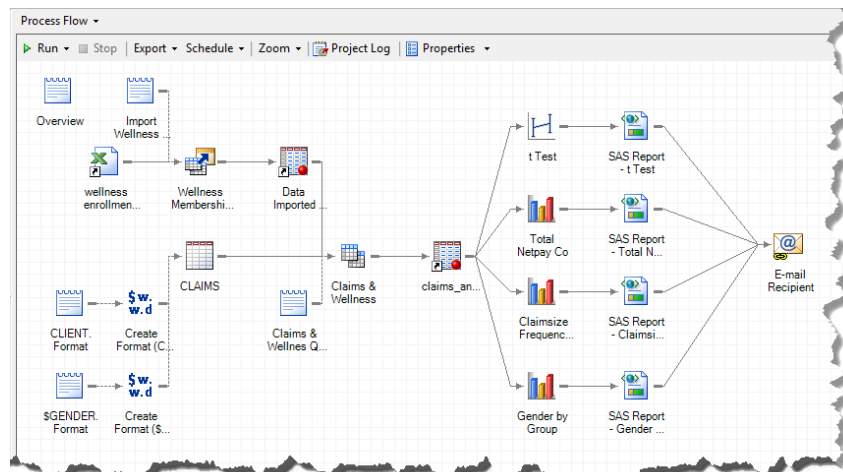




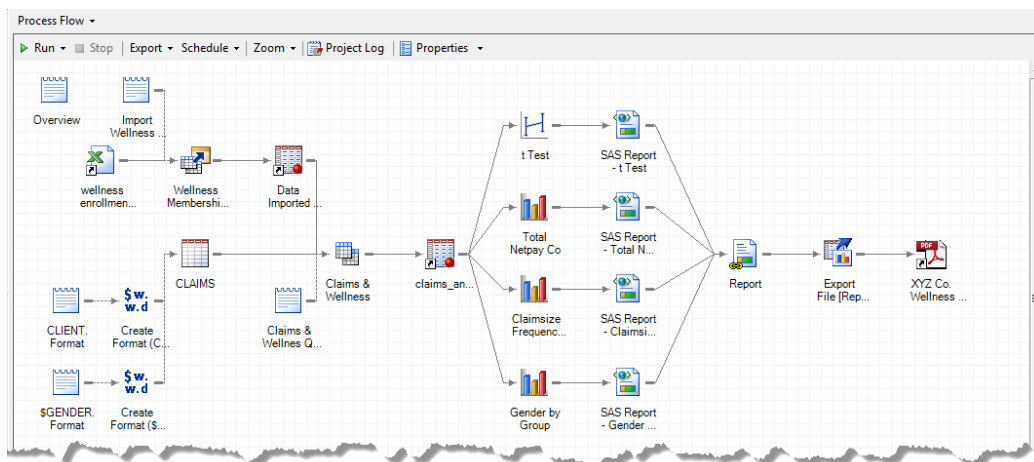
In addition to the Total Netpay x Program Participation bar chart, you can create additional views of these same data by repeating the steps with other Graph tasks or by specifying different combinations of options for the same task. In just a few seconds you could create all of the following views (and more) of these data.



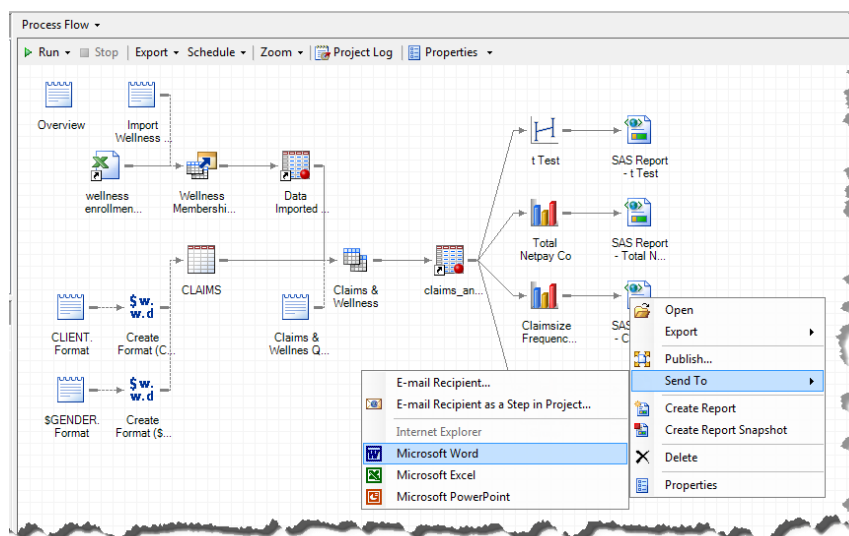
Having transformed the source data and produced the various analyses and charts, there are a number of next steps you could take to further enhance your project. You could attach the various reports as output to an e-mail (to yourself, your supervisor, etc.),



...put the individual pieces together in a single report file, format the layout of that report, add your own text to explain the report, and export the report to .PDF format,

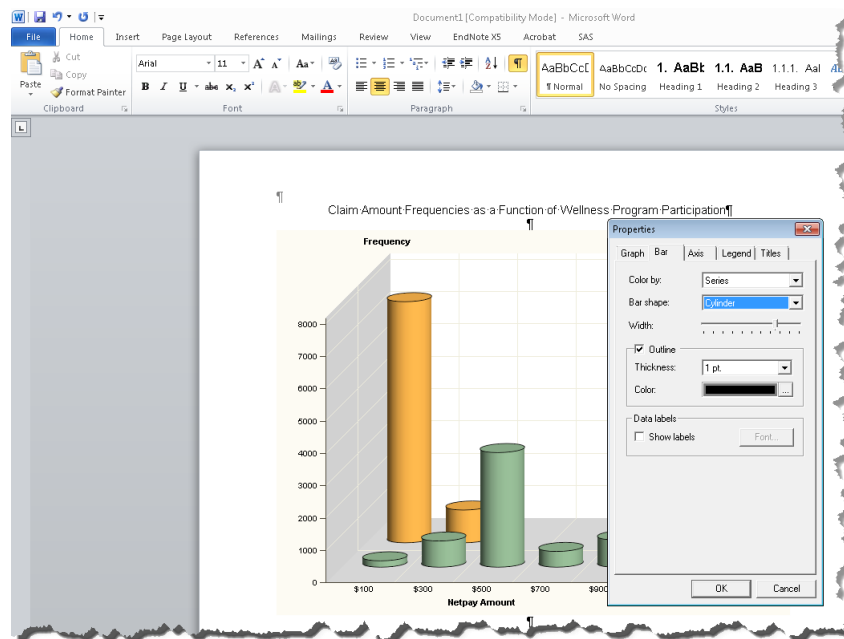


...or, sending the results to Microsoft Word, Excel, or PowerPoint documents.





Once sent to Microsoft Word, Excel, or PowerPoint via the SAS Add-In for Microsoft Office, the output is not just "pasted" as an image, but can be manipulated interactively to change the styles and colors of the graphics output.



Regardless of what you end up doing with the output of your project, however, one of the most important next steps you can take to add value to a project like the one in this example is to make it flexible enough to be repeated for any client without having to change any of the tasks you have already built. This goal can be achieved using the Prompt resource.

## CREATING DYNAMIC SOLUTIONS WITH ENTERPRISE GUIDE PROMPTS

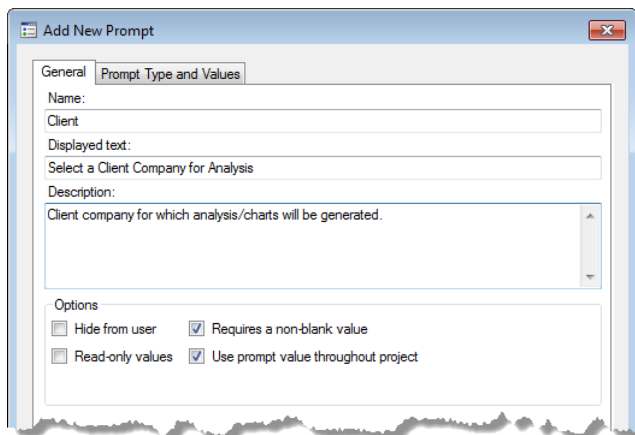
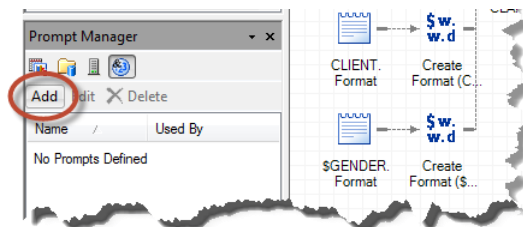
Finally, at the end of the analysis of the Wellness Program at XYZ Co., if the generated report(s) fulfilled the needs of your supervisor and the benefits organization at XYZ Co., it is likely that you will be asked to produce those same reports for other client organizations.

You could, of course, make a copy of the project for each client organization (e.g. "ABC, Inc.", "123, LLC", etc.) and change the filter criteria in your Query Builder task for each of these project files. However, as your projects become more complex—for example, with multiple queries from multiple data sources—this approach can get you in trouble. Suppose you have a project involving multiple queries against multiple data sources. If you specify a filter on each query to get data associated with a specific client, when changing that project to accommodate a request from a different client you will need to change each of those queries. If you accidentally fail to change one of the queries, the results will be invalid due to simple, avoidable human error. Also, if you take this approach and end up with dozens of Enterprise Guide projects that all do the same thing (each for a different customer) and then need to make a change to the project, you are potentially left in the position of having to make that change in each one of the customized copies of the project as well. This can quickly become a change-management nightmare.

Instead, you can create one version of the program that can dynamically produce the report for any one of your client companies.

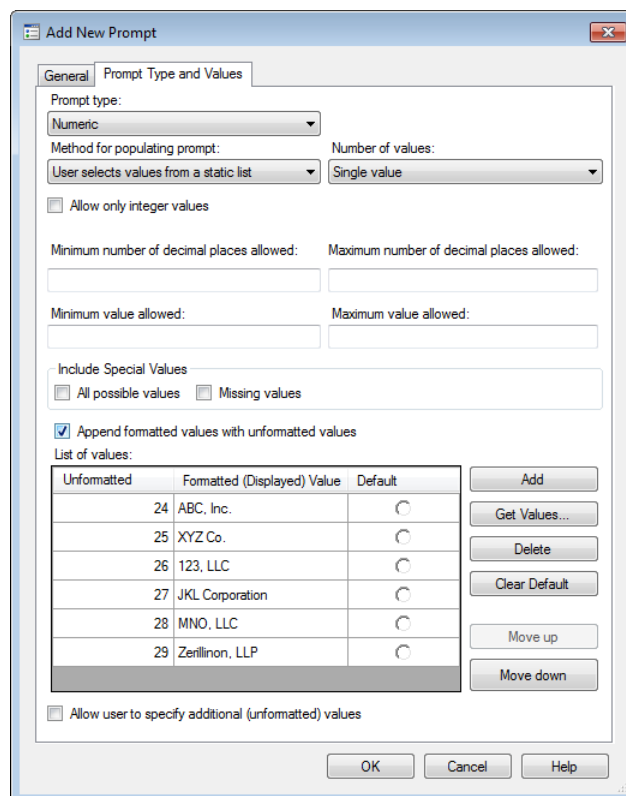
**PROMPTS.** In Enterprise Guide, the key to creating this flexibility is the **Prompt Resource**. As mentioned earlier, prompts facilitate user interaction with projects by allowing users to specify values that will impact the execution of the project. These prompt values can be used to dynamically specify the names and paths of files to import, user IDs and passwords for databases, criterion values in query filters, etc. In the Wellness Program analysis example, a prompt is added to the existing project to allow the user to specify the client company for which the analyses and reports will be generated.

The first step in rewriting this dynamic version of the project is to add the prompt "Client" by clicking on "Add" in the **Prompt Manager**.

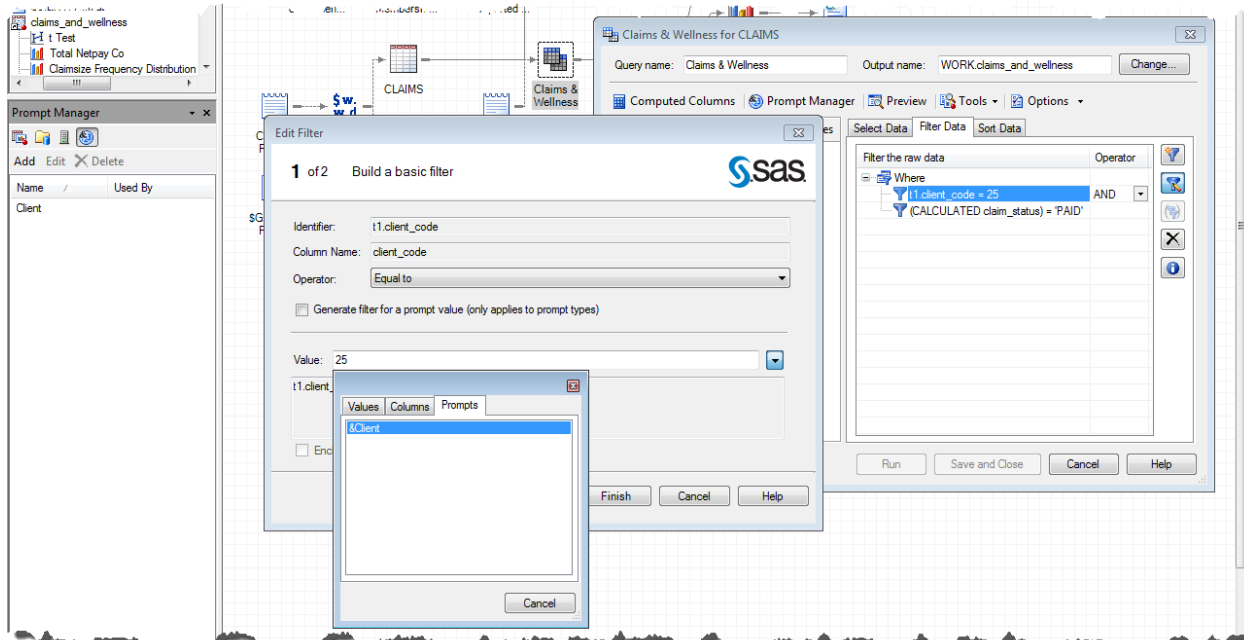


In the Add New Prompt dialog, give the new prompt a name, specify the text that will be displayed in the user interface when the project is executed, and provide an informative description of the prompt. As shown in the prompt options, this new "Client" prompt will be required to have a non-null value and will retain its value throughout the execution of the project.

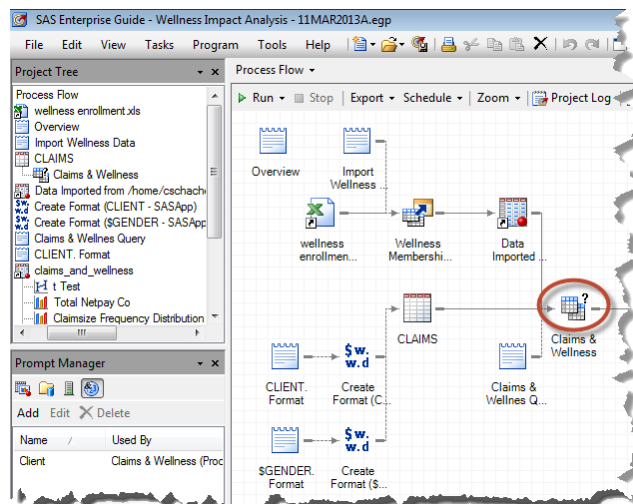
Because the "client\_code" variable used in the filter of the Query Builder task is numeric, the Prompt Type for "Client" is specified as numeric. You want users of this prompt to select their client company by choosing it from a drop-down list, so "User selects values from a static list" is selected under "Method for populating prompt". As indicated in "Number of Values", users of this prompt will be allowed to select a single value from the list presented to them—the individual client for which the reports will be generated. The values for this static list can be manually entered using the Add button next to the List of Values, or automatically populated from a data source using the Get Values button. Finally, when the list is displayed to the end-user, the Formatted Value of the list entry will be appended with the Unformatted Value (e.g., "ABC, Inc. [24]")—as this extra bit of information can be useful to users who are as familiar with the client code as they are with the Formatted (Displayed) Value. Once the prompt is built, click "OK" to save the prompt to the project, and you will see that it is now added as a prompt resource in your project.



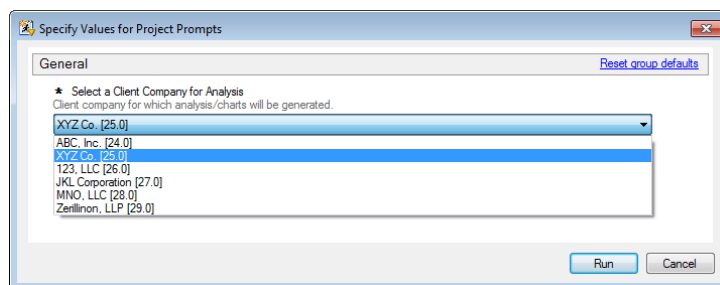
Next, the prompt needs to be associated with the filter (or WHERE clause) of the Query Builder task so that the client\_code selected in the "Client" prompt will be used to filter the dataset "claims\_and\_wellness". This association is made by right-clicking the Query Builder task in the process flow and choosing "Modify Claims & Wellness". Once in the Query Builder task, navigate to the "Filter Data" tab, select the existing filter and click on the "Edit" button. Instead of providing a specific client\_code (e.g., "25") as the value for this condition (as was done originally), navigate to the Prompts tab of the Value drop-down and select "&Client" to assign the prompt value as the evaluation criterion.



Once the query is changed to filter claims records based on the "Client" prompt instead of a specific, hard-coded value, the process flow is ready to be run in a manner that is driven by the user's response to the "Client" prompt. [Notice how the icon representing the query has changed to indicate that the task now involves a prompt.]



When the process flow is next run, the user will be presented with the new prompt. Once a prompt value is chosen by the user, clicking Run will execute the Query Builder task and produce the analysis and reports for the selected client company. To produce the output for a different company, all one needs to do is run the process flow again and choose a different value in the "Client" prompt.



## CONCLUSION

Creating reusable projects like the "Wellness Impact Analysis" is just one small example of how you can use Enterprise Guide to contribute to your organization. Hopefully this introduction has provided you with some insights into the types of functionality available to you as a new Enterprise Guide user. In addition to the relatively simple graphs and charts presented in the current work, Enterprise Guide affords you the ability to combine output into informative report documents, use conditional processing within the process flow to execute different branches of your project depending on the outcomes of prior tasks, specific summary values, or prompt selections, and even schedule projects to run unattended on a daily, weekly, or monthly basis. If you are a new Enterprise Guide user, the next step is to simply decide on an analysis or report you want to generate, and jump in to exploring the individual tasks, SAS Functions, and Enterprise Guide capabilities needed to access the data, transform it, and produce your desired output. With Enterprise Guide's user-friendly interface and the analytic capabilities of SAS software, it will not be long at all before you harness The Power to Know®.

## REFERENCES

- Aanderude, T. & Hall, A. (2012). Building Business Intelligence Using SAS®: Content Development Examples. Cary, NC: SAS Institute, Inc.
- Aster, R. & Seidman, R. (1997). Professional SAS Programming Secrets. McGraw-Hill.
- Bangi, A., Hemedinger, C., Slocum, S. (2010). New Goodies in SAS® Enterprise Guide® 4.3. Proceedings of SAS Global Forum 2010. Cary, NC: SAS Institute, Inc.
- Carpenter, A. (2012). Carpenter's Guide to Innovative SAS Techniques. Cary, NC: SAS Institute, Inc.
- Cody, R. (2007). Learning SAS® by Example: A Programmer's Guide. 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.
- Hemedinger, C. (2012). Review your metadata profiles using SAS Enterprise Guide automation. SAS Dummy Blog (February 2, 2012). <http://blogs.sas.com/content/sasdummys/2012/02/02/profiles-via-eg-automation/>.
- Lafler, K. (2004). PROC SQL: Beyond the Basics Using SAS®. Cary, NC: SAS Institute, Inc.
- Overton, S. (2012). SAS Enterprise Guide: Best Practice for Centralized User Storage. Retrieved February 16, 2013 from: <http://www.bi-notes.com/2012/06/best-practice-for-centralized-user-storage/>.
- Ravenna, A. (2011). Becoming a Better Programmer with SAS® Enterprise Guide®. Proceedings of SAS Global Forum 2011. Cary, NC: SAS Institute, Inc.
- SAS Institute, Inc. (2010). SAS Learning Edition>>SAS Procedures. Retrieved July 28, 2010 from: [http://support.sas.com/learn/le/proc/proc\\_2.html](http://support.sas.com/learn/le/proc/proc_2.html).
- SAS Institute, Inc. (2011). SAS® 9.3 Stored Process Developer's Guide. Cary, NC: SAS Institute Inc.
- Schacherer, C. (2012a). Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hocs to robust enterprise solutions. Proceedings of SAS Global Forum 2012. Cary, NC: SAS Institute, Inc.
- Schacherer, C. (2012b). The FILENAME Statement: Interacting with the world outside of SAS®. Proceedings of SAS Global Forum 2012. 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.
- Smith, C. (2012). Best Practices for Administering SAS® Enterprise Guide®. Proceedings of SAS Global Forum 2012. Cary, NC: SAS Institute, Inc.
- Sucher, K. (2010). Interactive and Efficient Macro Programming with Prompts in SAS® Enterprise Guide® 4.2. Proceedings of SAS Global Forum 2010. Cary, NC: SAS Institute, Inc.
- Whitlock, I. (2007). How to Think Through the SAS DATA Step. Proceedings of the SAS Global Forum 2007. 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  
E-mail: [CSchacherer@cdms-llc.com](mailto:CSchacherer@cdms-llc.com)  
Web: [www.cdms-llc.com](http://www.cdms-llc.com)

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.