Best Tips, Tricks, and Code Snippets from 30 Years Programming in SAS

Jeff LaMar

2025 Iowa and Nebraska SAS User Groups

May 19, 20

Objective

- The objective is for everyone to walk away with at least one idea/tidbit/code snippet that they can implement immediately in their current role
- Presentation will go from basic / common sense to more advanced – so hold on tight!

Topics To Cover

- Best Practices / Advice
- Stop the madness (c’mon man!)
- Error checking / Debugging
- Key Knowledge & Tidbits (Macros & Call Symputx)
- The MOST USEFUL and powerful technique (looping code)
- Key Knowledge & Tidbits (Snippets, Ordered Lists, etc.)

Audience Questions

- How long have you been coding in SAS:
  - <= 3 years?
  - 3 to 10 years?
  - 10 years plus?

Best Practices / Advice

- Create one program that has sample code snippets for later reference – go here first before “Googling”
  - (Example code notes: transpose, arrays, counters, univariate, etc.)
  - You will create many programs during your “analytical lifetime”
- Get to know the SAS Macro language and start at the basic level (it is the most powerful piece of SAS!)
- Duplicates, duplicates, oh my! Know your joins and merges. Understand your Unique's.
  - ➢ One to One
  - ➢ One to Many
  - ➢ Many to Many (Cartesian Join – Yuk!)
- Always sanity check your data for counts, always!
- Always check your code for duplicates, always!
- You will “inherit” code throughout your career – take some time to learn what it’s doing, even if complex

Stop the Madness!

If You're doing these things, stop! C'mon Man!

- Do not create macros for the sake of creating macros! Don't create macros, within macros, within macros …..
  - Why? It's hard to debug and just creates confusion.
- Create macro variables on things that can change (think of it as a Parameter setting) - If it's static, you don't really need to make it a macro variable
- End all macros with %mend <macro name> Why? It helps with reading the code, especially, with lots of embedded macros
- Do not Replace datasets with the same name (including SQL)! Why? You can't debug. You may have destroyed your input dataset and won't be able to check any transformations.
- Do not create complex SQL Queries with tons of inner joins and subselect queries. Why? Again, you can't debug very easily (do you see a theme here?)
  - ➢ Break up your code into manageable sections
  - ➢ People who have to audit (or get the pleasure of inheriting) your code will be very, very thankful

Stop the Madness!

If You’re doing these things, stop! C’mon Man!

Do not create fancy header records. Why? It’s a waste of time to update and “make pretty”. Use block comments that can easily be updated (for others who might inherit your code)

Poor Header setup:

```
/**********************************************
PROGRAM:  rock_star.sas                       *
AUTHOR:   Jeff LaMar                          *
CREATED:  April 2025                          *
-----------------------------------------------
DESCRIPTION                                   *
Rock Star list                                *
-----------------------------------------------
CHANGE HISTORY                                *
Apr 2025 Original program created             *
May 2025 Reluctantly added Taylor Swift       *
**********************************************/
```

Better way (take out the asterisks so you don’t have to mess with them if you want to change something)

```
/**********************************************
PROGRAM: rock_star.sas
AUTHOR:  Jeff LaMar
CREATED: April 2025
-----------------------------------------------
DESCRIPTION
Rock Star list
-----------------------------------------------
CHANGE HISTORY
Apr 2025 Original program created
May 2025 Reluctantly added Taylor Swift
**********************************************/
```

Error Checking / Debugging

- Remember, the Semi-colon is your friend!
  - If you get an error, it should be automatic that you check for a missing Semi-colon
- Solve errors top down. Many times errors propagate - fix the first one and re-run
- Always check your log before going any further, always, especially with PC SAS (i.e. Do NOT run code and then open the dataset without checking the log!)
- Search for these key words when checking the log
  - error
  - warning (note: please try to modify code where you have warnings)
  - uninit (short for uninitialized) - IMPORTANT: do not discount this advice! – SAS will give an "Uninitialized NOTE" if, in new variable assignment statement, you try to use a variable that doesn't exist in the DATA set. Your new variable will exist but be totally hosed up. SAS gives a NOTE for this and not an error!
- KEY advice: Do NOT <u>name</u> any of your datasets or <u>make comments</u> containing strings of "error", "warning", or "uninit" Why? You'll pick up these strings when you're checking the log for the real thing.
  - I like to use "issue" instead of "error"

Key Knowledge & Tidbits

- **SAS Macros 101 - Macro Variables**
  - SAS Macro Variables are just a string of characters (i.e. NOT a datatype of any sort)
  - SAS will resolve the macro variable as a string of characters.

  <u>Example 1 (Macro assignment and usage)</u>
  - %let drummer = Ringo; *** This assigns macro variable;
  - Usage: Where top_drummer = "&drummer"; (Important note, if you need to put tick marks around a macro variable, you need to use double quotes). The ampersand tells SAS to resolve the macro variable.

  <u>Example 2 (including the quotes in the macro assignment)</u>
  - %let drummer = 'Ringo'; (Note: The quotes are just part of the text string and nothing else)
  - Usage: Where top_drummer = &drummer; (Note: I don't need quotes at all, because SAS will resolve the string which already includes quotes needed for a character variable)

  <u>Example 3</u>
  - %let Billboard_cutoff = 25;
  - Usage: Where Billboard_occurrences > &Billboard_cutoff; (The ampersand tells SAS to grab the macro variable and will insert 25 into the code)

Key Knowledge & Tidbits

- **SAS Macros 101 – Macro code**

  General Macro code:

  ```
  %macro rock_it_out;
       /* Only use block comments in macros */
       *** Sometimes line comments mess the macro up;
       < code goes here >
  %mend rock_it_out;
  ```

  Command to execute the macro:

  ```
  %rock_it_out;
  ```

IMPORTANT TIDBIT HERE! on Apostrophes and Contractions in comments

STAY AWAY from contractions(i.e. it’s) and single apostrophes in your comments!

In certain circumstances, it’s not an issue but sometimes it is, especially with macro programming. So it’s best to get into the habit of avoiding them all together

```
*** Don’t have contractions in comments; “bad” comment
/* Don’t have contractions in comments */ “bad” comment

*** Do NOT have contractions in comments: “better comment”
/* Do NOT have contractions in comments */ “better comment”
```

Key Knowledge & Tidbits

<u>Special macro call function: CALL SYMPUTX</u>

Purpose – To assign a value to a macro variable during the execution of a data step.
This is an EXTREMELY useful and powerful SAS function. Get to know it well!

Note: Difference between symput and symputx is that symputx removes leading and trailing blanks before assigning the value to the macro variable

Syntax: call symputx(‘macro-var’, character-value)

Example usage: Note: run date on 5/15/2025

```
data _null_;
Name = 'Switchfoot';
call symputx('yyyymm',put(today(),yymmn8.));
call symputx('today',"'"||put(today(),date9.)||"'d");
call symputx('band_name',Name);
run;

%put yyyymm = &yyyymm;
%put today = &today;
%put band_name = &band_name;
```

Example log

```
%put yyyymm = &yyyymm;
yyyymm = 202505
%put today = &today;
today = ’15MAY2025'd
%put band_name = &band_name;
band_name = Switchfoot
```

The MOST useful code (looping algorithm)

The following section contains code that I have used in EVERY job over the last 15 years. This code loops through an "incoming" dataset, performs key operations, and stacks the final results. It is extremely versatile and can be applied in multiple situations:

Here are examples of how I've previously used this technique:

- Check lengths of all character variables in a dataset
- Run proc univariate on all numeric variables in a dataset
- Check table counts on all tables in a Teradata schema
- Run a distribution analysis on character variables
- Run Statistical Process Control charts / or line charts on a list of numeric variables

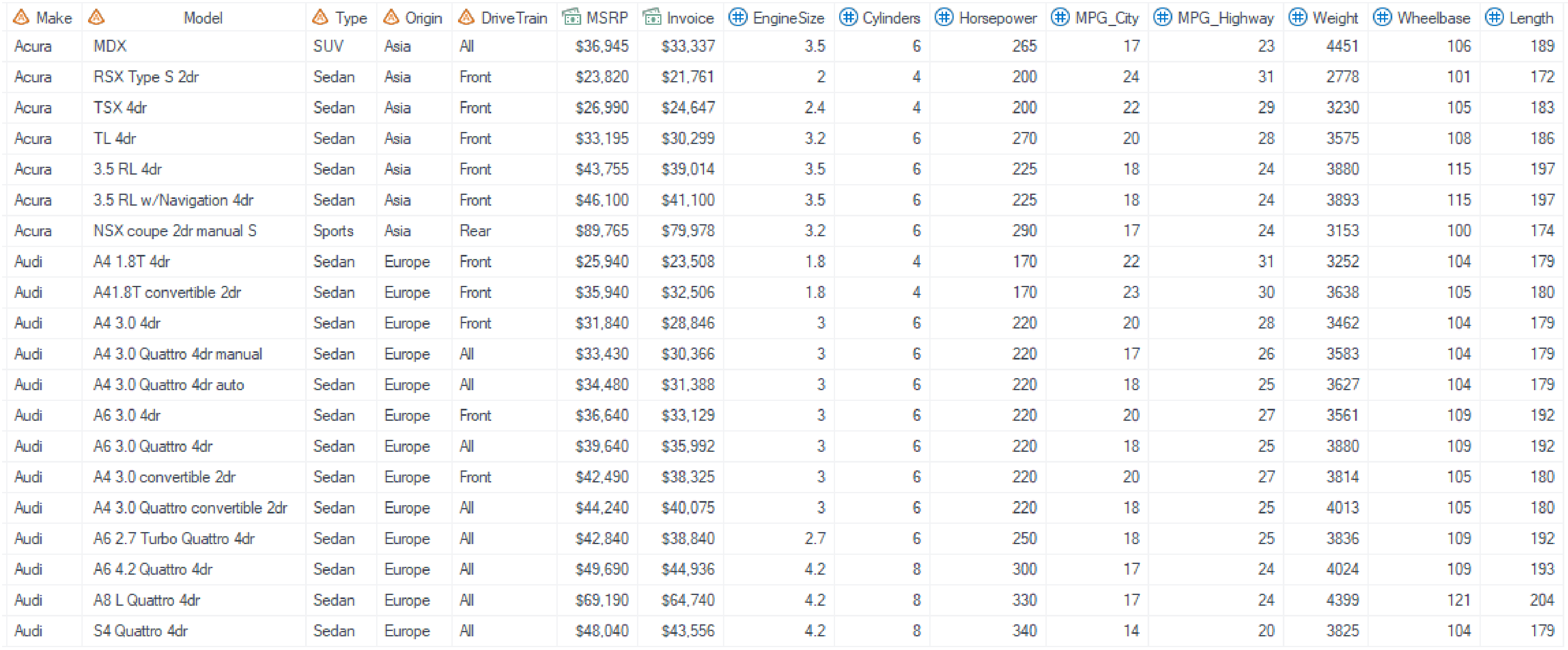
Code Overview:

```
Start with a SAS Dataset or database table
Created an "incoming" dataset that contains the key field you want to loop through
        Set obs_nbr = _N_;
Set up a macro that loops through the key field (i = 1 to total_obs)
        Delete temporary datasets created in the macro
        Pull in the "incoming" dataset
        Use Symputx to create macro variables based on a row in the incoming table
        Perform operations on the "main" dataset (this varies depending on your objective)
        Run proc append to stack results
Macro End
Initialize run by deleting the "appended" dataset (i.e. start fresh)
Run Looping Macro
```

The MOST useful code (looping algorithm) Basic Code Template

The best way to explain the code is to provide an example. Use this example as your “looping template”. I basically copied my “looping” template and adjusted accordingly.

Objective: Loop through sashelp.cars dataset and create a final dataset containing distributions of values for all the character variables in the dataset.
SASHELP. CARS:



Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower	MPG_City	MPG_Highway	Weight	Wheelbase	Length
Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5	6	265	17	23	4451	106	189
Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2	4	200	24	31	2778	101	172
Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4	4	200	22	29	3230	105	183
Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6	270	20	28	3575	108	186
Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6	225	18	24	3880	115	197
Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	$46,100	$41,100	3.5	6	225	18	24	3893	115	197
Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765	$79,978	3.2	6	290	17	24	3153	100	174
Audi	A4 1.8T 4dr	Sedan	Europe	Front	$25,940	$23,508	1.8	4	170	22	31	3252	104	179
Audi	A41.8T convertible 2dr	Sedan	Europe	Front	$35,940	$32,506	1.8	4	170	23	30	3638	105	180
Audi	A4 3.0 4dr	Sedan	Europe	Front	$31,840	$28,846	3	6	220	20	28	3462	104	179
Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	$33,430	$30,366	3	6	220	17	26	3583	104	179
Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	$34,480	$31,388	3	6	220	18	25	3627	104	179
Audi	A6 3.0 4dr	Sedan	Europe	Front	$36,640	$33,129	3	6	220	20	27	3561	109	192
Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All	$39,640	$35,992	3	6	220	18	25	3880	109	192
Audi	A4 3.0 convertible 2dr	Sedan	Europe	Front	$42,490	$38,325	3	6	220	20	27	3814	105	180
Audi	A4 3.0 Quattro convertible 2dr	Sedan	Europe	All	$44,240	$40,075	3	6	220	18	25	4013	105	180
Audi	A6 2.7 Turbo Quattro 4dr	Sedan	Europe	All	$42,840	$38,840	2.7	6	250	18	25	3836	109	192
Audi	A6 4.2 Quattro 4dr	Sedan	Europe	All	$49,690	$44,936	4.2	8	300	17	24	4024	109	193
Audi	A8 L Quattro 4dr	Sedan	Europe	All	$69,190	$64,740	4.2	8	330	17	24	4399	121	204
Audi	S4 Quattro 4dr	Sedan	Europe	All	$48,040	$43,556	4.2	8	340	14	20	3825	104	179

The MOST useful code (looping algorithm) - Basic Code Template

```
*********************************************************************************;
*** Standard loop code;
*** Loop through the files and build the dataset;
*********************************************************************************;
*** Start with an "incoming" dataset;
*** In this dataset, set obs_nbr = _N_ for looping;
*** Then get total observations;

*********************************************************************************;
*** Get names of character variables from the “main dataset”;
*** In this case, it is from sashelp.cars (but it could be anything)
*********************************************************************************;

proc contents noprint data=sashelp.cars out=char_attrs(where=(type=2) keep=name
type libname memname memlabel); run;

*** Add obs_nbr to incoming dataset (this will be used to drive the looping);
data incoming;
set char_attrs end = last;
where name NE 'Model'; *** Removing model from this analysis;
obs_nbr = _N_;
if last then call symputx('total_obs',obs_nbr);
run;

*** Get total observations for input into looping macro;
%put total_obs = &total_obs;
```

The MOST useful code (looping algorithm) - Basic Code Template Cont.

```
%macro car_dist(obs);
%do i = 1 %to &obs;
/* Use when testing - Need to comment when running else you will be in an infinite loop! */
/*   %let i = 1; */

/* Read incoming dataset (row by row) and create macro variables for later use */
     data null_;
     set incoming;
     where obs_nbr = &i.;
     call symputx ('name',strip(name));
     call symputx ('libname',strip(libname));
     call symputx ('memname',strip(memname));
     run;

/* Output macro variables in log */
     %put obs=&i, name=&name, libname=&libname, memname=&memname;

/* Code goes here: could be lots of things */
/* Important!: "Standardize the length of character variables before proc append */
     proc sql;
     create table char_dist as
     select "&libname..&memname" as Dataset length=32, "&name." as Var length=32
     , &name. as Value length=32, count(1) as Counts
     from sashelp.cars(keep=&name.)
     group by 1,2,3
     order by 1,2,3
     ;
     quit;

/* In many cases, appending is done */
     proc append base=char_distributions data=char_dist; run;
/* Delete datasets created in the loop */
     proc datasets library=work nolist; delete char_dist; quit;

%end;
%mend car_dist;
```

The MOST useful code (looping algorithm) - Basic Code Template Cont.

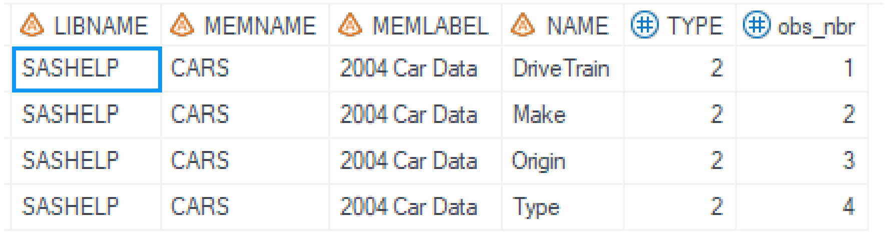
```
Run the macro here:

*** Delete the appended dataset for a fresh run;
proc datasets library=work nolist; delete char_distributions; quit;

*** Run the looping macro;
%car_dist(&total_obs);
```

Screenshots:

Incoming:

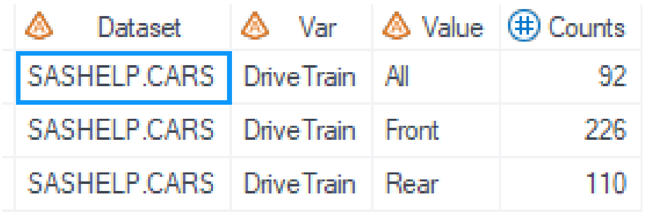


LIBNAME	MEMNAME	MEMLABEL	NAME	TYPE	obs_nbr
SASHELP	CARS	2004 Car Data	DriveTrain	2	1
SASHELP	CARS	2004 Car Data	Make	2	2
SASHELP	CARS	2004 Car Data	Origin	2	3
SASHELP	CARS	2004 Car Data	Type	2	4

Macro Output (from 1st row of incoming):

obs=1, name=DriveTrain, libname=SASHELP, memname=CARS

Char_dist:

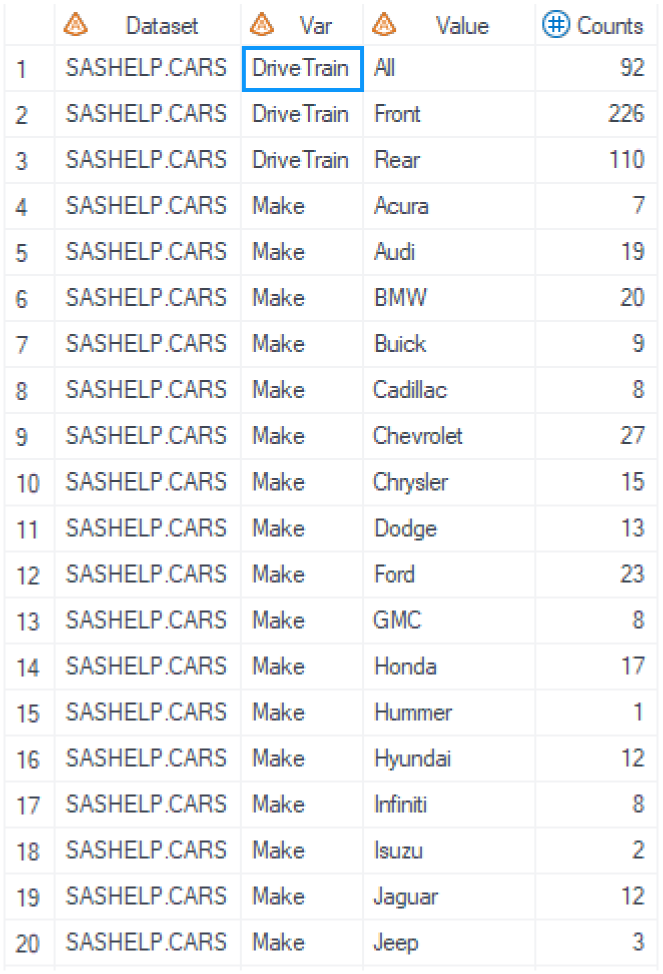


Dataset	Var	Value	Counts
SASHELP.CARS	DriveTrain	All	92
SASHELP.CARS	DriveTrain	Front	226
SASHELP.CARS	DriveTrain	Rear	110

The MOST useful code (looping algorithm) - Basic Code Template Cont.

Screenshots:

Final Stack (Char_distributions):



	Dataset	Var	Value	Counts
1	SASHELP.CARS	DriveTrain	All	92
2	SASHELP.CARS	DriveTrain	Front	226
3	SASHELP.CARS	DriveTrain	Rear	110
4	SASHELP.CARS	Make	Acura	7
5	SASHELP.CARS	Make	Audi	19
6	SASHELP.CARS	Make	BMW	20
7	SASHELP.CARS	Make	Buick	9
8	SASHELP.CARS	Make	Cadillac	8
9	SASHELP.CARS	Make	Chevrolet	27
10	SASHELP.CARS	Make	Chrysler	15
11	SASHELP.CARS	Make	Dodge	13
12	SASHELP.CARS	Make	Ford	23
13	SASHELP.CARS	Make	GMC	8
14	SASHELP.CARS	Make	Honda	17
15	SASHELP.CARS	Make	Hummer	1
16	SASHELP.CARS	Make	Hyundai	12
17	SASHELP.CARS	Make	Infiniti	8
18	SASHELP.CARS	Make	Isuzu	2
19	SASHELP.CARS	Make	Jaguar	12
20	SASHELP.CARS	Make	Jeep	3

This is just one example:

In past projects, I have pulled character distributions from a prior month and compared to current month. Then checked distribution percentages for significant distribution shifts (The technique is called Portfolio Stability Index).

You can also do something similar with the numeric variables and compare min, max, #missing, etc.

Key Knowledge – SAS Snippets
Used in both Enterprise Guide and SAS Studio

- I use snippets ALL the time! This is an extremely powerful tool
- What are SAS snippets?
  - A SAS snippet is simply a reusable block of SAS code, typically containing frequently used or complex code that you want to insert into your program quickly and easily
  - Snippets can be customized for later use which makes the programming process more efficient
  - You can insert the snippet by typing the abbreviation and hit enter
  - Snippets allow you to insert “shell” code into your program which reduces repetitive typing (and errors)
  - My all-time favorite snippet I use is called “pfreq”. When I enter pfreq, it gives me this (why type it out? I just need to fill in data and tables):

```
proc freq data = xxx; tables yyyy / missing list; run;
```

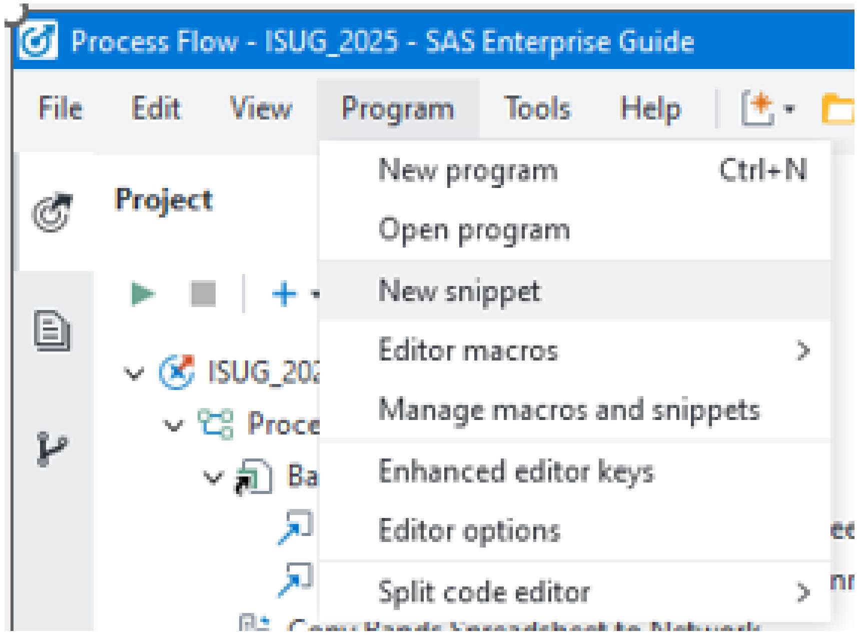
Key Knowledge – SAS Snippets
How to set up in Enterprise Guide

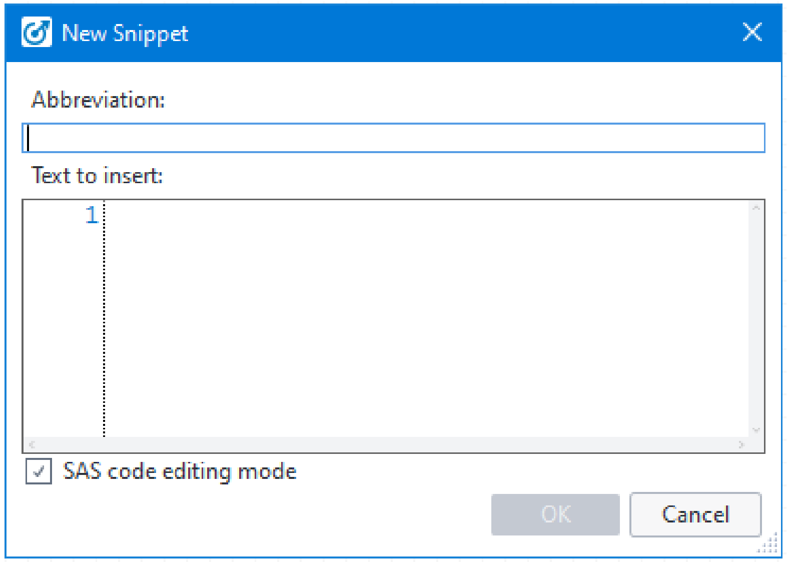
- Type up the code you want as a snippet, highlight and "copy" (i.e. Ctrl-C)

  Example:

  ```
  proc freq data = xxx; tables yyyy / missing list; run;
  ```

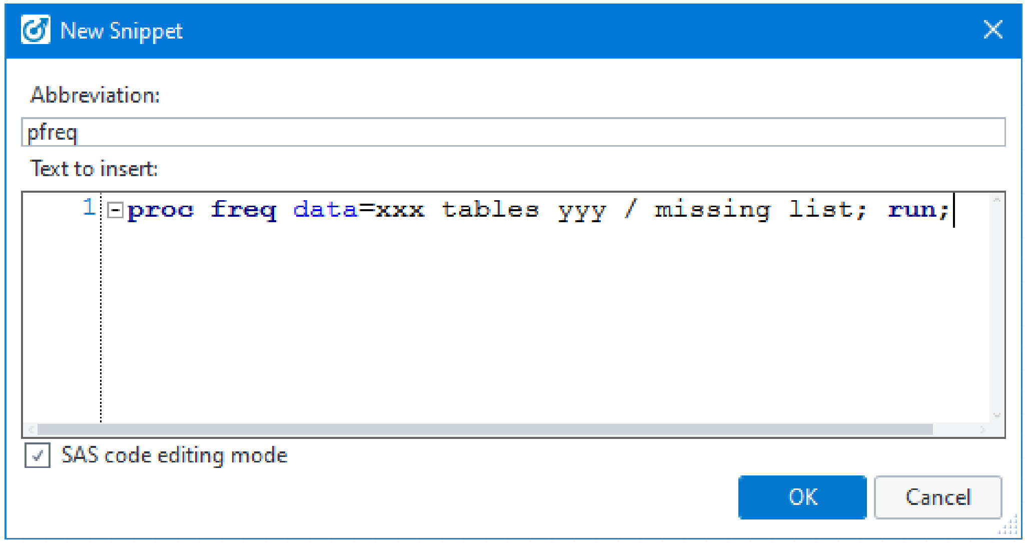
- Click on Program/New snippet… (in PC SAS, it's Tools/Add Abbreviation)





Key Knowledge – SAS Snippets
How to set up in Enterprise Guide

- Type an abbreviation name for your code snippet (I’m using pfreq)
- Paste the code you had copied previously in the “Text to insert” box (i.e. Ctrl-V)



Abbreviation:
pfreq
Text to insert:

```
proc freq data=xxx tables yyy / missing list; run;
```

- Click on OK
- To use in the SAS editor, type pfreq and hit enter – the following code will be pasted in your editor:

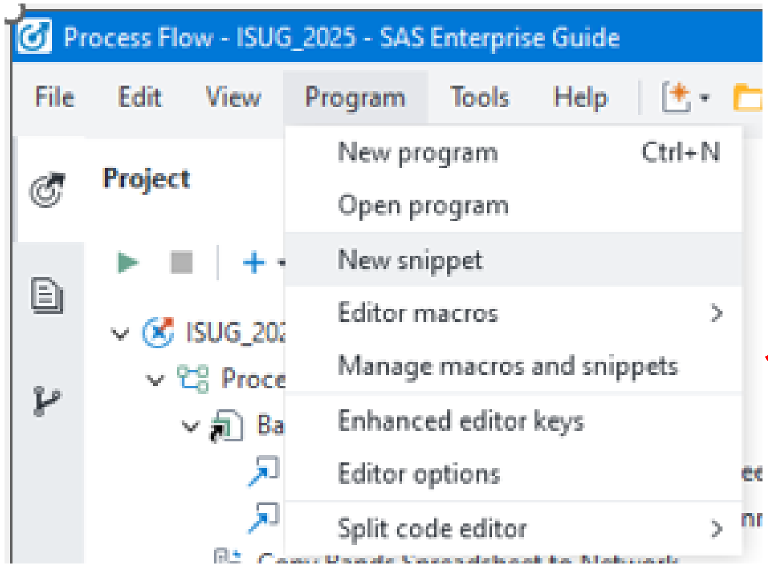
```
proc freq data = xxx; tables yyyy / missing list; run;
```

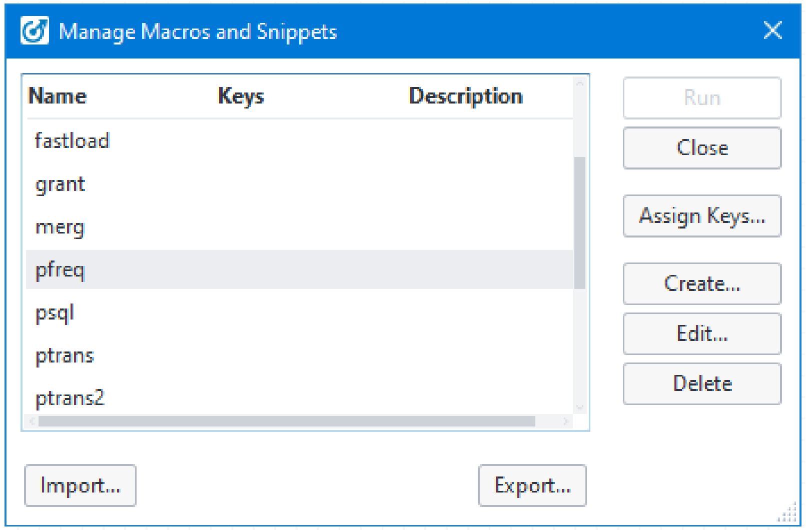
- And WaPow!, No more typing it out!

Key Knowledge – SAS Snippets

DO THIS so you won't get burned!

- IMPORTANT NOTE: If your EG SAS gets upgraded or re-installed (i.e. You are getting a new laptop/desktop), you will lose ALL your snippet that you have set up.
- To fix this potential problem, EACH time you create a snippet macro, export the macro to a designated macro folder for safekeeping (looks like pfreq.kmf). See below for how:
- Click on Program/Manage macros and snippets (in PC SAS – Tools/Keyboard Macros/Macros)





- Highlight the snippet name and click on "Export" to save to a location of your choice
- Note: As you can see, you can also "Import" a snippet (which will obviously come in handy if you need to get your snippets back!)

Key Knowledge – SAS Snippets Most Used

Snippet Examples (How to use: type in the snippet and hit "enter")
Be creative! I usually add a "p" in front when defining the abbreviation
For the most part, don't use real words (snippets will pop up unnecessarily)
Set up Snippets for:

1. Code you are using over and over!
2. Code where you don't have the syntax memorized
3. Shell code

cmt (set the length you use for comments)

```
****************************************************************;
```

pfreq

```
proc freq data = xxx; tables yyyy / missing list; run;
```

psql

```
proc sql;
create table xxxx as
select
from
join
on
where
group by
order by
;
quit;
```

Key Knowledge – SAS Snippets Most Used cont.

Duplicate Check

Purpose: To check for “duplicate” records. This code will identify the duplicates and then merge back to the original dataset so you can see what’s going on and code accordingly if necessary. (I use this all the time and I mean ALL the time)

dupe

```
%let dupe_file = best_rock_bands;
%let dupe_key = band_name,best_song; /* add comma between names */

%let group_by = 1,2; /* Update "group by" if more than one dupe_key */
%let key1 = band_name;
%let key2 = best_song;
%let key3 = xxxx;

proc sql;
create table dupes as
select &dupe_key., count(*) as counts
from &dupe_file.
group by &group_by.
having counts > 1
;
quit;

proc sql;
create table duplify as
select b.counts, a.*
from &dupe_file. a
join dupes b
on a.&key1. = b.&key1. /* Update "on" if more than one dupe key */
and a.&key2 = b.&key2.
/*and a.&key3 = b.&key3.*/
order by a.&dupe_key.
;
quit;
```

Key Knowledge – SAS Snippets Most Used cont.

Datasets Match Check

Purpose: To investigate and compare how two datasets match on the “by” variables. Also check for records which show up in one dataset and not the other. (Note: Again, I have this set this up as a macro snippet)

Note: For this code, check the log and it will indicate the observations in each dataset for a quick assessment of what has matched and not matched.

merg

```
*** Prep datasets for merge;
proc sort data=table1; by var1; run;
proc sort data=table2; by var2; run;


data matched table1_only table2_only;
merge table1(in=a) table2(in=b);
by var1;
if a and b then output matched;
else if a and not(b) then output table1_only;
else if not(a) and b then output table2_only;
run;
```

Key Knowledge – SAS Snippets Most Used cont.

Random Selection
Purpose: To Randomly select a subset of records from a dataset

“strata” is optional (for stratified sampling).
“seed” is also optional.

Note: `selectall` is extremely important. If you don’t use selectall and the data is smaller than the sample size selected, you will get an error.

psurv

```
*** SRS - Simple Random Sampling;
proc sort data = all_bands; by band_type;

proc surveyselect noprint data=all_bands
out=all_bands_random
method = srs
seed = 8675309
sampsize=10 selectall;
strata band_type;
run;
```

Key Knowledge – SAS Snippets
Most Used cont. (especially useful in PC SAS)

The “Black Hole”

Sometimes you inadvertently have a missing quote or parenthesis, and you run some code, then SAS gets totally messed up and goes into what I call the “Black Hole”.

How do you know you are in the Black Hole? When you submit SAS code and nothing happens! (e.g. no output on a proc freq). If you check the log, you get no errors but no observations and/or output. What? . . . . YOU ARE IN THE **BLACK HOLE**!

Run this “Black Hole” code to recover your session:

You will see a bunch of errors in your log, that’s okay. Your session should be recovered from the black hole that sucks the life out of your code! I have a snippet for this:

black

```
*'; *"; *); */; %mend; run;
*'; *"; *); */; %mend; run;
*'; *"; *); */; %mend; run;
*'; *"; *); */; %mend; run;
*'; *"; *); */; %mend; run;
*'; *"; *); */; %mend; run;
*'; *"; *); */; %mend; run;
*'; *"; *); */; %mend; run;
*'; *"; *); */; %mend; run;
*'; *"; *); */; %mend; run;
```

Key Knowledge – Enterprise Guide

Ordered List

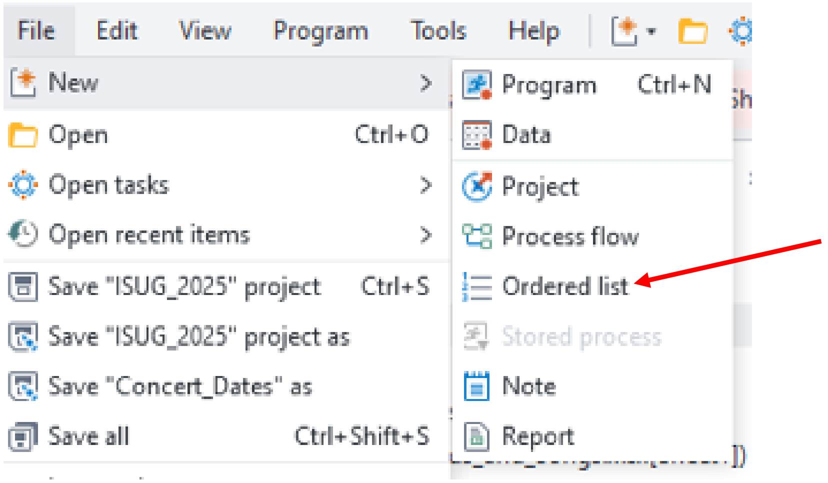
- It allows you to create a customized list of programs and tasks to run sequentially
- THIS is the BEST thing in EG – learn it!
- Game changer – it is why I code exclusively in EG

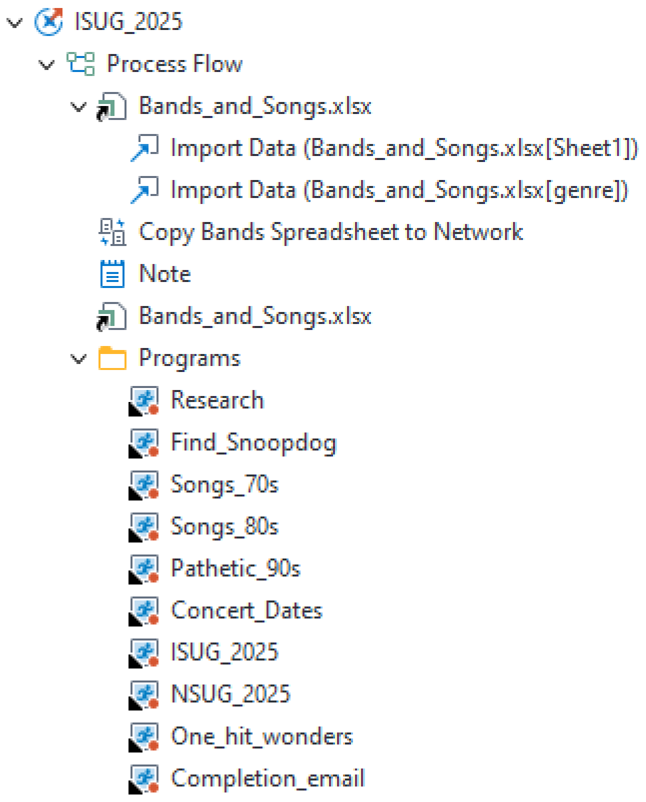
Your EG project could have multiple programs / tasks, but you only want to run a selected group:

Let's say you want to run the following in order:

- Import Data
- Songs_70s
- Songs_80s
- Pathetic_90s
- Completion_email

Here's how to do it with an ordered list:
Choose File/New/Ordered List:

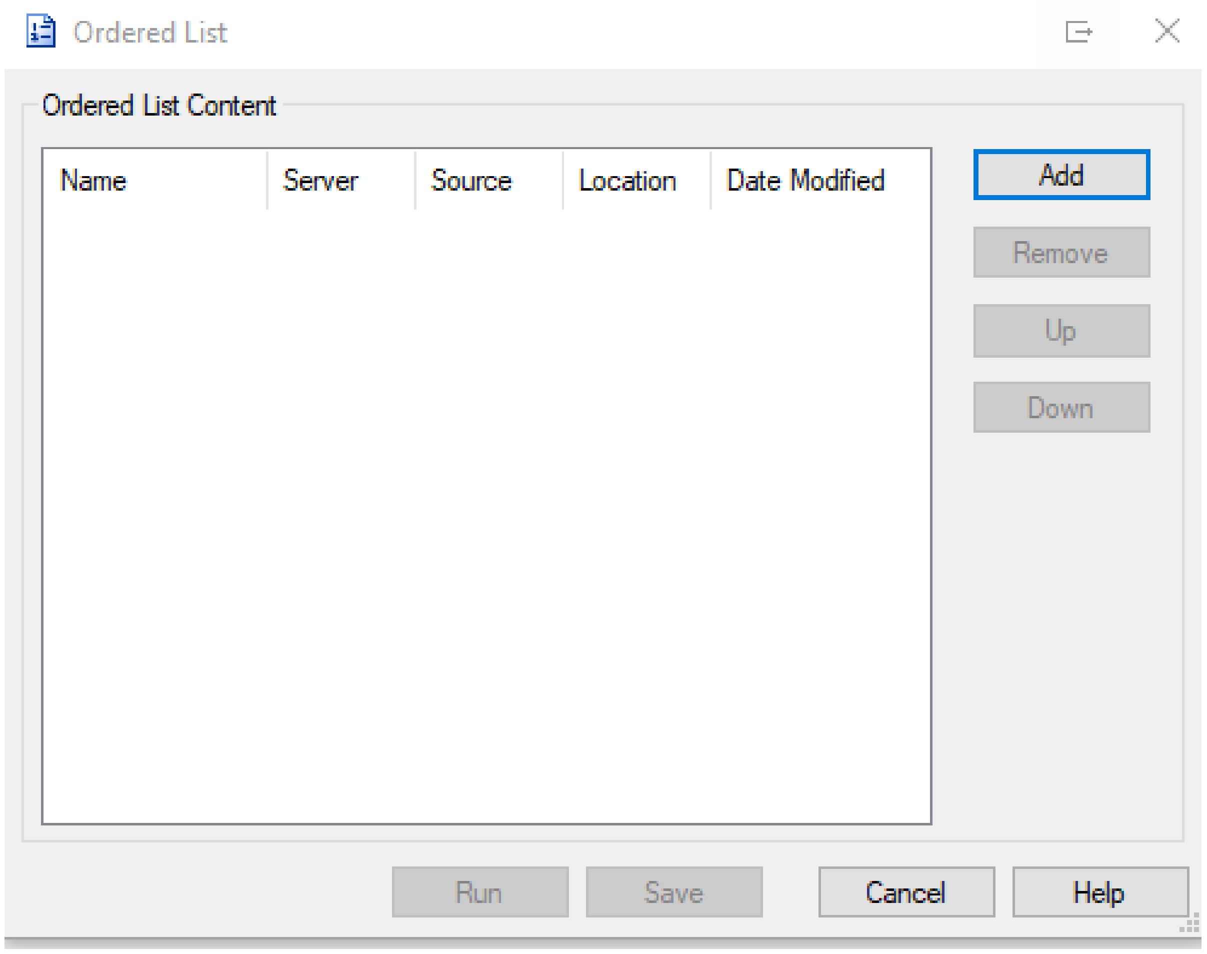




Key Knowledge – Enterprise Guide

Ordered List

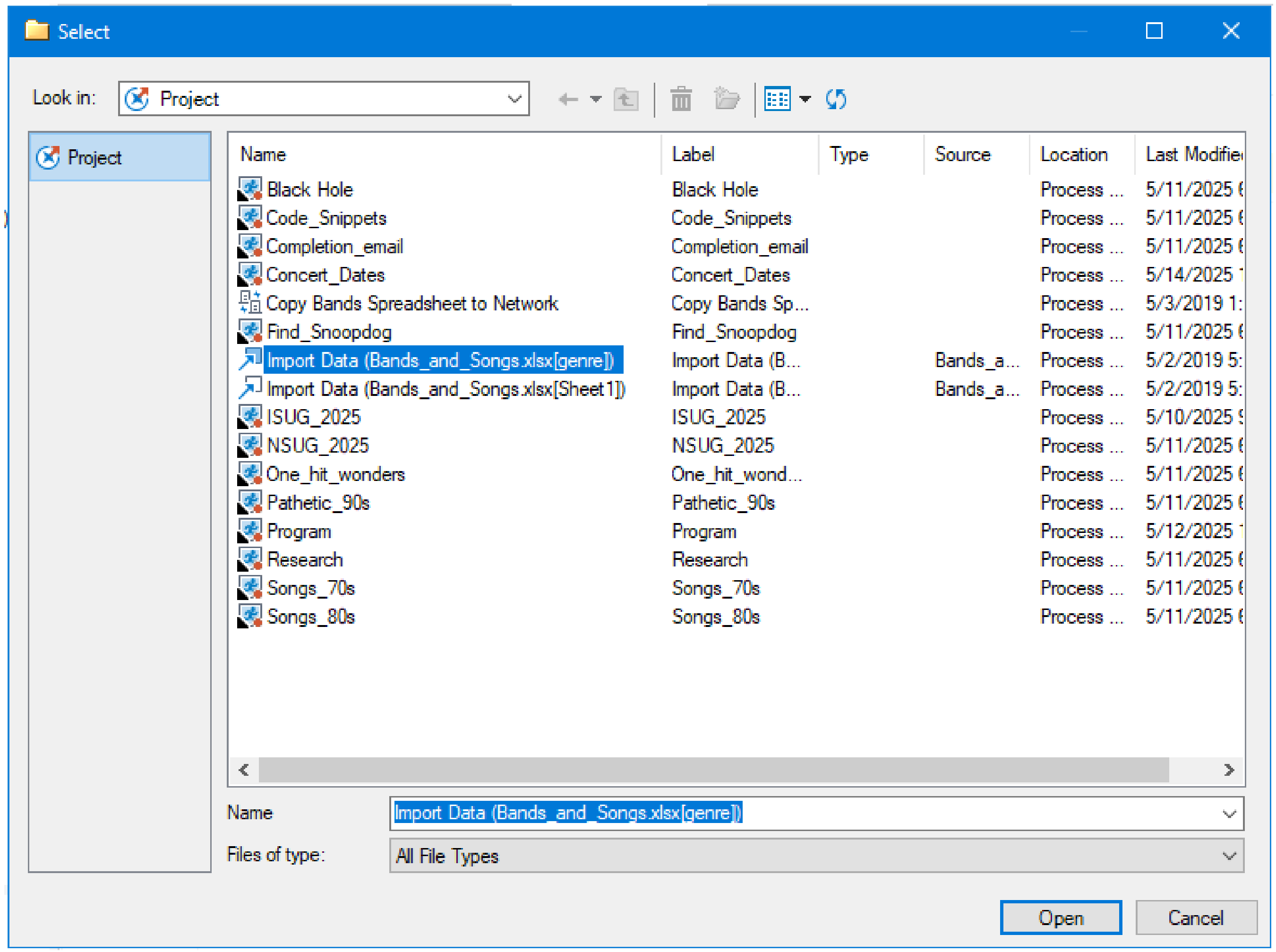
On the Ordered List box, click on “Add” to choose your Content



Key Knowledge – Enterprise Guide

Ordered List

Select the first program/task you want to run
Click on “Open”



Select

Look in: Project

Project

Name	Label	Type	Source	Location	Last Modifie
Black Hole	Black Hole			Process ...	5/11/2025 6
Code_Snippets	Code_Snippets			Process ...	5/11/2025 6
Completion_email	Completion_email			Process ...	5/11/2025 6
Concert_Dates	Concert_Dates			Process ...	5/14/2025 1
Copy Bands Spreadsheet to Network	Copy Bands Sp...			Process ...	5/3/2019 1:
Find_Snoopdog	Find_Snoopdog			Process ...	5/11/2025 6
Import Data (Bands_and_Songs.xlsx[genre])	Import Data (B...		Bands_a...	Process ...	5/2/2019 5:
Import Data (Bands_and_Songs.xlsx[Sheet1])	Import Data (B...		Bands_a...	Process ...	5/2/2019 5:
ISUG_2025	ISUG_2025			Process ...	5/10/2025 9
NSUG_2025	NSUG_2025			Process ...	5/11/2025 6
One_hit_wonders	One_hit_wond...			Process ...	5/11/2025 6
Pathetic_90s	Pathetic_90s			Process ...	5/11/2025 6
Program	Program			Process ...	5/12/2025 1
Research	Research			Process ...	5/11/2025 6
Songs_70s	Songs_70s			Process ...	5/11/2025 6
Songs_80s	Songs_80s			Process ...	5/11/2025 6

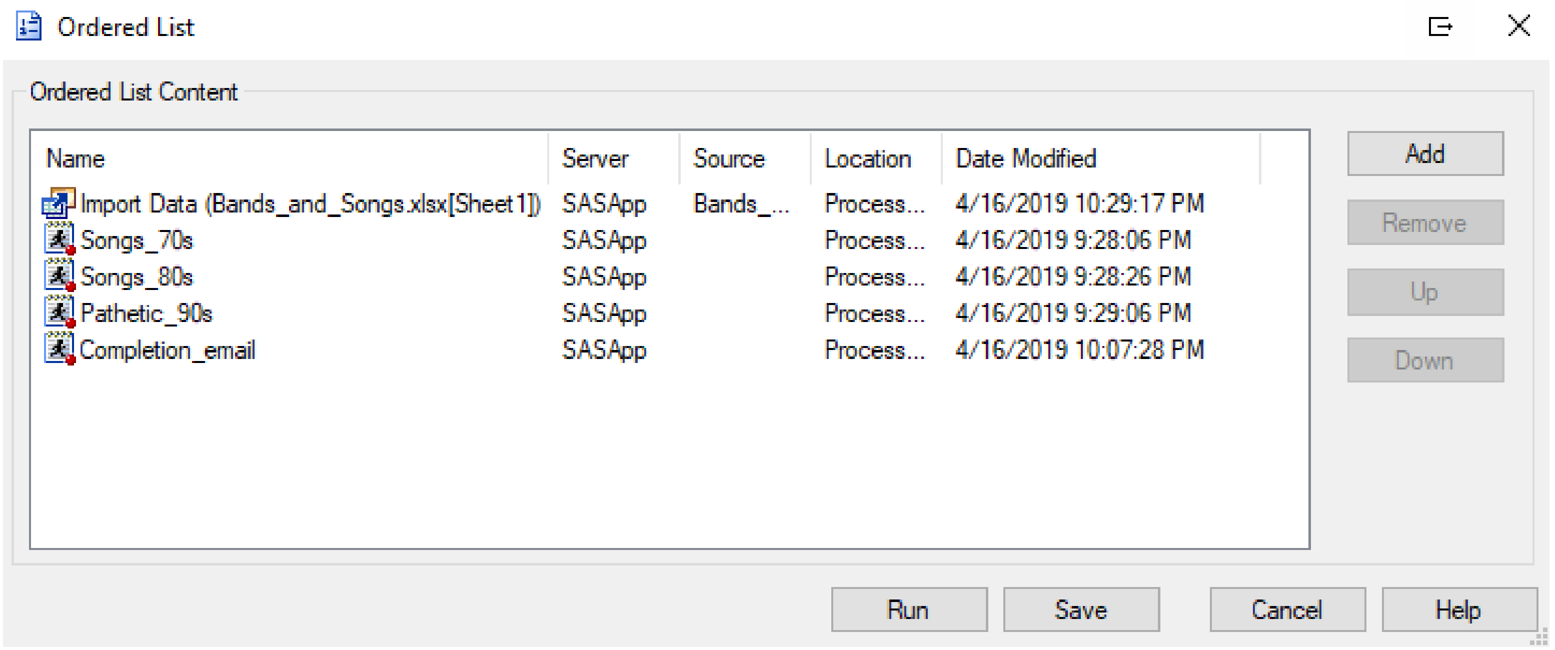
Name: Import Data (Bands_and_Songs.xlsx[genre])

Files of type: All File Types

Key Knowledge – Enterprise Guide

Ordered List

- Continue the same process and select the program/tasks you want to run
- Note that you can move them up or down, or even remove them from the list



Name	Server	Source	Location	Date Modified
Import Data (Bands_and_Songs.xlsx[Sheet1])	SASApp	Bands_...	Process...	4/16/2019 10:29:17 PM
Songs_70s	SASApp		Process...	4/16/2019 9:28:06 PM
Songs_80s	SASApp		Process...	4/16/2019 9:28:26 PM
Pathetic_90s	SASApp		Process...	4/16/2019 9:29:06 PM
Completion_email	SASApp		Process...	4/16/2019 10:07:28 PM

- Click on Save when you are finished

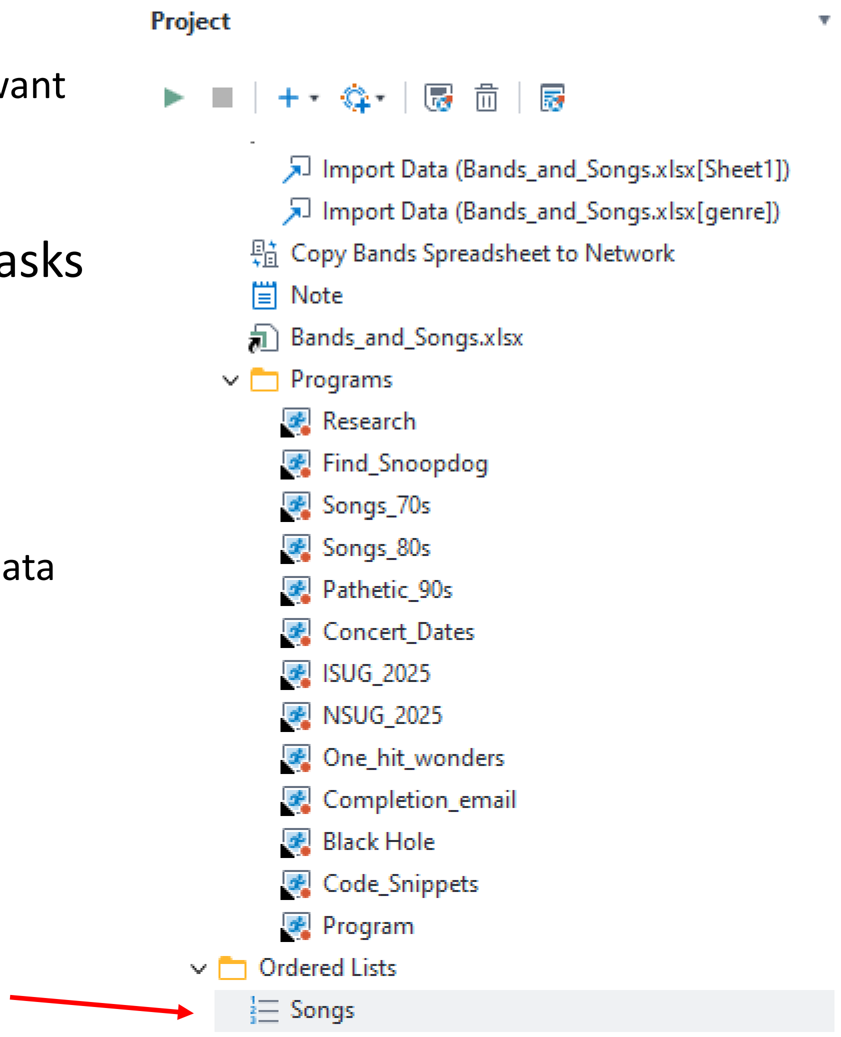
Key Knowledge – Enterprise Guide

Ordered List

- Right click on "OrderList" and rename if you want
- You can always right click and open to edit
- To run, right click and choose "Run Songs"
- And WaPow!, it will run the codes/tasks in your list

- You can create more than one Ordered List
- Say you have a bunch of "Import Data" tasks
- You can create an ordered list to just import data
- The possibilities are endless!

- Again, this was a "Game Changer" for me!



Project

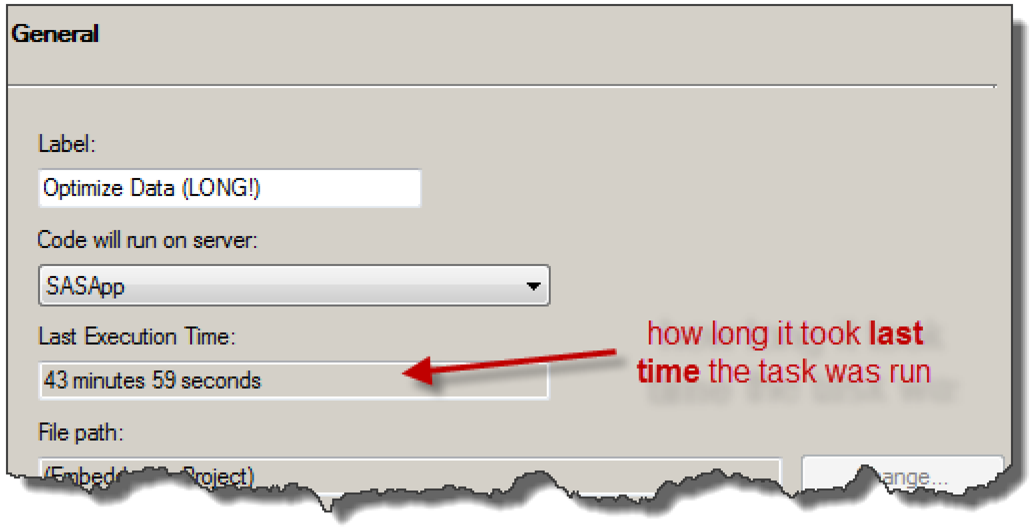
- Import Data (Bands_and_Songs.xlsx[Sheet1])
- Import Data (Bands_and_Songs.xlsx[genre])
- Copy Bands Spreadsheet to Network
- Note
- Bands_and_Songs.xlsx
- Programs
  - Research
  - Find_Snoopdog
  - Songs_70s
  - Songs_80s
  - Pathetic_90s
  - Concert_Dates
  - ISUG_2025
  - NSUG_2025
  - One_hit_wonders
  - Completion_email
  - Black Hole
  - Code_Snippets
  - Program
- Ordered Lists
  - Songs

Key Knowledge – Enterprise Guide

How much time will your process flow take to run?

It's difficult to predict how long a SAS process will take to run, as it depends more upon the data than on the actual program instructions. In SAS Enterprise Guide, you can find this information at the task level by the following:

1. Right-clicking on the task (or program node)
2. Selecting Properties
3. On the General tab, you'll see the "Last execution time".


General

Label:
Optimize Data (LONG!)

Code will run on server:
SASApp

Last Execution Time:
43 minutes 59 seconds

File path:

Key Knowledge & Tidbits

- SAS Dates, Datetime, Time
  - A SAS date is a number! It is the number of days from Jan 1, 1960 to a specified date. Some of us have negative birthday dates! Sad ☹
  - A Datetime is also a number! It is the number of seconds between Jan 1, 1960 and a specified date and time (down to the year/month/day/hour/minute/second)
  - And Time is a number. It is the number of seconds since midnight of the current day.
  - For each type of variable above, you will need to assign a format. I mainly use dates so my favorite format is mmddyy10. (i.e. 05/13/2019). Perform a Google search on "SAS date formats". Here's a useful link

https://documentation.sas.com/?docsetId=lrcon&docsetTarget=p1wj0wt2ebe2a0n1lv4lem9hdc0v.htm&docsetVersion=9.4&locale=en

Key Knowledge & Tidbits
Dates cont.

- Use ‘ddmmmyyyy’d to specify a date in SAS code (i.e.’01JAN2019’D)
  - Example: Where rock_induction_date >= ‘01Jan2018’d;
  - Or if using macro variables, I highly recommend that you include everything in the macro string (remember macro variables are just strings of characters).
  - %let compare_dt = ‘01Jan2018’d;
  - So the code would look like this: Where rock_induction_date >= &compare_dt;

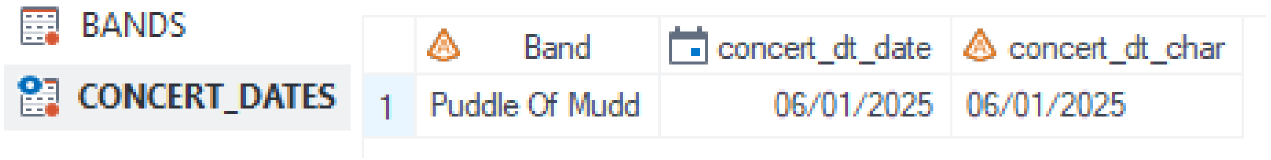
Key Knowledge & Tidbits

Date variable and Date text strings – NOT the same thing

Example:

```
data concert_dates;
set bands;
format concert_dt_date mmddyy10.;
if Band='Puddle Of Mudd' then do;
         concert_dt_date = '01jun2025'd;
         concert_dt_char = '06/01/2025';
end;
run;
```

OUTPUT:



BANDS
CONCERT_DATES

	Band	concert_dt_date	concert_dt_char
1	Puddle Of Mudd	06/01/2025	06/01/2025

Key tidbit: It is important to understand that the two concert date variables have to be handled accordingly in any future SAS processing. One is a <u>Date data type</u> and the other is a <u>Character data type</u> – even though they look the same in the output.

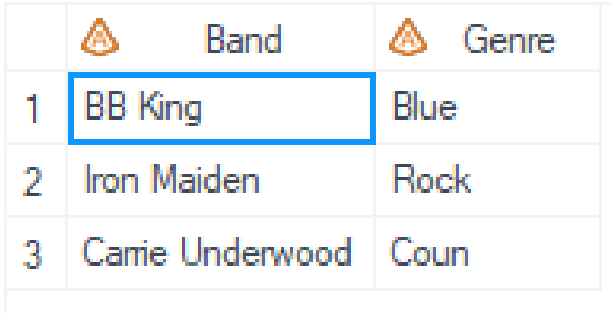
Key Knowledge & Tidbits

Length of variables – Important when using the data step (Not an issue when using proc SQL). Key tidbit: In Data Step programming: When you set up a character variable assignment, SAS will set the length of the variable on the first assignment.

If you do this:

```
data genre;
set bands2;
if Band = 'Iron Maiden' then Genre = 'Rock';
else if Band = 'Carrie Underwood' then Genre = 'Country';
else if Band = 'BB King' then Genre = 'Blues';
run;
```

Your output will look like this:



	Band	Genre
1	BB King	Blue
2	Iron Maiden	Rock
3	Carrie Underwood	Coun

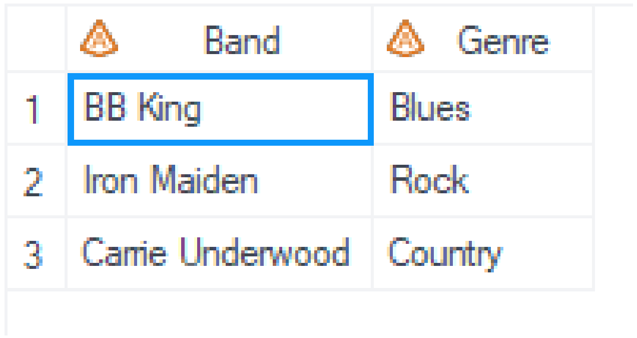
Key Knowledge & Tidbits

Length of variables – Use the length statement to avoid “character cut-off” (Note: the value of length needs to be greater or equal to the longest assignment length)

<u>To fix the previous issue, do this:</u>

```
data genre;
set bands2;
length Genre $8.;
if Band = 'Iron Maiden' then Genre = 'Rock';
else if Band = 'Carrie Underwood' then Genre = 'Country';
else if Band = 'BB King' then Genre = 'Blues';
run;
```

<u>Your output will look like this:</u>



	Band	Genre
1	BB King	Blues
2	Iron Maiden	Rock
3	Carrie Underwood	Country

Programming Tips

Assignment Logic audit checking

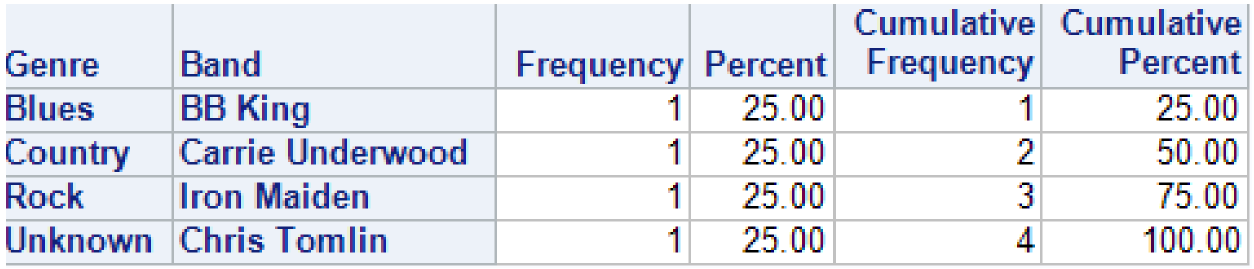
When you make assignment "logic statements" in your code, especially with complex logic assignments, ALWAYS, and I say ALWAYS run a "Proc Freq" to check your results

Back to Genre … it's a good practice to add "Unknown" at the end of a logic assignment and do a proc freq to check if you captured everything.

```
data genre2;
set bands3;
length Genre $8.;
if Band = 'Iron Maiden' then Genre = 'Rock';
else if Band = 'Carrie Underwood' then Genre = 'Country';
else if Band = 'BB King' then Genre = 'Blues';
else Genre = 'Unknown';
run;

/* Check Assignments */
proc freq data = genre2; tables Genre*Band / missing list; run;
```

Output (oops! You have something that didn't get assigned! - It's 'Christian' by the way):



Genre	Band	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Blues	BB King	1	25.00	1	25.00
Country	Carrie Underwood	1	25.00	2	50.00
Rock	Iron Maiden	1	25.00	3	75.00
Unknown	Chris Tomlin	1	25.00	4	100.00

Programming Tips

Functions I love:

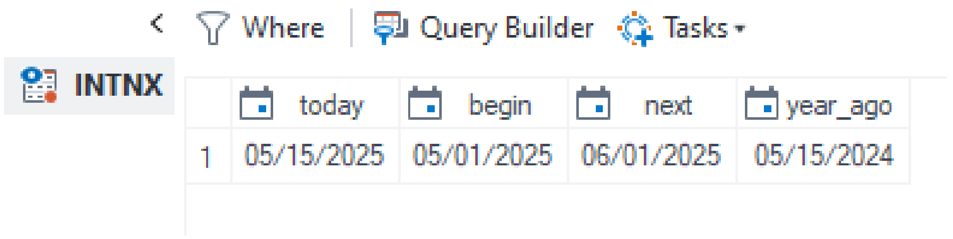
Intnx – to return the date after a specific number of intervals have passed. I like to think of it as "date math"

Example1: take today's date and set the date as the first day of the current month
Example2: take today's date and set the date as the first day of the next month
Example3: take today's date and go back to the same day a year ago

Note: I ran this code on 5/15/2025.

```
data intnx;
format today begin next year_ago mmddyy10.;
today = today();
begin = intnx('month',today,0);
next = intnx('month',today,1);
year_ago = intnx('day',today,-365);
run;
```

Output:



	today	begin	next	year_ago
1	05/15/2025	05/01/2025	06/01/2025	05/15/2024

Programming Tips

Putting it all together with CALL SYMPUTX and INTNX

Say your dashboard month is April, 2019

```
%let dashmonth = 201904; /* Note: this is the only input parameter you need */

data Test;
format month_begin today prior_month mmddyy10.;
month = substr("&dashmonth",5,2);
year =  substr("&dashmonth",1,4);
month_begin = mdy(month,1,year);
today = today();
prior_month = INTNX('MONTH',Today(),-1);
call symputx('month_begin_SQL',"'"||strip(year)||'-'||strip(month)||'-01'||"'");
call symputx('month_begin_SAS',"'"||put(month_begin,date9.)||"'"||'D');
call symputx('month_next_SQL',"'"||put(intnx('month',month_begin,1),yymmdd10.)||"'");
call symputx('month_next_sas',"'"||put(intnx('month',month_begin,1),date9.)||"'"||'D');
call symputx'today',put(today,year4.)||put(month(today),z2.)||put(day(today),z2.));
call symputx("prior_month",put(prior_month,YYMMN.));
run;

%put dashmonth = &dashmonth;
%put prior_month = &prior_month;
%put month_begin_SQL = &month_begin_SQL;
%put month_begin_SAS = &month_begin_SAS;
%put month_next_SQL = &month_end_SQL;
%put month_next_SAS = &month_end_SAS;
%put today = &today;
```

Programming Tips

Putting it all together with CALL SYMPUTX and INTNX

Log output

```
%put dashmonth = &dashmonth;
dashmonth = 201904

%put prior_month = &prior_month;
prior_month = 201903

%put month_begin_SQL = &month_begin_SQL;
month_begin_SQL = '2019-04-01'

%put month_begin_SAS = &month_begin_SAS;
month_begin_SAS = '01APR2019'D

%put month_next_SQL = &month_end_SQL;
month_next_SQL = '2019-05-01'

%put month_next_SAS = &month_end_SAS;
month_next_SAS = '01MAY2019'D

%put today = &today;
today = 20190430
```

Be creative and you can become a SAS Rock Star, oh yeah!

Questions?

Contact Info:
Jeff LaMar
Jeffrey.c.lamar@wellsfargo.com

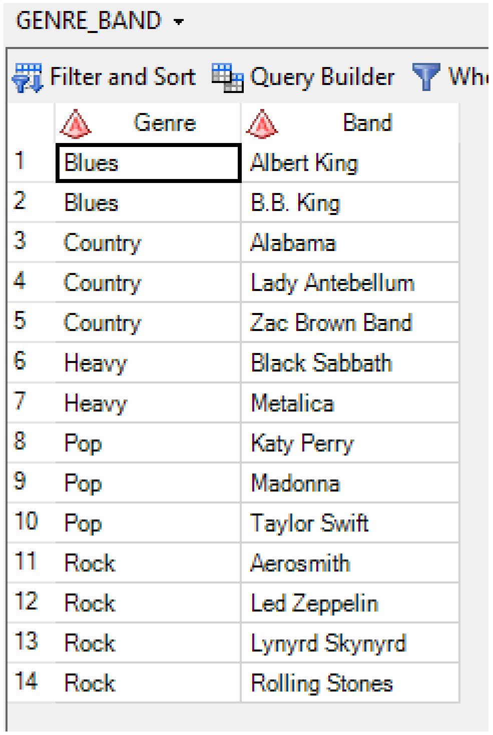
Appendix

Code Snippets

Counter Code

Purpose: To create counter variable numbers for by groups within a dataset
Note: This can be very useful for many different purposes. Keep this code handy in your "code snippet" file.

Example1: Incoming Dataset



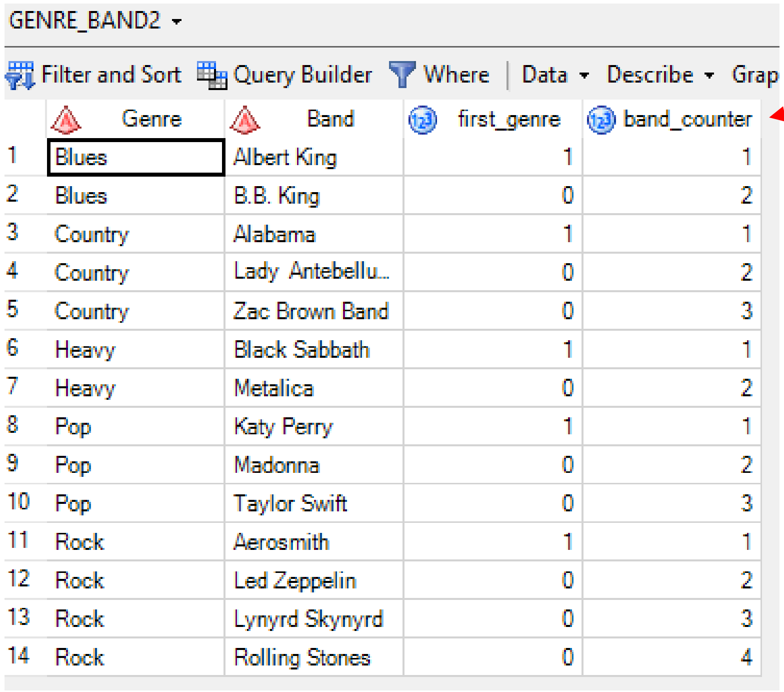
	Genre	Band
1	Blues	Albert King
2	Blues	B.B. King
3	Country	Alabama
4	Country	Lady Antebellum
5	Country	Zac Brown Band
6	Heavy	Black Sabbath
7	Heavy	Metalica
8	Pop	Katy Perry
9	Pop	Madonna
10	Pop	Taylor Swift
11	Rock	Aerosmith
12	Rock	Led Zeppelin
13	Rock	Lynyrd Skynyrd
14	Rock	Rolling Stones

Objective
Create some "counter" variables for further logic processing

Code Snippets

Counter Code

```
*** Counter with ONE by group;
*** Increment the counter for each band in the Genre;
*** Reset the counter to 1 at a new Genre;
data genre_band2;
set genre_band;
first_genre = first.genre; /*optional:so you can see the first.genre variable*/
band_counter + 1; /* Note: This is a “sum statement” */
by Genre;
if first.Genre then band_counter = 1;
Run;
```



	Genre	Band	first_genre	band_counter
1	Blues	Albert King	1	1
2	Blues	B.B. King	0	2
3	Country	Alabama	1	1
4	Country	Lady Antebellu...	0	2
5	Country	Zac Brown Band	0	3
6	Heavy	Black Sabbath	1	1
7	Heavy	Metalica	0	2
8	Pop	Katy Perry	1	1
9	Pop	Madonna	0	2
10	Pop	Taylor Swift	0	3
11	Rock	Aerosmith	1	1
12	Rock	Led Zeppelin	0	2
13	Rock	Lynyrd Skynyrd	0	3
14	Rock	Rolling Stones	0	4

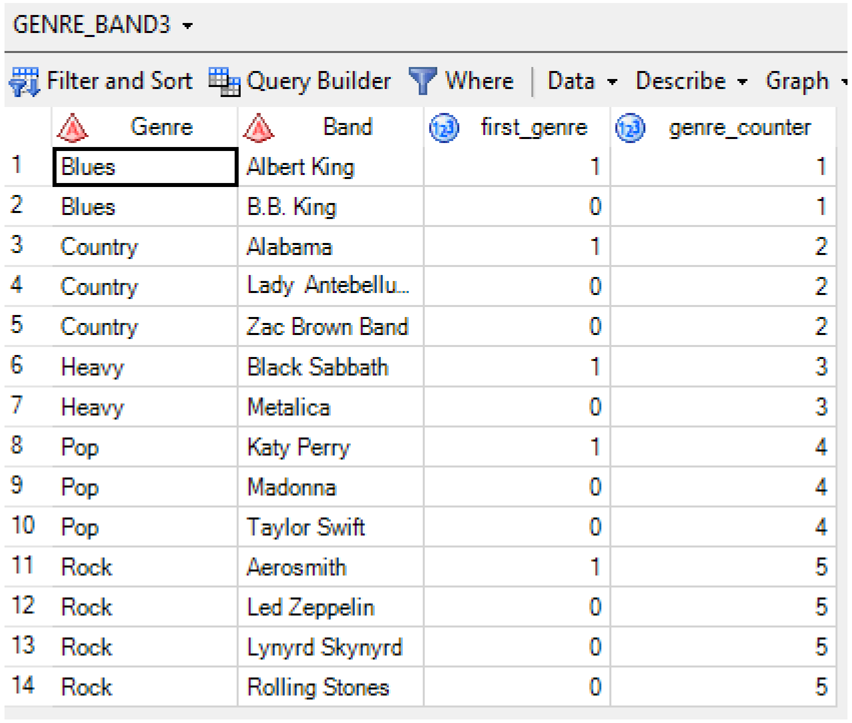
Output

Note that the band_counter increments with each change in Genre (and resets back to 1)

Code Snippets

Counter Code

```
*** Counter with ONE by group;
*** Increment the counter for each change of Genre;
data genre_band3;
set genre_band;
first_genre = first.genre; /* optional */
by Genre;
if first.Genre then genre_counter + 1; /* This is a "Sum Statement */
run;
```



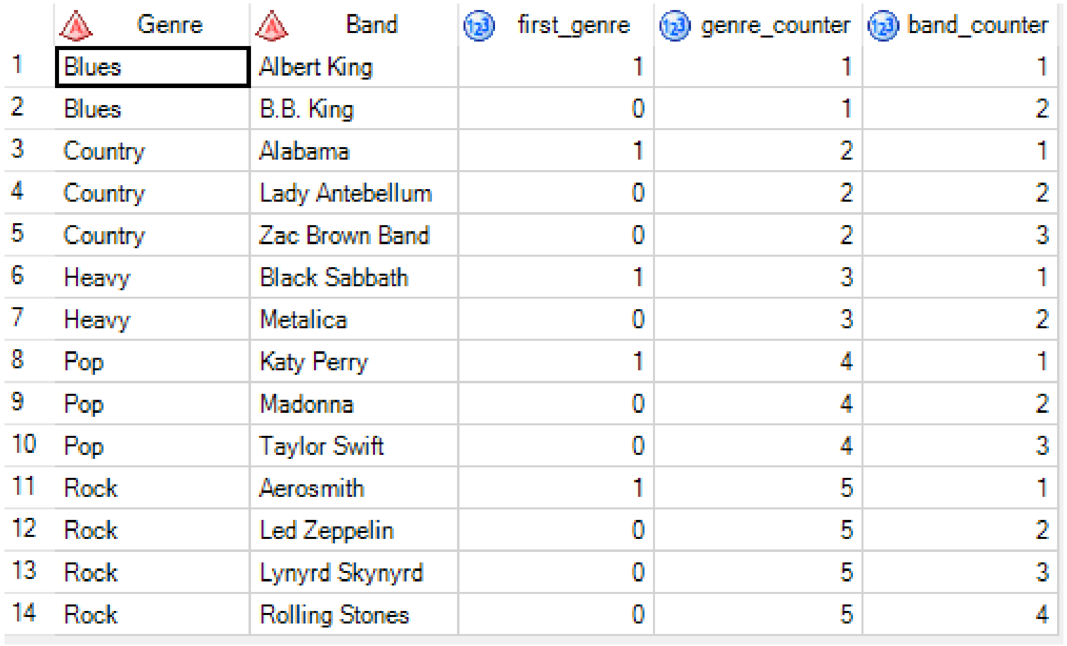
	Genre	Band	first_genre	genre_counter
1	Blues	Albert King	1	1
2	Blues	B.B. King	0	1
3	Country	Alabama	1	2
4	Country	Lady Antebellu...	0	2
5	Country	Zac Brown Band	0	2
6	Heavy	Black Sabbath	1	3
7	Heavy	Metalica	0	3
8	Pop	Katy Perry	1	4
9	Pop	Madonna	0	4
10	Pop	Taylor Swift	0	4
11	Rock	Aerosmith	1	5
12	Rock	Led Zeppelin	0	5
13	Rock	Lynyrd Skynyrd	0	5
14	Rock	Rolling Stones	0	5

Output
Note that the genre_counter increments with each change in Genre.

Code Snippets

<u>Counter Code</u>

```
*** Putting both together;
data genre_band4;
set genre_band;
first_genre = first.genre; /* optional: just so you can see the first.genre
variable */
retain genre_counter 0;
band_counter + 1;
by Genre;
if first.Genre then do;
         band_counter = 1;
         genre_counter + 1;
end;
run;
```



	Genre	Band	first_genre	genre_counter	band_counter
1	Blues	Albert King	1	1	1
2	Blues	B.B. King	0	1	2
3	Country	Alabama	1	2	1
4	Country	Lady Antebellum	0	2	2
5	Country	Zac Brown Band	0	2	3
6	Heavy	Black Sabbath	1	3	1
7	Heavy	Metalica	0	3	2
8	Pop	Katy Perry	1	4	1
9	Pop	Madonna	0	4	2
10	Pop	Taylor Swift	0	4	3
11	Rock	Aerosmith	1	5	1
12	Rock	Led Zeppelin	0	5	2
13	Rock	Lynyrd Skynyrd	0	5	3
14	Rock	Rolling Stones	0	5	4

Output
Both counters in one dataset

Code Snippets

Email Template

Purpose: To send an email after the completion of a code run. Include final condition code for a quick assessment of the run.

Part 1: Beginning and middle of code

```
*****************************************************************;
*** Enterprise Guide email template;
*****************************************************************;
*** Send email after job completion which includes condition codes;
*** At the top of the code, reset the condition codes;

*** Reset condition codes for email output;
%let syscc = 0; /* SASGRID1 */

*** Get the time the SAS program started;
%let timenow=%sysfunc(datetime(),datetime.);

*** Your SAS code goes here;
*** Example below;
data genre_band2;
set genre_band;
first_genre = first.genre;
band_counter + 1;
by Genre;
if first.Genre then band_counter = 1;
Run;
```

Code Snippets

Email Template

Purpose: To send an email after the completion of a code run. Include final condition code for a quick assessment of the run.

Part 2: Put at end of program

```
*** Send email after program is finished;
*** Get the time the SAS program completed;
%let timeend=%sysfunc(datetime(),datetime.);
%put timeend = &timeend;

*** Calculate minutes;
data _null_;
seconds=intck('seconds',"&timenow."dt,"&timeend."dt);
minutes = seconds/60.0;
call symput('minutes',strip(put(minutes,8.1)));
run;

filename OUTBOX email
TO=("&useremail")
FROM=("&useremail")
SUBJECT="Enterprise Guide Program Completed. Condition Code = &syscc";

data _null_;
        file outbox;
        put "Enterprise Guide Program Completed";
        put / "SASGRID SYSCC Error Code is: &syscc";
        put / "SAS program started at: &timenow";
        put / "SAS program ended at : &timeend";
        put / "SAS total run time: &minutes ";
        put / '**This email is generated from an automated SAS job 
process**';
run;
```

Code Snippets

<u>Email Template that checks condition code and sends accordingly</u>

Purpose: To check the condition code and send an email to the appropriate audience based on the status of the job run.

A note on &syscc (A system Automatic Macro Variable):

- If &syscc=0 then no errors and no warnings
- If &syscc=4 then no errors but at least one warning
- If &syscc=1012 or 3000 then ERRORS in code (anything above a 4 is an error!)

```
%macro send_email;
%if &syscc. <= 4 %then %do;

    filename OUTBOX email
    TO=(‘buspartner1.yourcompany.com’ ‘busparter2.yourcompany.com’)
    FROM=("&useremail")
    SUBJECT="Enterprise Guide Program Completed";

    data _null_;
        file outbox;
        put "Enterprise Guide Program Completed";
        put / '**This email is generated from an automated SAS job
    process**';
    run;

%end;
```

Code Snippets

Email Template that checks condition code and sends accordingly
Purpose: To check the condition code and send an email to the appropriate audience based on the status of the job run.

```
%else %do;

    filename OUTBOX email
    TO=("&useremail")
    FROM=("&useremail")
    SUBJECT="Enterprise Guide Program Completed with ERRORS.
    Condition Code = &syscc";

    data _null_;
        file outbox;
        put "Enterprise Guide Program Completed with ERRORS";
        put / "SASGRID SYSCC Error Code is: &syscc";
        put / "SAS program started at: &timenow";
        put / "SAS program ended at : &timeend";
        put / "SAS total run time: &minutes ";
        put / '**This email is generated from an automated SAS job
    process**';
    run;

%end;

%mend send_email;

%send_email;
```