Spring 2009 OASUS Q&A

The following answers are provided to the benefit of the OASUS Users Group and are not meant to replace SAS Technical Support. Also, an Enterprise Guide project is provided as a companion to this document. The project is available on the OASUS web site (www.oasus.ca) under the Fall 2009 meeting since this is when these answers have been presented formally.

Question 1 : How can we keep multiple versions of a dataset?

Often, one needs to maintains multiple versions of a given dataset for historical/archival as a well as backup/recovery purposes (there may be other reasons not as obvious). There are different ways this could be done in SAS. For example, one could program a custom solution in a SAS application that would enable versioning one or many datasets. This custom solution would certainly add some complexity to the application.

Another approach could be to use the SAS audit trail facility which logs all the modifications to a SAS data set during its lifecycle. It does not maintain multiple versions of a data set however.

SAS has yet another facility called the *Generation Data Sets*. A generation data set is an archived data set that is part of a generation data group. Under a generation data group, each time a data set is replaced by a new one under the same name, a copy of the old version is kept under a specific generation number. The most recent version is called the *base version*. A generation group is created when the dataset option *GENMAX* is specified with a number greater than 0 (0 is the default and means that the generation data sets feature is not in effect). This option specifies the maximum number of versions to keep. The following example creates a generation data group with a maximum number of versions of 4.

```
/* Dataset creation and first generation */
data testgen(genmax=4);
   input var1 $ var2 $;
   datalines;
a11 b11
a12 b12
;
```

Except for the current version, each historical version of a data set in generation data group is stored using an absolute version number appended to its member name. The absolute version starts with # and is followed with 3 digits. For example, for data set A:

A base (current) version
A#003 most recent historical version
A#002 second most recent historical version
A#001 oldest historical version.

When the maximum number of generations is reached, SAS will delete the oldest version in order to keep only the number copies as specified by the *GENMAX* option.

Once you have a generation group created, how do you reference the various data sets associated with it? First rule, except for the current version, never use the physical name

with the #. This will lead to a syntax error since data set names cannot contain a #! Instead use the *GENNUM* data set option as followed:

GENNUM=0 : to access the current version. This is the default, so you don't need to specify it.
GENNUM=-x : to access the version x generations back. This is relative reference.
GENNUM=x : to access version x. This is an absolute reference.

The following example illustrates the concept.

```
/* Print most recent version */
title 'most recent version';
proc print data=testgen;
run;

/* Print generation back from the current version */
title 'Version -2';
proc print data=testgen(gennum=-2);
run;

/* Print generation #003 */
title 'Version #003';
proc print data=testgen(gennum=3);
run;

/* This file reference is invalid */
/* A valid SAS file name cannot contain an # */
title 'Version #003';
proc print data=testgen#003;
run;
```

To manage generation groups, it is recommended to use *PROC DATASETS*. *PROC DATASETS* allows you to:

- Change the number of versions to be kept.
- Delete specific versions and an entire generation group.
- Rename versions

SAS generation data sets are a very handy and easy-to-use facility to keep multiple versions of a dataset. Keep in mind however that the storage requirements of your SAS application may increase quickly.

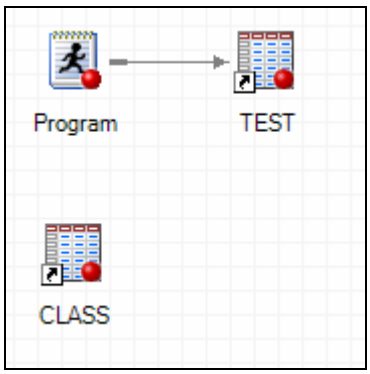
Reference :
SAS Global Forum Paper 253-31, Exploring SAS Generation Data Sets, Kirk Paul Lafler

Question 2 : How do you link code nodes in Enterprise Guide?

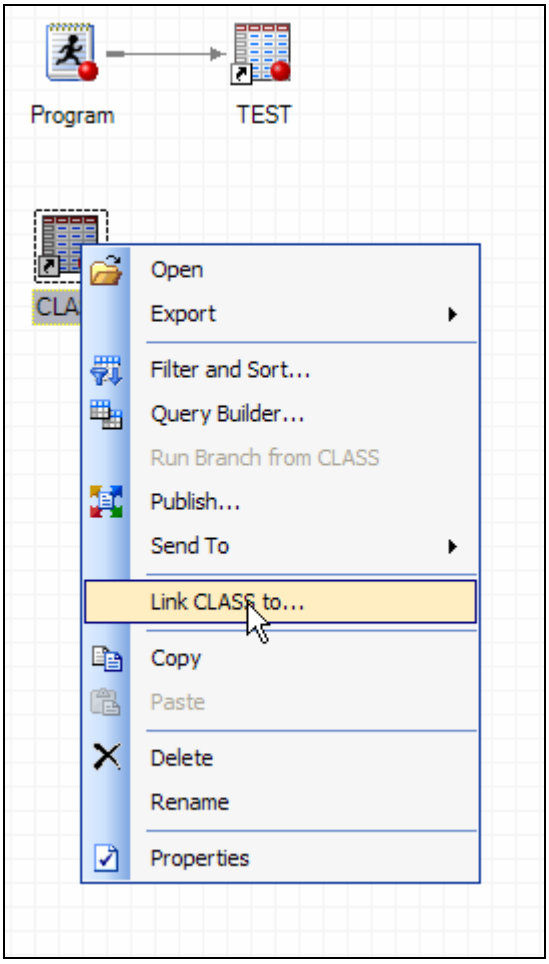
Most of the times, Enterprise Guide (EG) is good at connecting the different nodes in a process flow, especially when you use the Query Builder or one the many EG tasks. Sometimes however, EG simply does not connect some of the nodes. It is the case when you are writing your own code and you use an input SAS dataset. The input SAS dataset does get connected at all.

The good news is the EG lets you connect manually the nodes with the "Link to" facility. The following screens shots explain the operation.

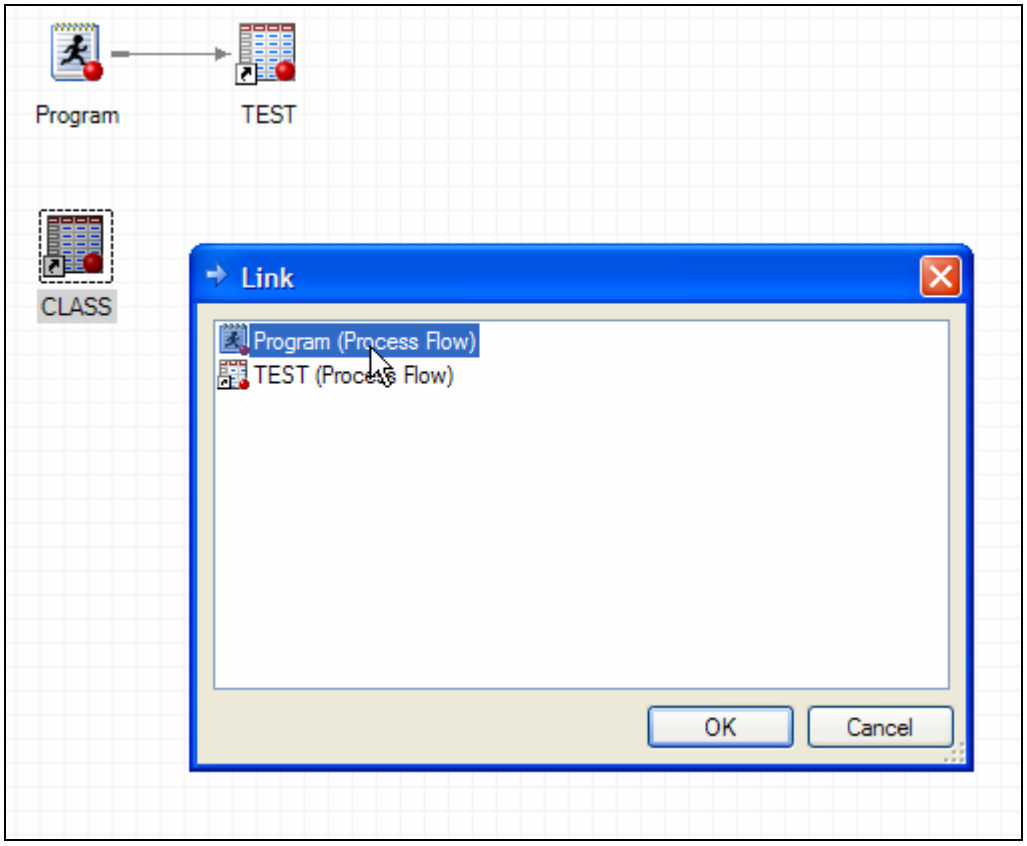
This is the situation, if you don't manually link the node. The data set CLASS is an input to the SAS program. The process flow does not reflect this



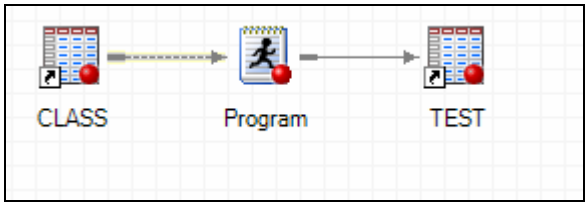
To link the CLASS data set to the program node, you select the CLASS data set and then do a right click to invoke object menu. Then you select Link Class to….



A selection screen then pops up. It allows you to select the node you want to link to, in our case the program node. You then click OK.



The resulting process flow a set of three nodes properly connected (noticed the connection added manually is a dotted line). Exactly what we want!



Having a process flow with nodes properly connected is a healthy practice. First it ensures that your process flow will run in the right order. Second, it makes your EG project self-documented by reflecting the real inputs/outputs of all the nodes.

Question 3 : How can you read a database table with a name that contains a blank?

To use table names in your SAS program that are not valid SAS names, use one of the following techniques:

- Use the PROC SQL option DQUOTE= and place double quotation marks around the table name. The libref must specify PRESERVE_TAB_NAMES=YES. For example:

```
libname mydblib oracle user=testuser password=testpass
        preserve_tab_names=yes;

proc sql dquote=ansi;
   select * from mydblib."my table";
quit;
```

- Use name literals in the SAS language. The libref must specify PRESERVE_TAB_NAMES=YES. For example:

```
libname mydblib oracle user=testuser password=testpass
      preserve_tab_names=yes;

proc print data=mydblib.'my table'n;
run;
```

Note that you may specify the alias PRESERVE_NAMES= to save time if you are specifying both PRESERVE_COL_NAMES= and PRESERVE_TAB_NAMES= in your LIBNAME statement.

It is worth explaining what a SAS name literal is. It is a name that is expressed as a string within quotation marks, followed by the letter n. Name literals allow you to use special characters (or blanks) that are not otherwise allowed in SAS names. Name literals are especially useful for expressing DBMS column and table names that contain special characters. They can also be used in a SAS Data Step for variable names such as in the following example.

```
data test;

      'My Var'n = 1;
      if 'My Var'n > 0 then 'My Second Var'n = 10;

run;
```

Question 4 : Is there a way to hard code functions in SAS EG as well as use the "point-and-click" functions?

There are two approaches to "extend" the Enterprise Guide functionality: by building a *Custom Task* or by creating a *Stored Process*.

A custom task extends Enterprise Guide by adding functionality directly into the product as if it was provided by SAS Institute. The Custom Tasks are developed using the Microsoft .Net Framework and compatible programming languages (such as VB.Net or C#). SAS provides class libraries and controls to help the developer with the Graphical Interface and the communication between the custom task and the core product. Custom tasks can meet some very specific data analysis and/or processing requirements while integrating very well wit the rest of EG. It is the Cadillac but requires a serious effort to develop.

Stored processes have been introduced with SAS 9. They are SAS programs stored centrally on a server and registered in a SAS Metadata Server. They can be executed by any client applications compatible with the new SAS Intelligent Architecture. Once properly registered, a stored process can be surfaced in EG and be executed on a SAS server. The interesting feature is that when a EG user launches a stored process, a graphical user interface gets built on the fly. This interface allows the user to enter any parameters required for execution. Contrary to custom tasks, stored processes are easy to build and do not require any programming languages other than SAS.

References:

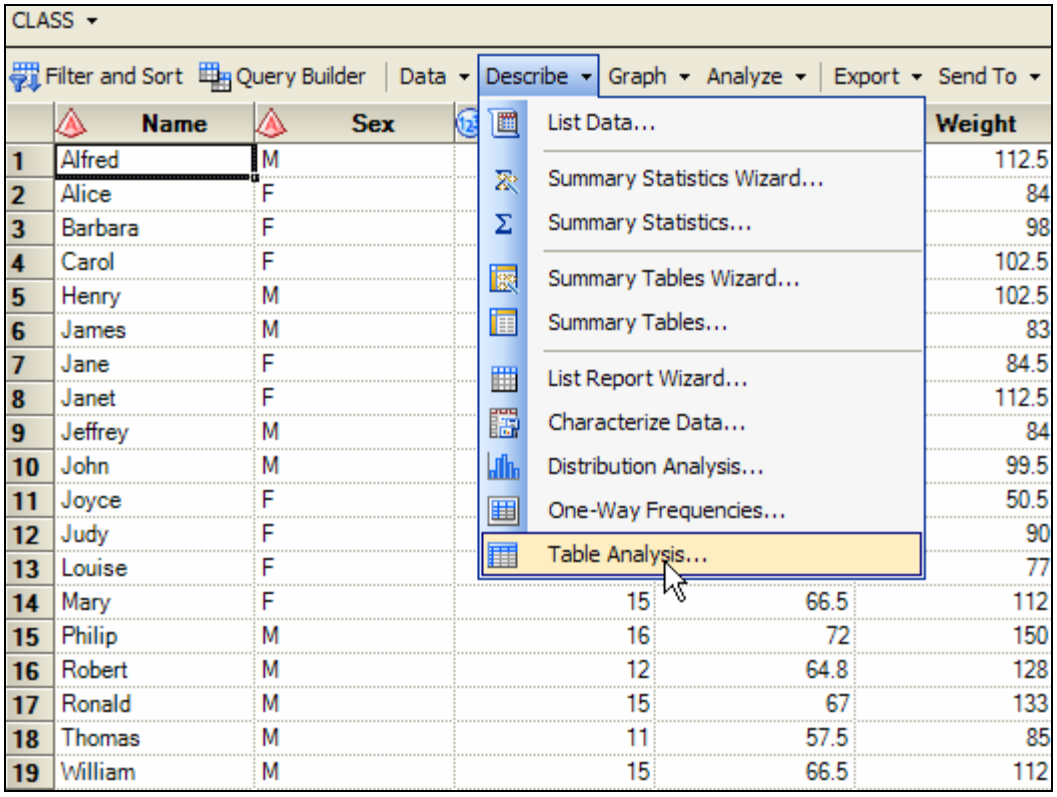
SAS Global Forum Paper A10-2007, Develop, Create and Deploy Sample Custom Tasks in SAS Enterprise Guide, Steve Gunawan

SAS Global Forum Paper 218-2007, SAS® Enterprise Guide® and Stored Processes, Frederick E. Pratter

Question 5 : How do I perform multiple tables through point and click in SAS EG?

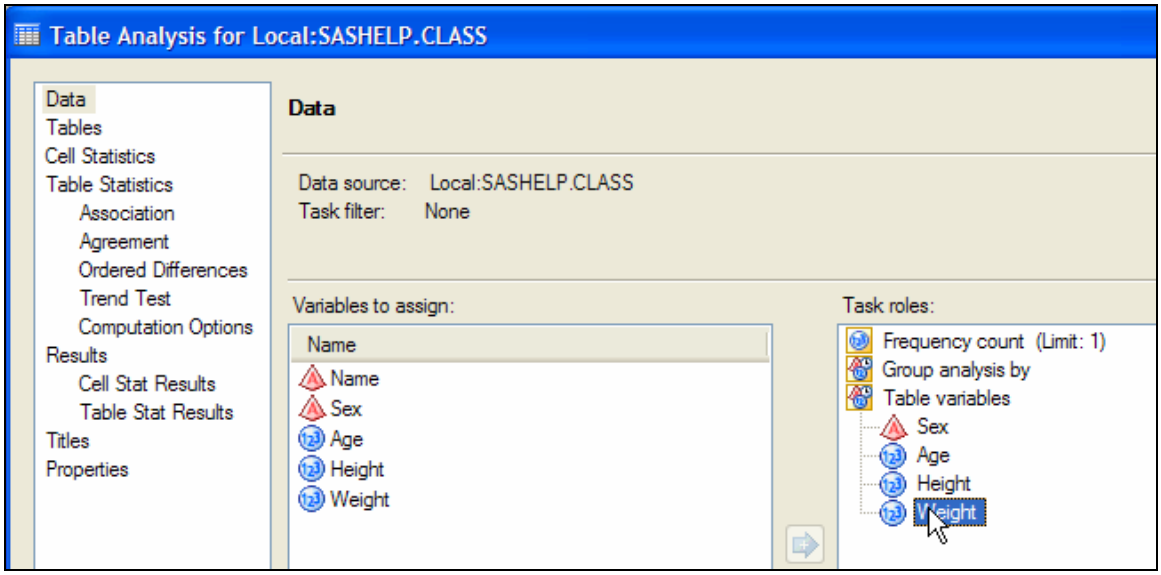
It is easy to produce multiple tables by using the Table Analysis wizard.

You can find it under the Describe menu in both EG 4.1 and EG 4.2.

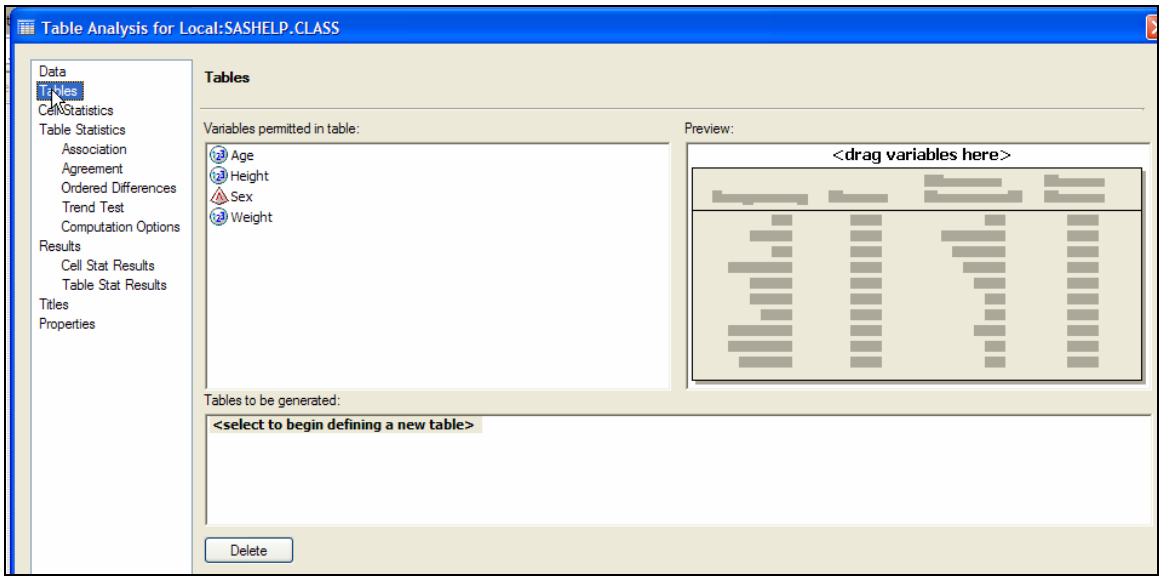


	Name	Sex			Weight
1	Alfred	M			112.5
2	Alice	F			84
3	Barbara	F			98
4	Carol	F			102.5
5	Henry	M			102.5
6	James	M			83
7	Jane	F			84.5
8	Janet	F			112.5
9	Jeffrey	M			84
10	John	M			99.5
11	Joyce	F			50.5
12	Judy	F			90
13	Louise	F			77
14	Mary	F	15	66.5	112
15	Philip	M	16	72	150
16	Robert	M	12	64.8	128
17	Ronald	M	15	67	133
18	Thomas	M	11	57.5	85
19	William	M	15	66.5	112

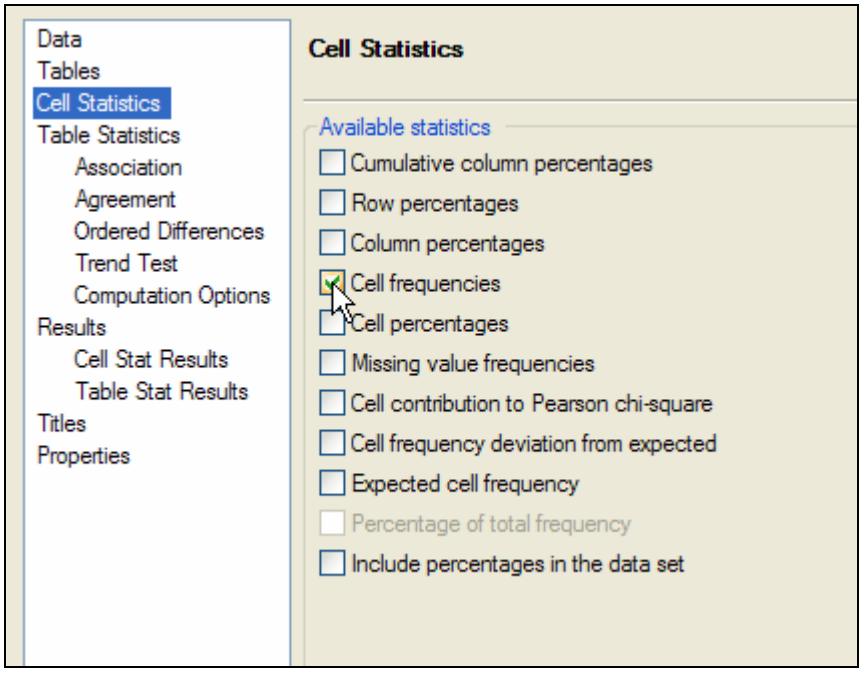
You then select the variables you wish to build your tables on.



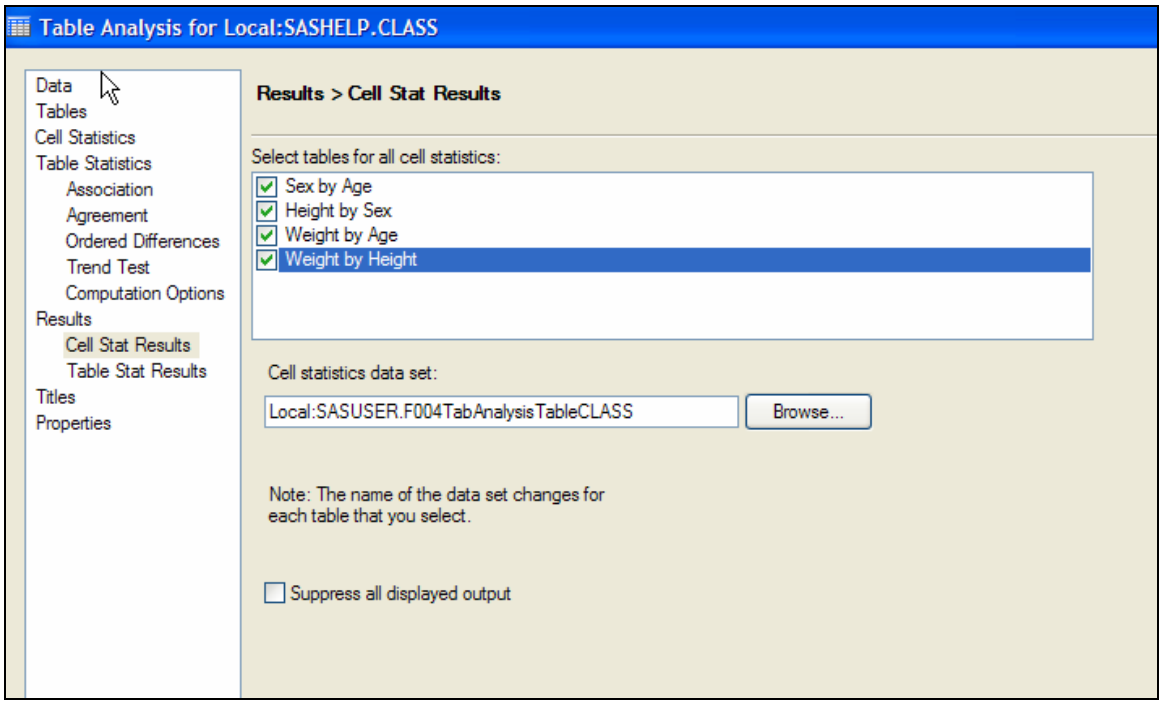
Each table can then be specified under the Tables tab.



You then select the appropriate statistics under the cell statistics tab.



Finally you must select which table to run. You can run as many tables as you want, but they must have been defined previously.



Here are the results !!

Table Analysis

Results

The FREQ Procedure

Table of Sex by Age

Sex		Age						Total
		11	12	13	14	15	16	
F	Frequency	1	2	2	2	2	0	9
	Col Pct	50.00	40.00	66.67	50.00	50.00	0.00	
M	Frequency	1	3	1	2	2	1	10
	Col Pct	50.00	60.00	33.33	50.00	50.00	100.00	
Total	Frequency	2	5	3	4	4	1	19

Table of Height by Sex

Height		Sex		Total
		F	M	
51.3	Frequency	1	0	1
	Col Pct	11.11	0.00	
56.3	Frequency	1	0	1
	Col Pct	11.11	0.00	
56.5	Frequency	1	0	1
	Col Pct	11.11	0.00	
57.3	Frequency	0	1	1
	Col Pct	0.00	10.00	
57.5	Frequency	0	1	1

Here is the generated code. Notice the multiple TABLES statement.

```
31 PROC FREQ DATA = WORK.SORTTempTableSorted
32    ORDER=INTERNAL
33 ;
34    TABLES Sex * Age /
35       NOROW
36       NOPERCENT
37       NOCUM
38       SCORES=TABLE
39       ALPHA=0.05  OUT=SASUSER.F003TabAnalysisTableCLASS(LABEL="Cell statistics for Sex by Age for SASH
40 ;
41    TABLES Height * Sex /
42       NOROW
43       NOPERCENT
44       NOCUM
45       SCORES=TABLE
46       ALPHA=0.05  OUT=SASUSER.F001TABANALYSISTABLECLASS_0000(LABEL="Cell statistics for Height by Sex
47 ;
48    TABLES Weight * Age /
49       NOROW
50       NOPERCENT
51       NOCUM
52       SCORES=TABLE
53       ALPHA=0.05  OUT=SASUSER.F002TabAnalysisTableCLASS(LABEL="Cell statistics for Weight by Age for S
54 ;
55    TABLES Weight * Height /
56       NOROW
57       NOPERCENT
58       NOCUM
59       SCORES=TABLE
60       ALPHA=0.05  OUT=SASUSER.F004TabAnalysisTableCLASS(LABEL="Cell statistics for Weight by Height fo
61 ;
```

Question 6 : How can you export HTML into Excel? To use in a presentation with other data (without having to re-format it).

An answer to this question was provided in the Fall 2005 Q&A.