How to build custom macros: A step-by-step approach to translating your SAS® code into a custom macro

Sara Richter, MS
MWSUG Annual Conference 2024
#BB044

If your work is anything like mine, your to do list is at least this long and you find yourself doing very similar tasks again and again and again. That might include similar data processing, similar data summaries or analysis either within a dataset or for a new dataset. {This example is the SAS® Help Heart dataset.)

	Status	Cause of Death	Age CHD Diagnosed	Sex	Age at Start	Cholesterol Status	Blood Pressure Status	Weight Status	Smoking Status
1	Dead	Other	.	Female	29		Normal	Overweight	Non-smoker
2	Dead	Cancer	.	Female	41	Desirable	High	Overweight	Non-smoker
3	Alive		.	Female	57	High	High	Overweight	Moderate (6-15)
4	Alive		.	Female	39	High	Normal	Overweight	Non-smoker
5	Alive		.	Male	42	High	Optimal	Overweight	Heavy (16-25)
6	Alive		.	Female	58	Desirable	High	Overweight	Non-smoker
7	Alive		.	Female	36	Desirable	Normal	Overweight	Moderate (6-15)
8	Dead	Other	.	Male	53	High	Normal	Normal	Non-smoker
9	Alive		.	Male	35	Borderline	Normal	Overweight	Non-smoker
10	Dead	Cerebral Vascular Disease	.	Male	52	High	Normal	Normal	Light (1-5)
11	Alive		.	Male	39	Borderline	Normal	Overweight	Very Heavy (> 25)
12	Alive		57	Male	33	Borderline	Optimal	Overweight	Non-smoker
13	Alive		55	Male	33	Desirable	High	Overweight	Non-smoker
14	Alive		79	Male	57		Normal	Overweight	Moderate (6-15)
15	Alive		66	Male	44	High	High	Normal	Very Heavy (> 25)
16	Alive		.	Female	37	Desirable	Normal	Normal	Moderate (6-15)
17	Alive		.	Male	40	Desirable	Normal	Overweight	Very Heavy (> 25)
18	Dead	Cancer	56	Male	56	Desirable	Normal	Underweight	Moderate (6-15)
19	Alive		.	Female	42	Borderline	High	Overweight	Light (1-5)
20	Dead	Coronary Heart Disease	74	Male	46	Borderline	High	Overweight	Very Heavy (> 25)
21	Alive		.	Female	37	Desirable	Optimal	Overweight	Moderate (6-15)
22	Alive		.	Female	45	Borderline	Normal	Overweight	Light (1-5)

Then you get another dataset and need to do more of the same. {This example is the SAS Help Heart dataset.)

```
PROC TABULATE DATA=SASHELP.Heart;
      CLASS Sex Status;
      TABLE (Sex ALL='Total'),(Status ALL='Total')
            *(N='Count' COLPCTN='Col %'
            ROWPCTN='Row %');
RUN;
```

	Status						Total		
	Alive			Dead					
	Count	Col %	Row %	Count	Col %	Row %	Count	Col %	Row %
Sex									
Female	1977	61.44	68.81	896	45.00	31.19	2873	55.15	100.00
Male	1241	38.56	53.13	1095	55.00	46.88	2336	44.85	100.00
Total	3218	100.00	61.78	1991	100.00	38.22	5209	100.00	100.00

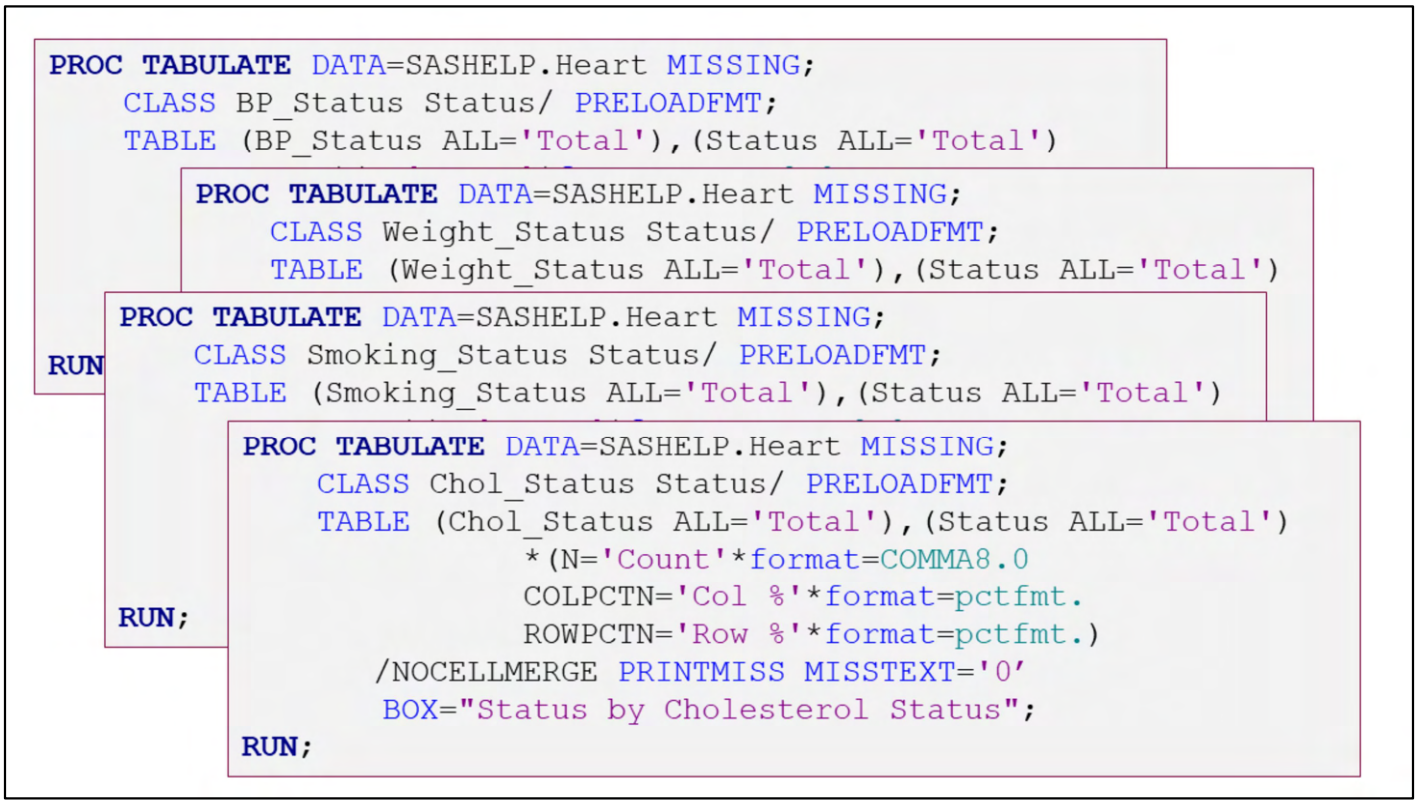
You're tasked with looking at the crosstab tables between status and demographic variables. You know your collaborator likes to have n's and percentages in their own columns so you use a PROC TABULATE. The result is OK, but not quite right.

```
PROC FORMAT;
      PICTURE pctfmt (ROUND) low-high="009%";
RUN;
PROC TABULATE DATA=SASHELP.Heart MISSING;
    CLASS Sex Status/ PRELOADFMT;
    TABLE (Sex ALL='Total'),(Status ALL='Total')
```

Status by Sex	Status						Total		
	Alive			Dead					
	Count	Col %	Row %	Count	Col %	Row %	Count	Col %	Row %
Sex									
Female	1,977	61%	69%	896	45%	31%	2,873	55%	100%
Male	1,241	39%	53%	1,095	55%	47%	2,336	45%	100%
Total	3,218	100%	62%	1,991	100%	38%	5,209	100%	100%

```
RUN;
```

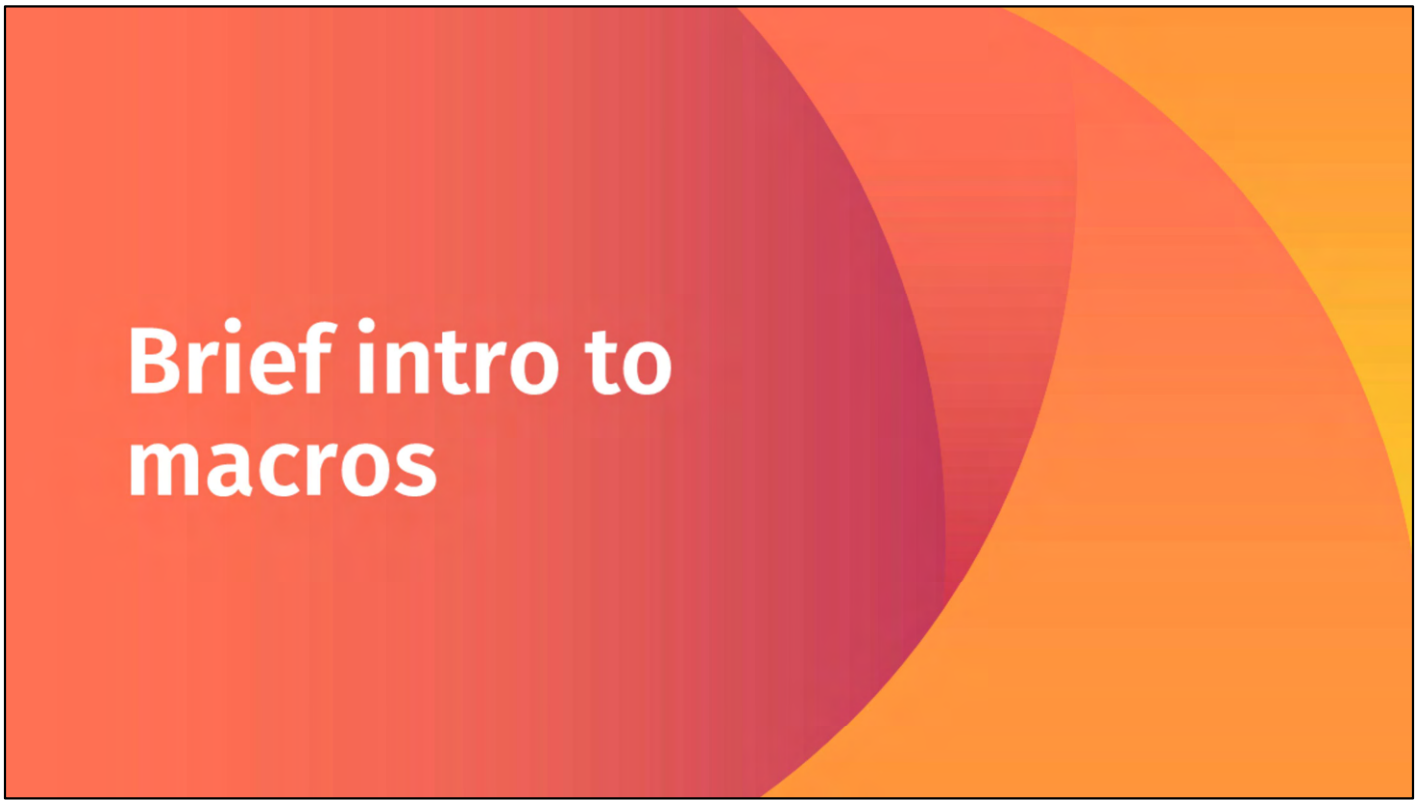
You keep tweaking the code until you get a results table you like. Here I've modified the formats a bit and added a label in the box. You're satisfied enough you proceed.



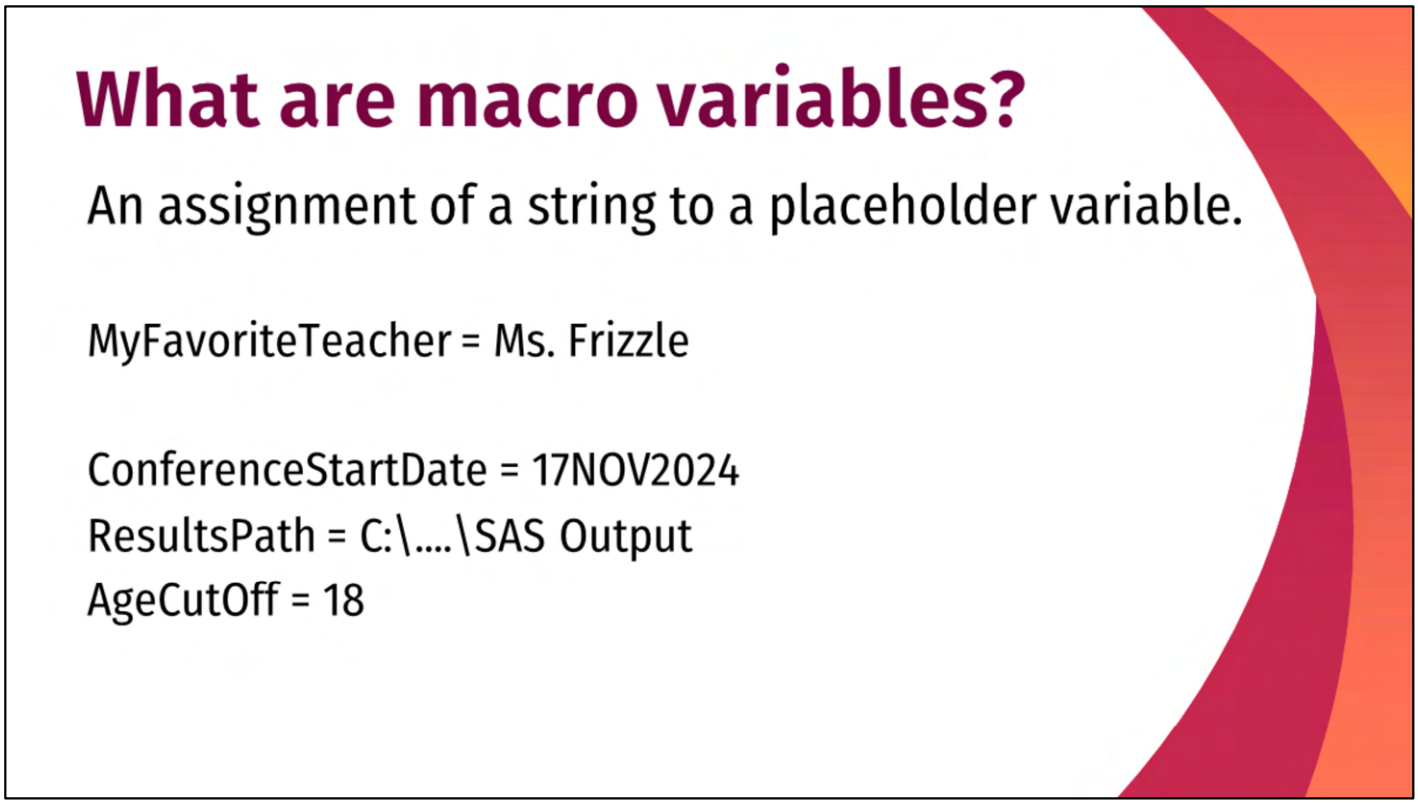
Then you copy/paste, change the variable names, and repeat again and again. Pretty soon your syntax file is 1,000 lines of code and it's difficult to navigate.

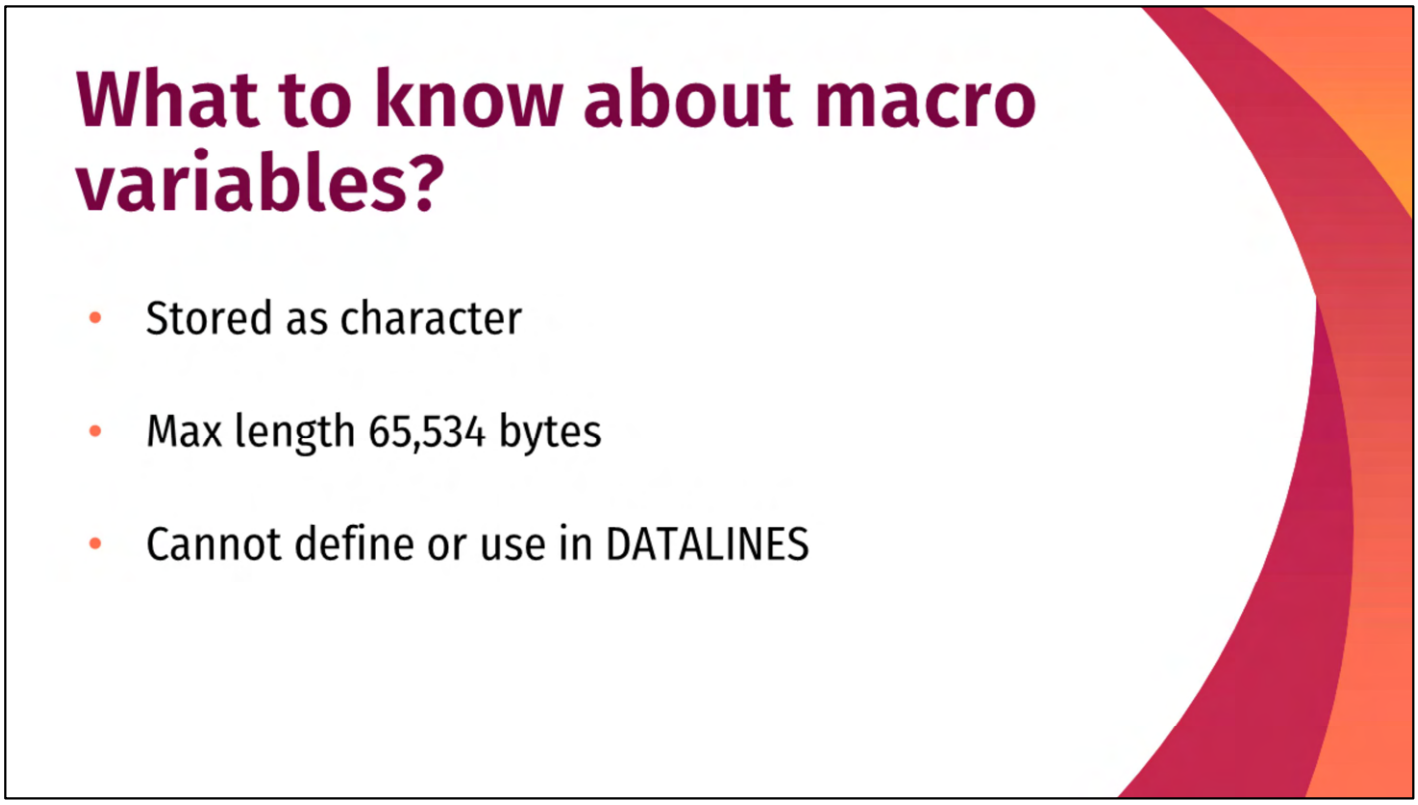
You can see that there must be a more efficient way, that it probably involves a macro, but how? And where to start?

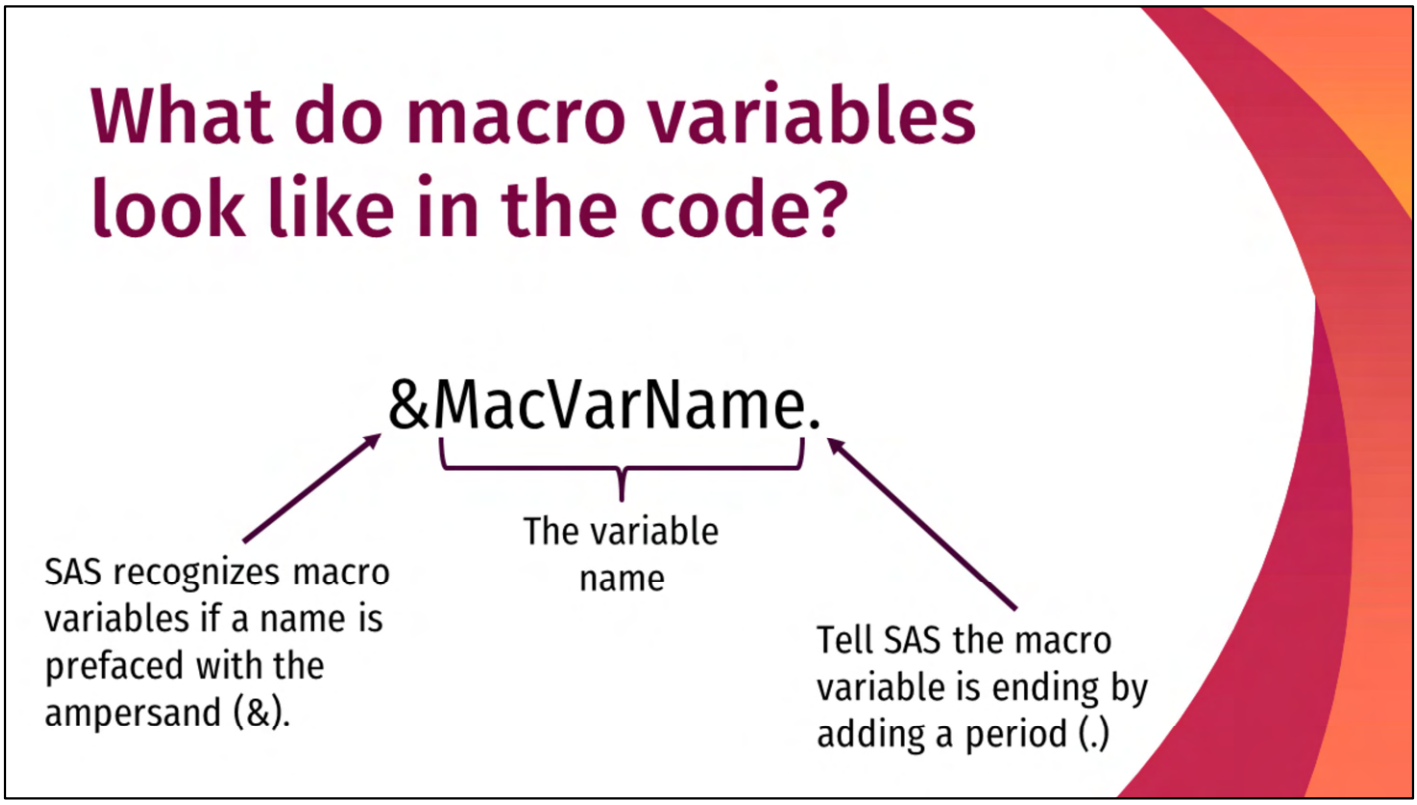
That's what we'll talk about today.

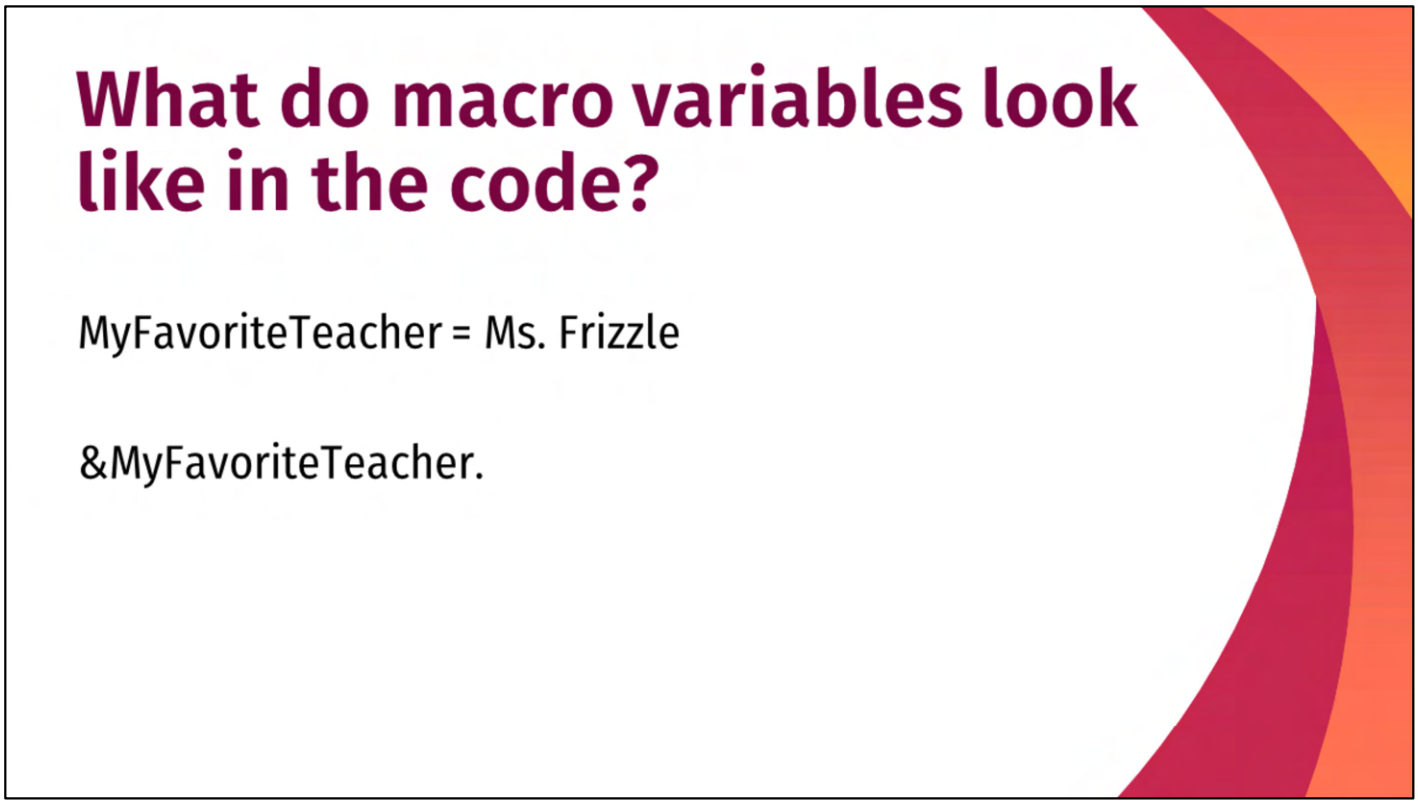


Let’s get started!


What are macro variables?
An assignment of a string to a placeholder variable.
MyFavoriteTeacher = Ms. Frizzle
ConferenceStartDate = 17NOV2024
ResultsPath = C:\....\SAS Output
AgeCutOff = 18


What to know about macro variables?
Stored as character
Max length 65,534 bytes
Cannot define or use in DATALINES

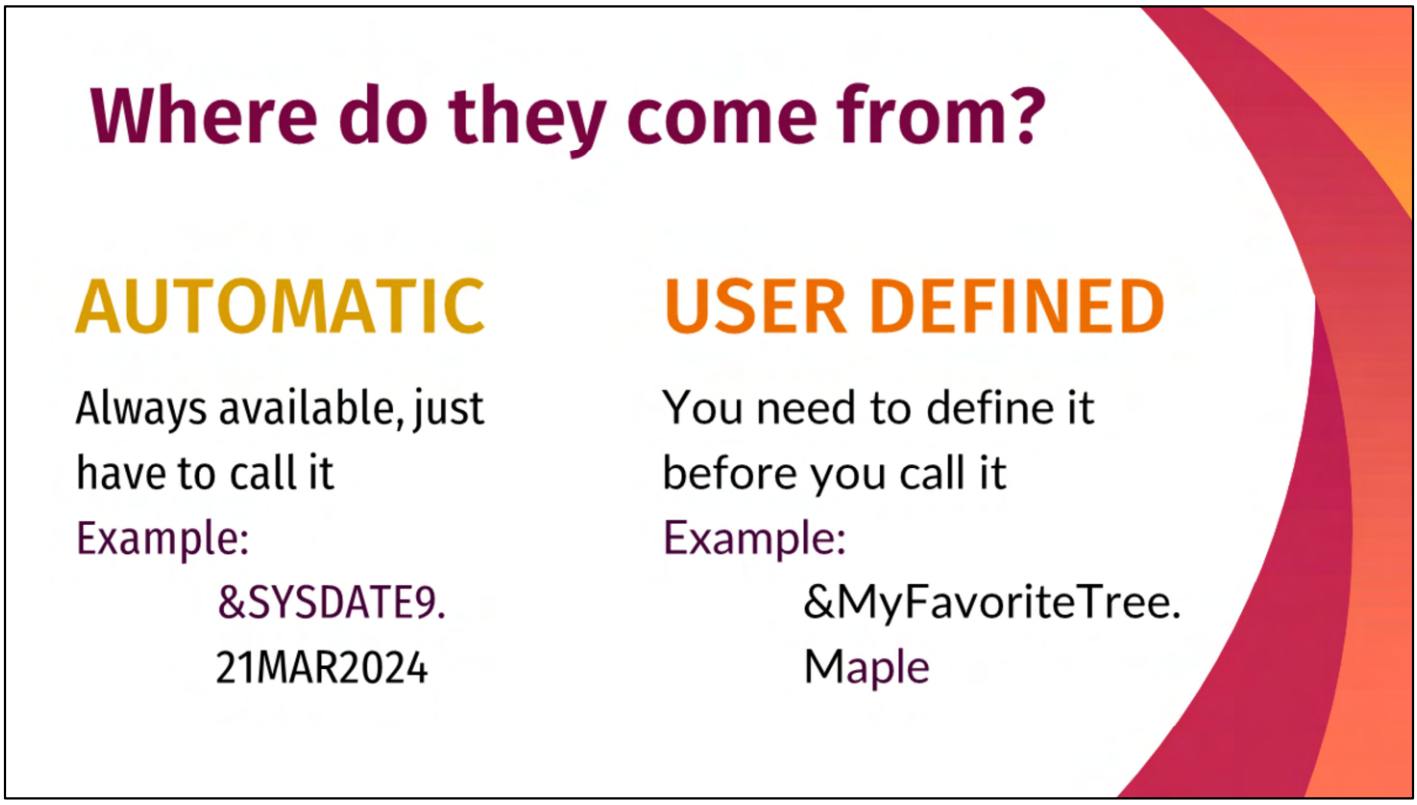

What do macro variables look like in the code?
&MacVarName.
SAS recognizes macro variables if a name is prefaced with the ampersand (&).
The variable name
Tell SAS the macro variable is ending by adding a period (.)

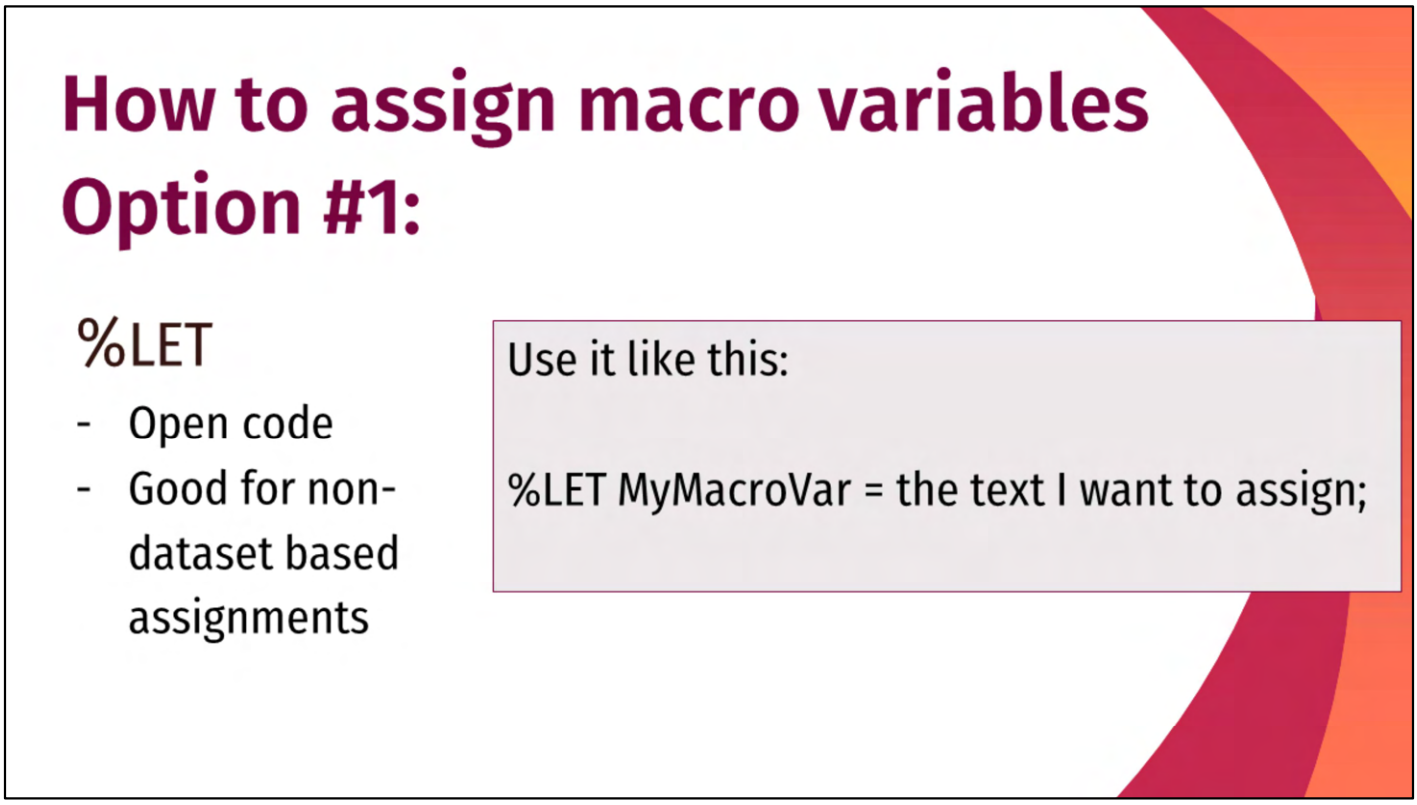

What do macro variables look like in the code?
MyFavoriteTeacher = Ms. Frizzle
&MyFavoriteTeacher.

What do macro variables look like in the code?

MyFavoriteTeacher = Ms. Frizzle
&MyFavoriteTeacher.

TodaysDate = 17NOV2024	→ &TodaysDate.
ResultsPath = C:\....\SAS Output	→ &ResultsPath.
AgeCutOff = 18	→ &AgeCutOff.


Where do they come from?
AUTOMATIC
Always available, just have to call it
Example:
&SYSDATE9.
21MAR2024
USER DEFINED
You need to define it before you call it
Example:
&MyFavoriteTree.
Maple


How to assign macro variables
Option #1:
%LET
- Open code
- Good for non-dataset based assignments
Use it like this:
%LET MyMacroVar = the text I want to assign;

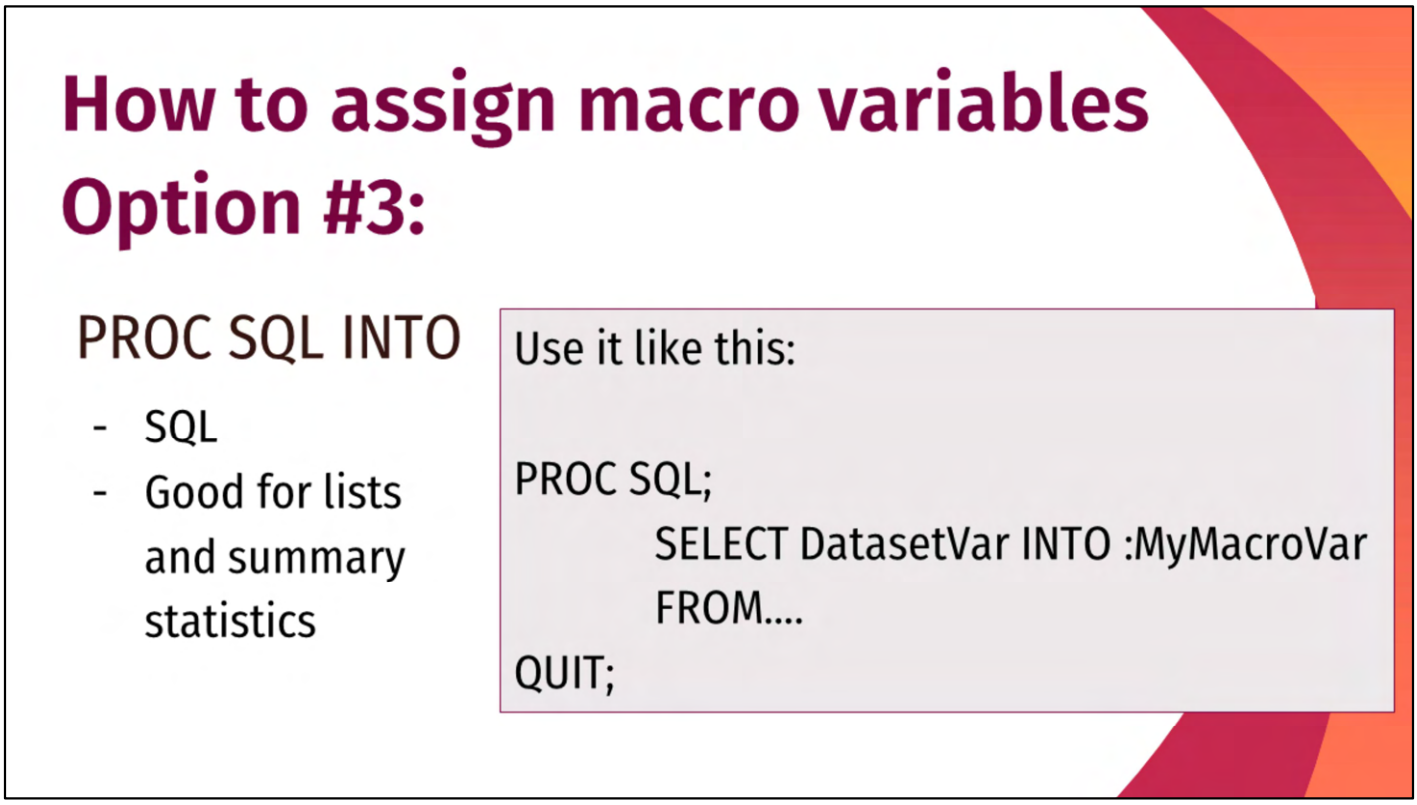
How to assign macro variables Option #2:

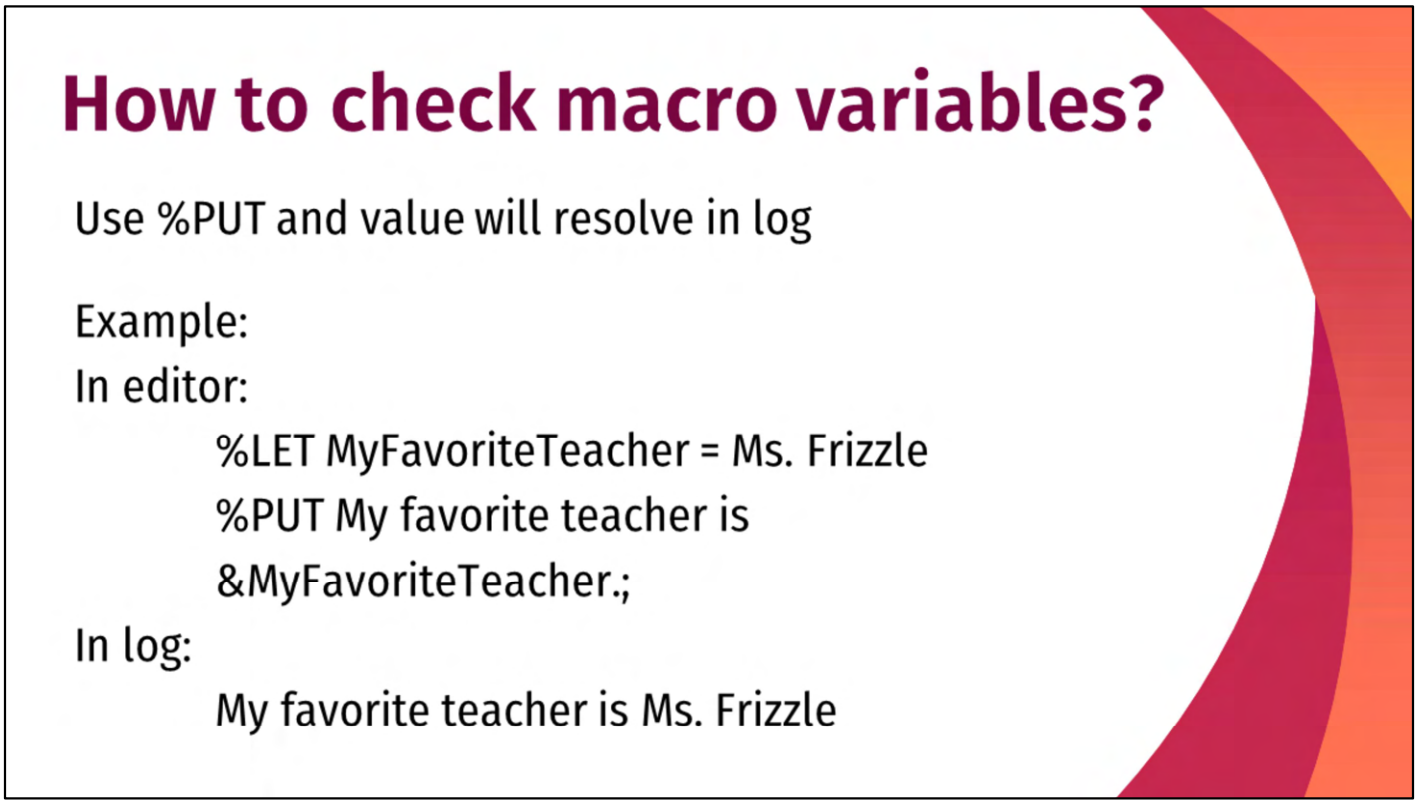
CALL SYMPUT

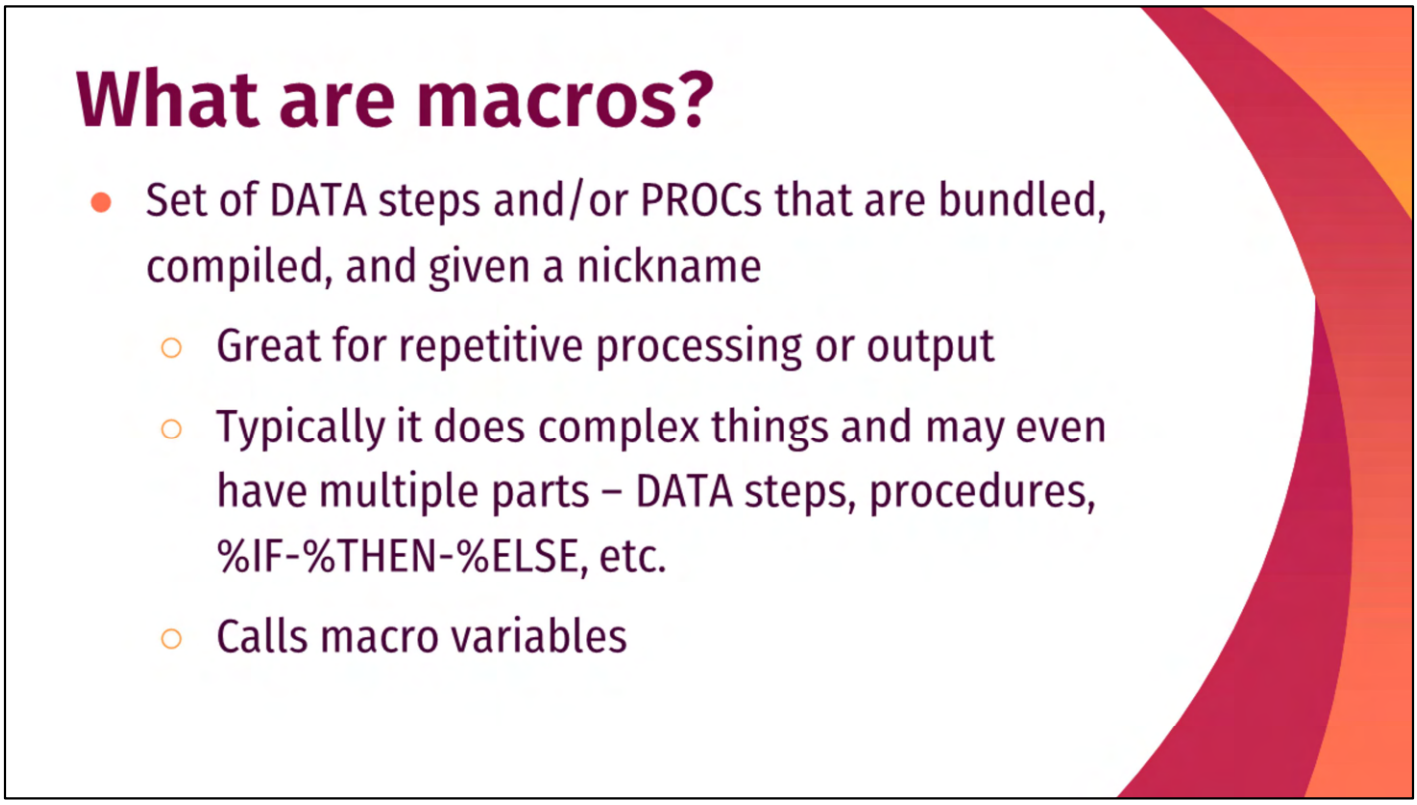
- Within DATA step
- Good for setting based on dataset value

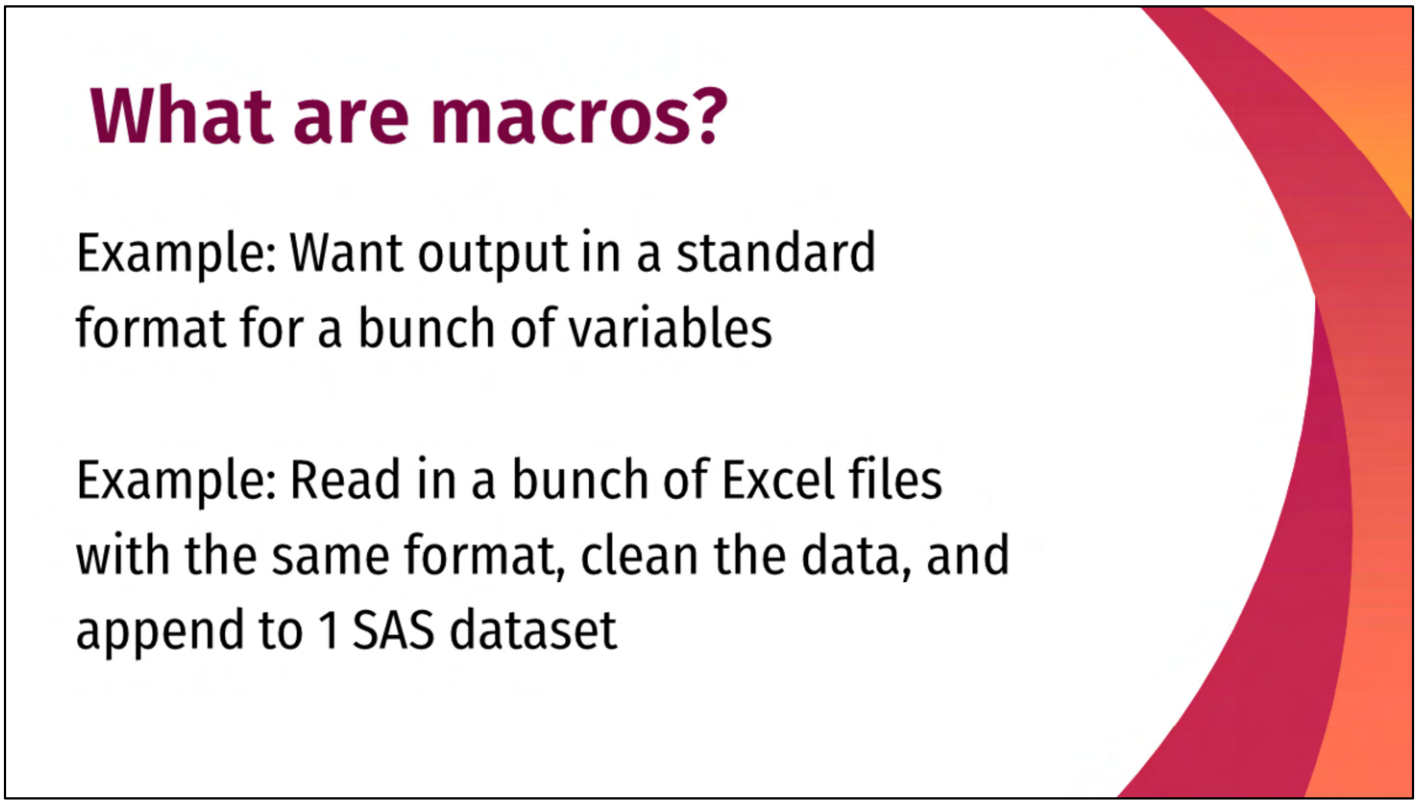
Use it like this:

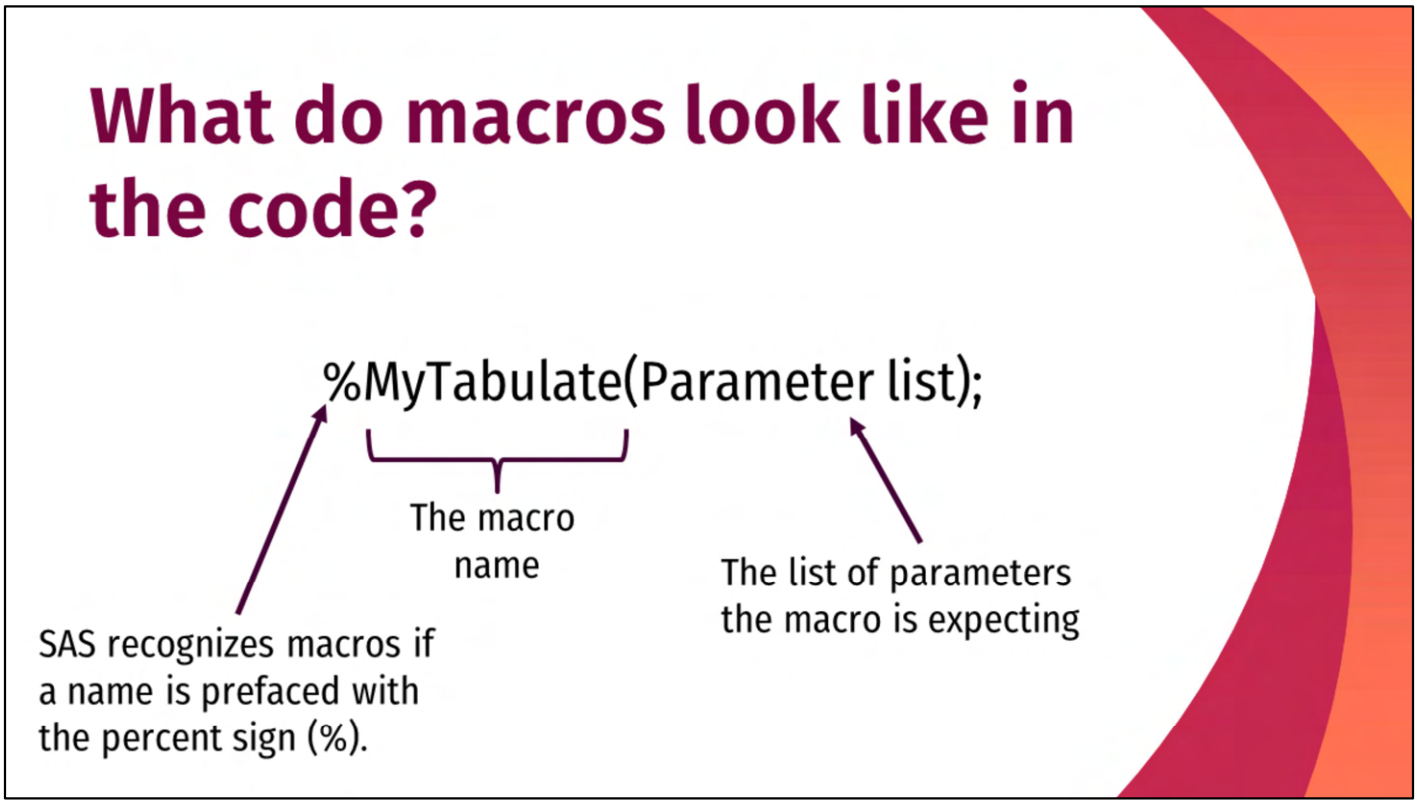
```
DATA ....;
...
CALL SYMPUT("MyMacroVar",  DatasetVar);
...
RUN;
```

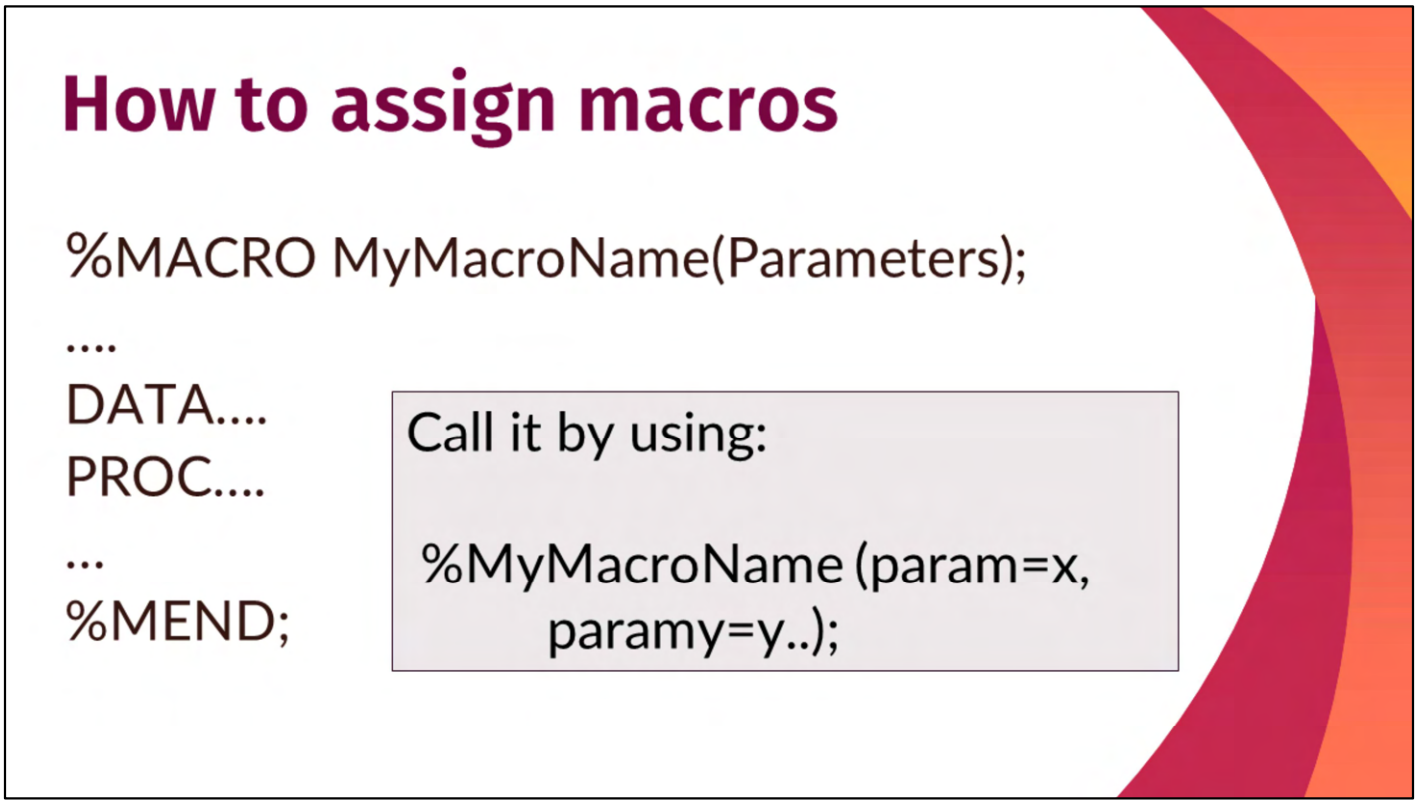

How to assign macro variables
Option #3:
PROC SQL INTO
- SQL
- Good for lists and summary statistics
Use it like this:
PROC SQL;
SELECT DatasetVar INTO :MyMacroVar
FROM....
QUIT;

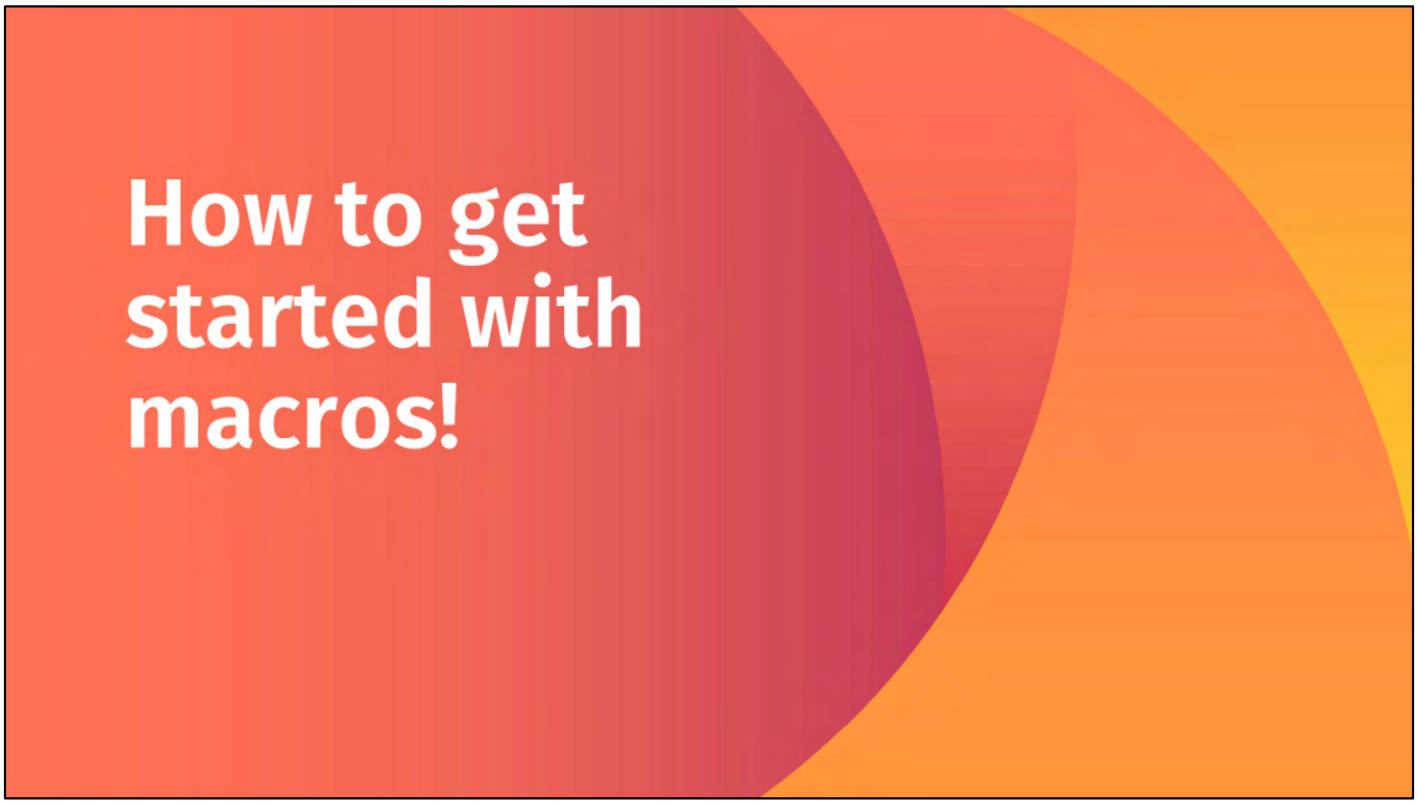

How to check macro variables?
Use %PUT and value will resolve in log
Example:
In editor:
%LET MyFavoriteTeacher = Ms. Frizzle
%PUT My favorite teacher is
&MyFavoriteTeacher.;
In log:
My favorite teacher is Ms. Frizzle


What are macros?
Set of DATA steps and/or PROCs that are bundled, compiled, and given a nickname
Great for repetitive processing or output
Typically it does complex things and may even have multiple parts – DATA steps, procedures, %IF-%THEN-%ELSE, etc.
Calls macro variables


What are macros?
Example: Want output in a standard format for a bunch of variables
Example: Read in a bunch of Excel files with the same format, clean the data, and append to 1 SAS dataset


What do macros look like in the code?
%MyTabulate(Parameter list);
The macro name
The list of parameters the macro is expecting
SAS recognizes macros if a name is prefaced with the percent sign (%).

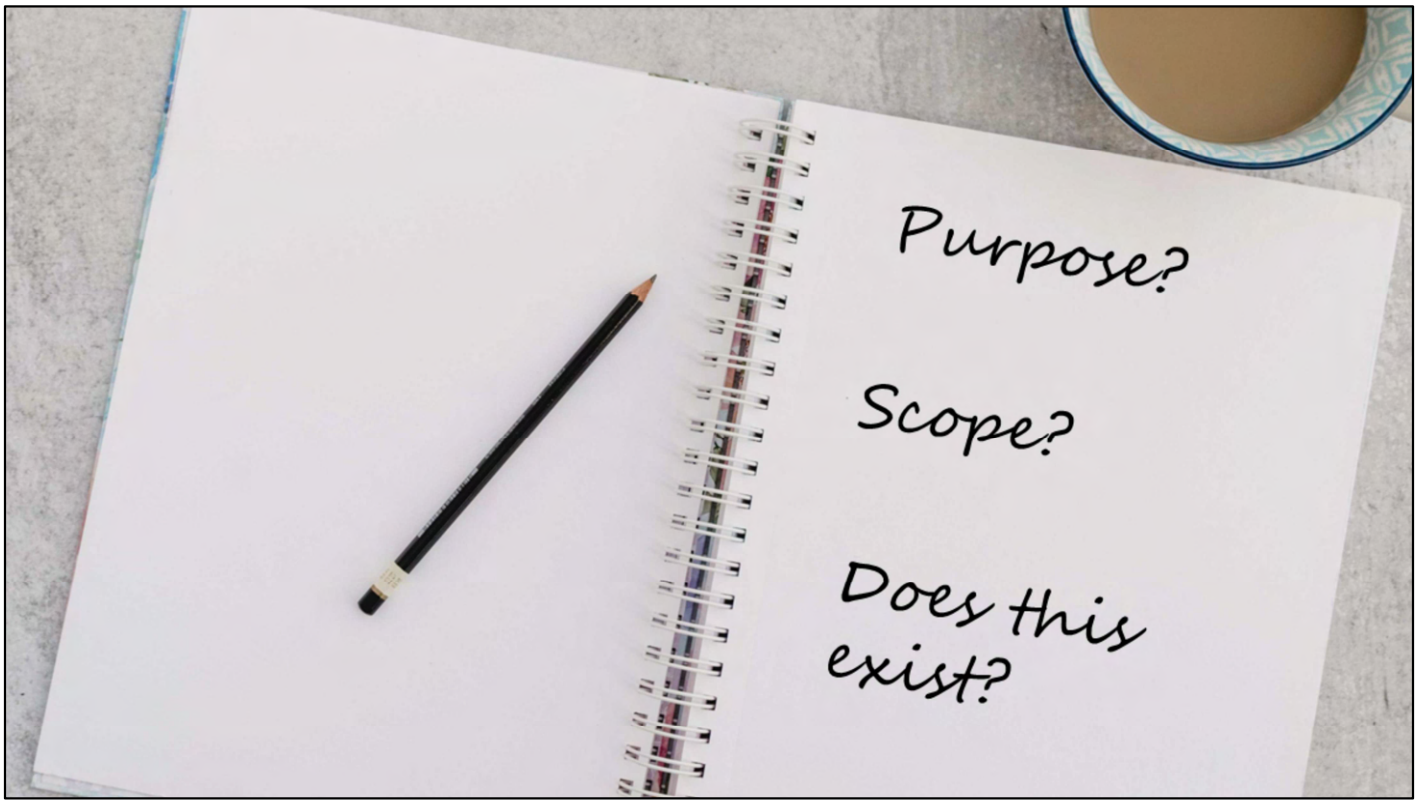

How to assign macros
%MACRO MyMacroName(Parameters);
....
DATA....
PROC....
...
%MEND;
Call it by using:
%MyMacroName (param=x,
paramy=y..);



So how do I implement these strategies in SAS? Let’s look through my toolbox.

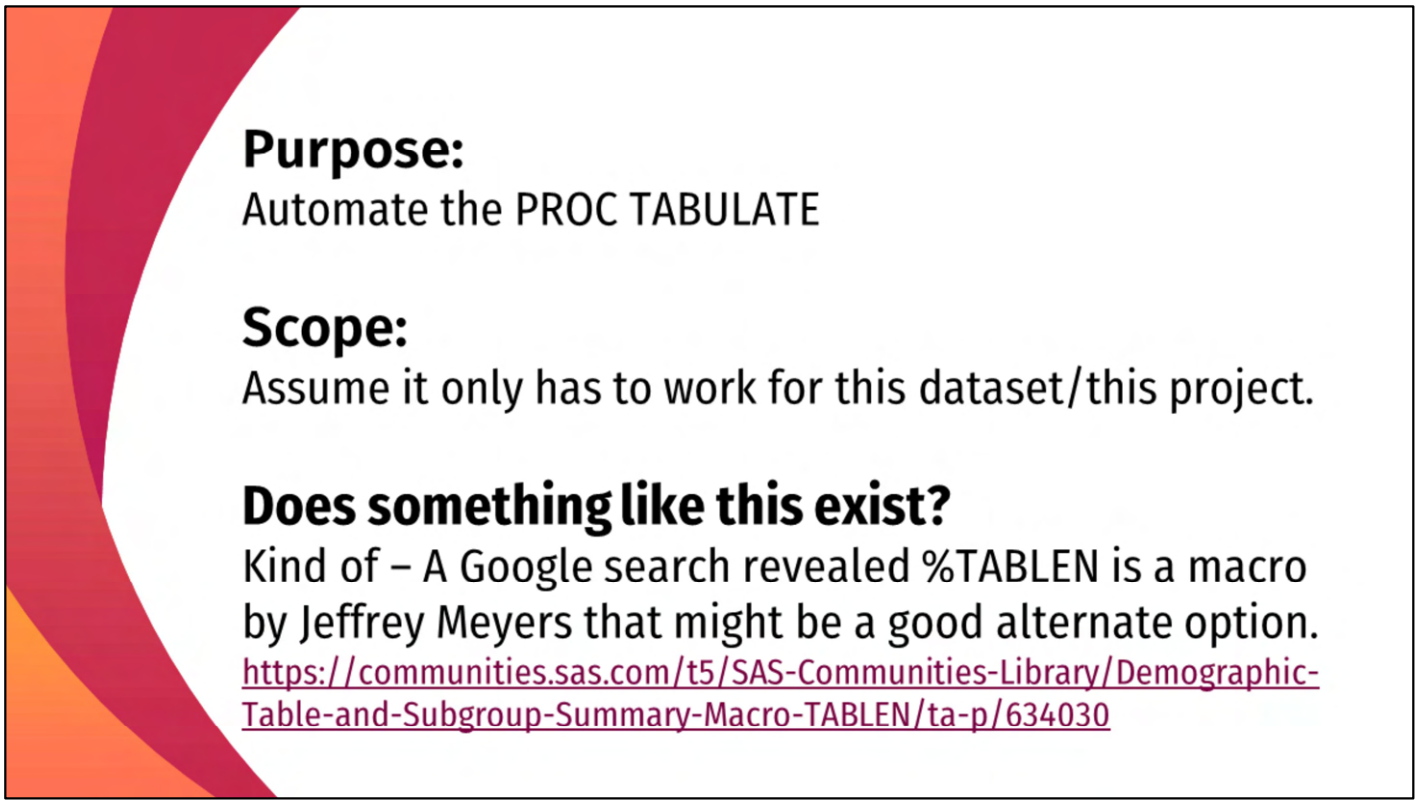
Status by Sex	Status						Total		
	Alive			Dead					
	Count	Col %	Row %	Count	Col %	Row %	Count	Col %	Row %
Sex									
Female	1,977	61%	69%	896	45%	31%	2,873	55%	100%
Male	1,241	39%	53%	1,095	55%	47%	2,336	45%	100%
Total	3,218	100%	62%	1,991	100%	38%	5,209	100%	100%

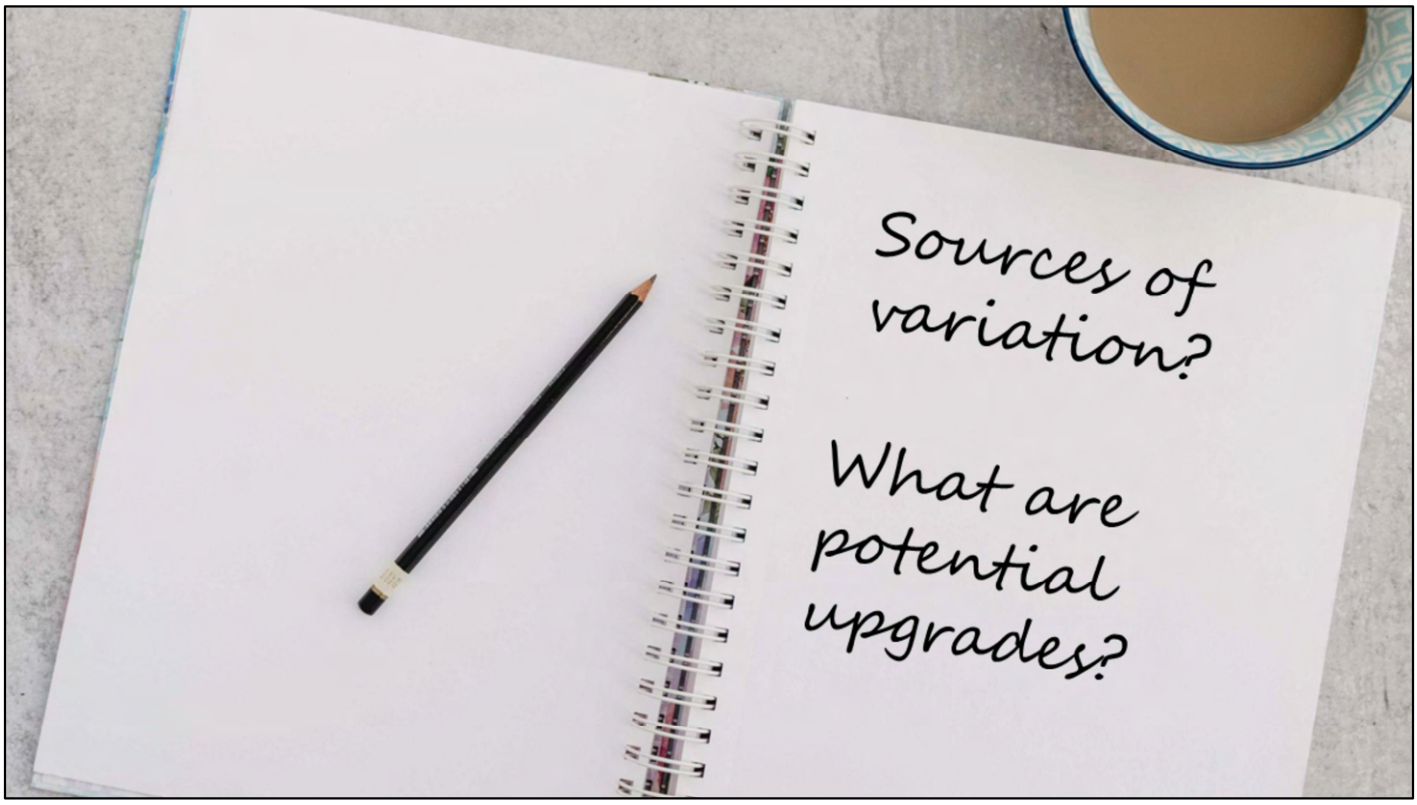
We're repeating this table.



Before we really dig in, there are some things to consider:

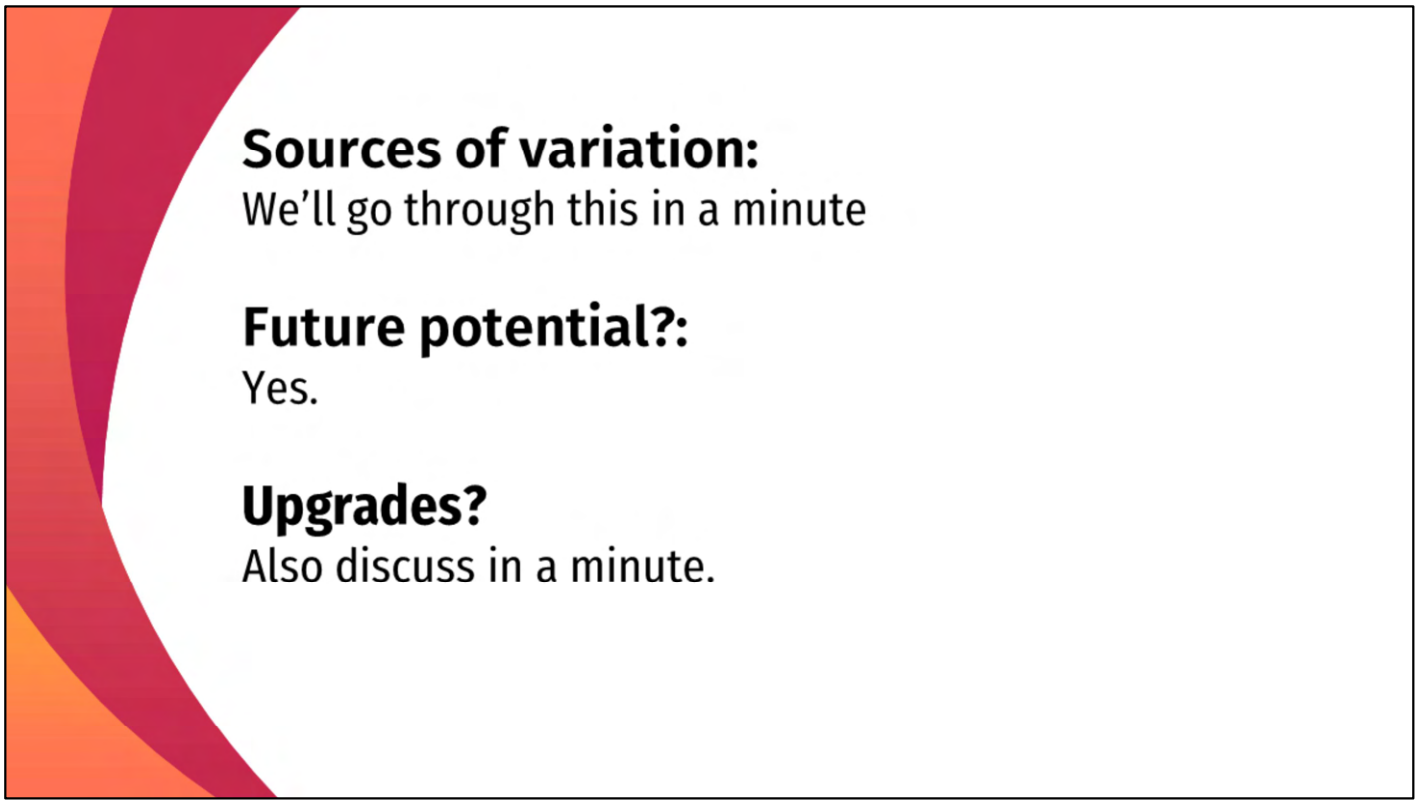
- What is the purpose of the macro? What do we want it to do? What do we want to do outside of the macro?
- What is the scope of the macro? Does it need to work with a lot of different datasets? Does it only need to do this one thing for the immediate project?
- Does something like this already exist?

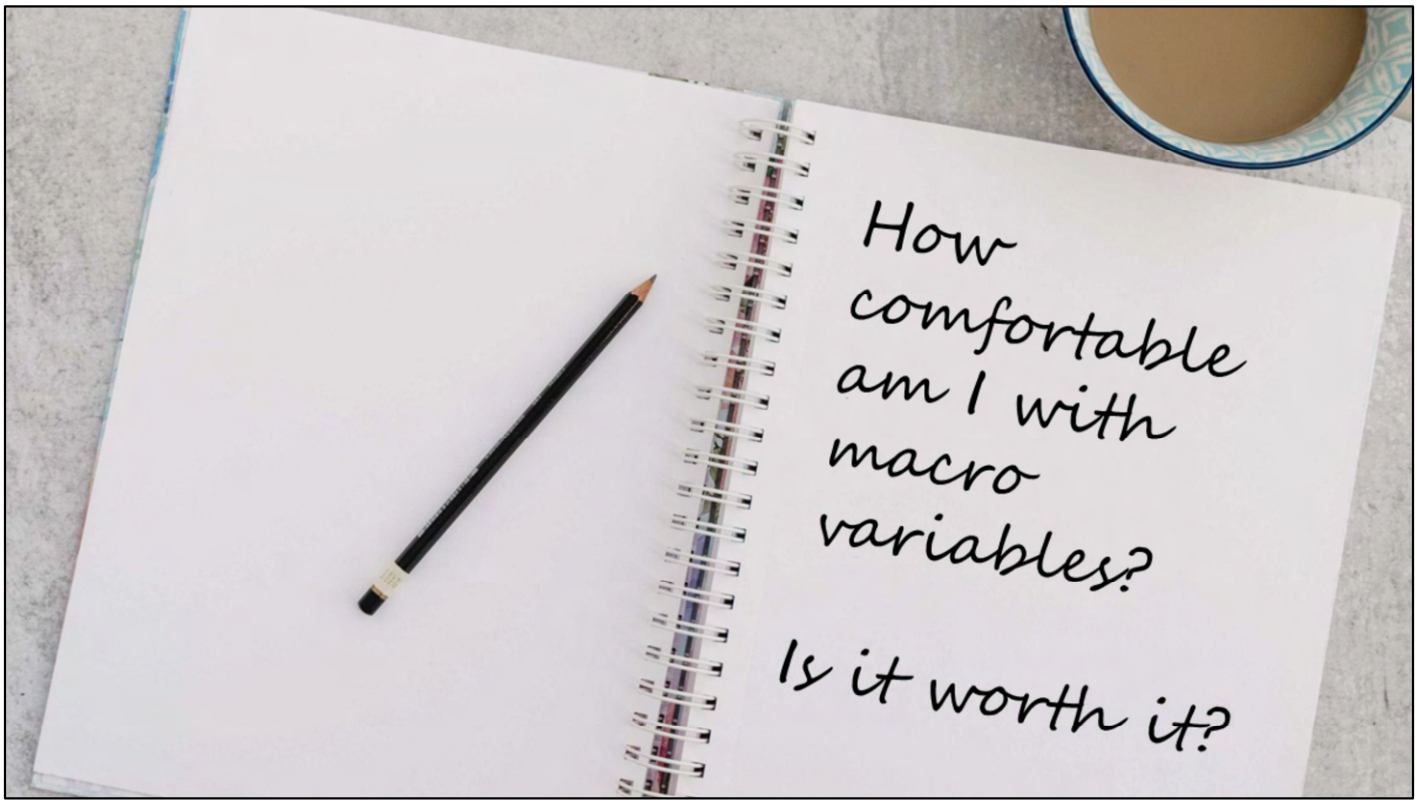

Purpose:
Automate the PROC TABULATE
Scope:
Assume it only has to work for this dataset/this project.
Does something like this exist?
Kind of – A Google search revealed %TABLEN is a macro by Jeffrey Meyers that might be a good alternate option.
https://communities.sas.com/t5/SAS-Communities-Library/Demographic-Table-and-Subgroup-Summary-Macro-TABLEN/ta-p/634030



What are the sources of variation that need to be accounted for?

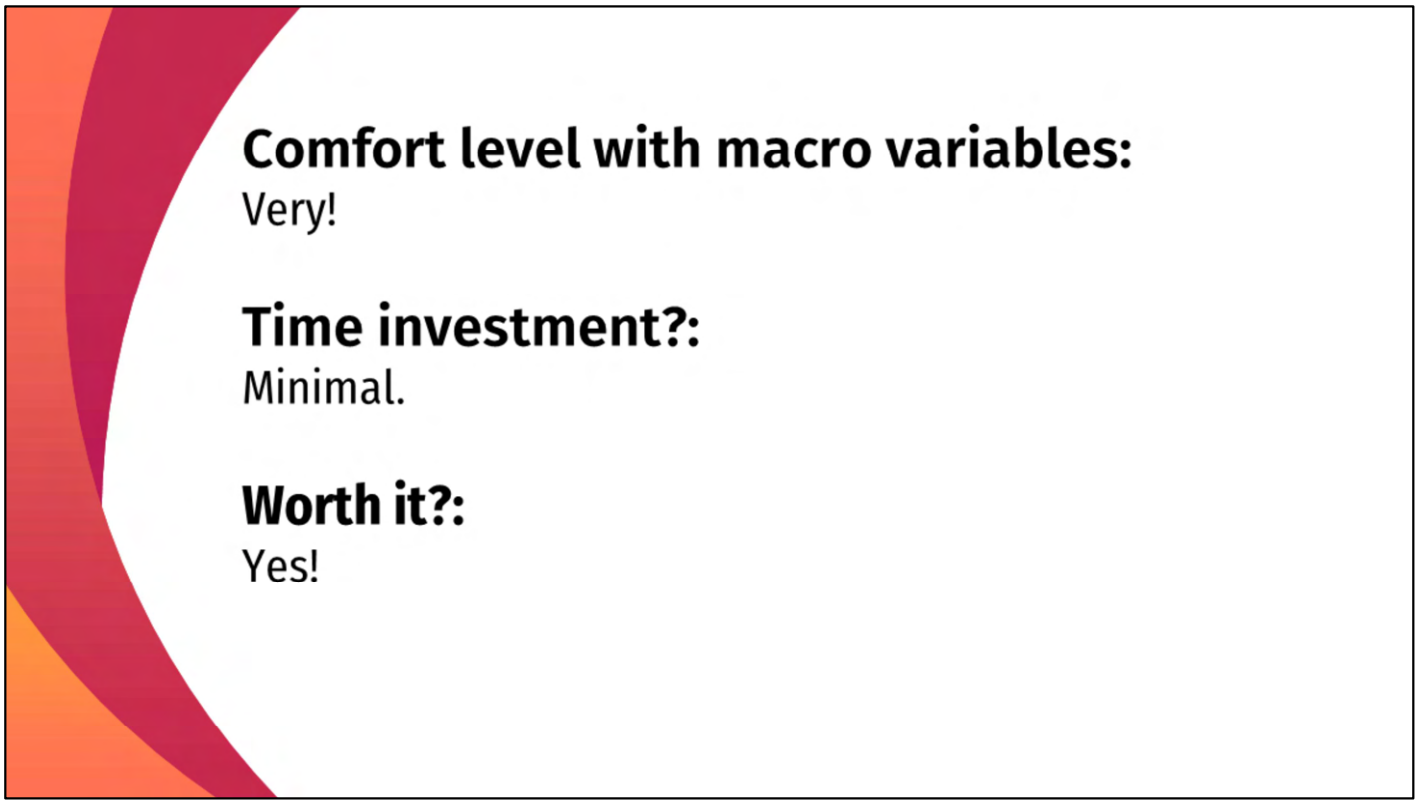
Does it have potential to be used beyond this project? If so, what are the potential upgrades that we may want to add?

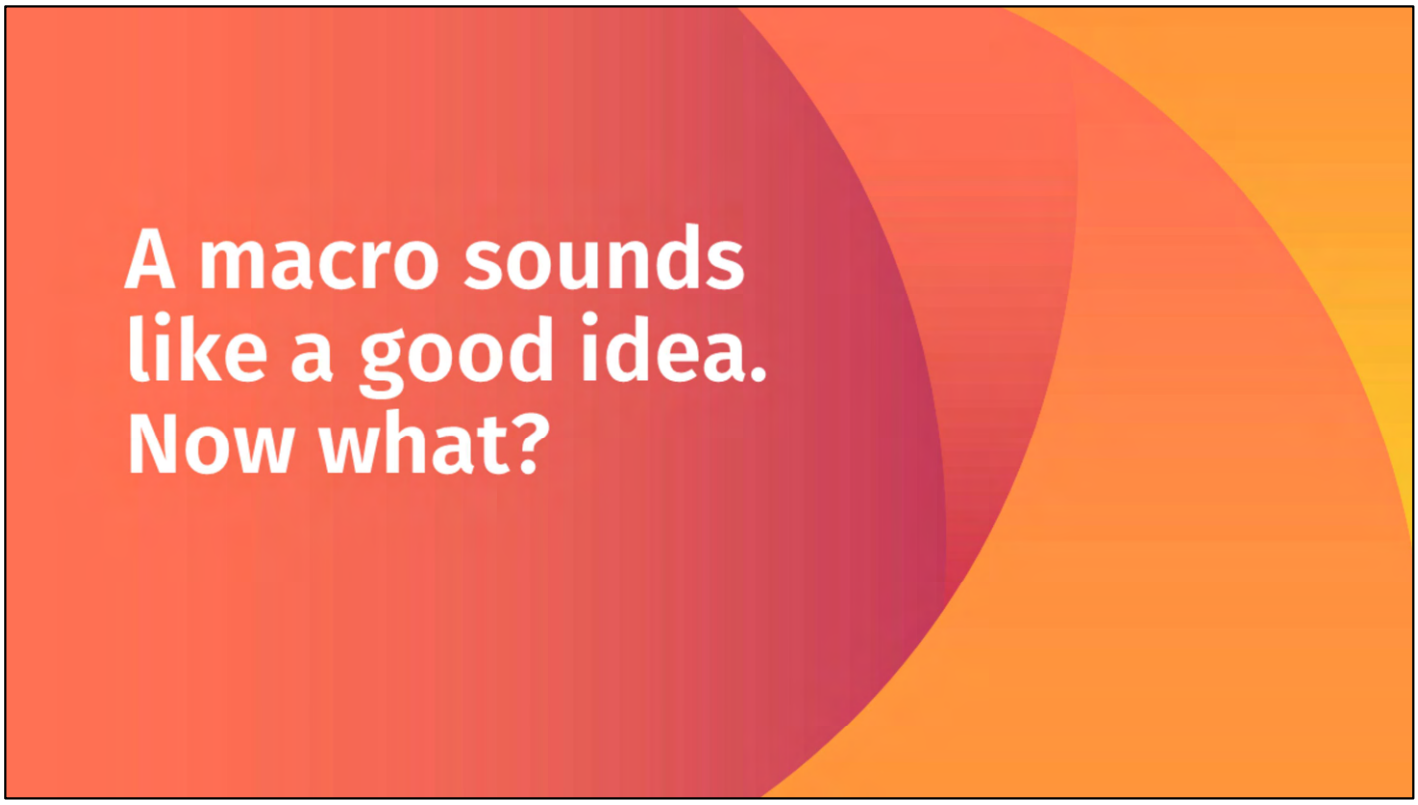

Sources of variation:
We'll go through this in a minute
Future potential?:
Yes.
Upgrades?
Also discuss in a minute.



Some of the other questions that may play into whether or not you want to create a macro are:

- How comfortable am I with macro variables? If not at all, then you might want to practice that as a stepping stone.
- Do I have the time to devote? Even with experience building macros, they take time. They can be finnicky. Do you have the time to devote to doing it well?
- Is it worth it? If it's going to take you 20 hours to build the macro and you're only going to use it for this one project, probably not worth it. If it's going in your code library, then might be more worth the time.


Comfort level with macro variables:
Very!
Time investment?:
Minimal.
Worth it?:
Yes!

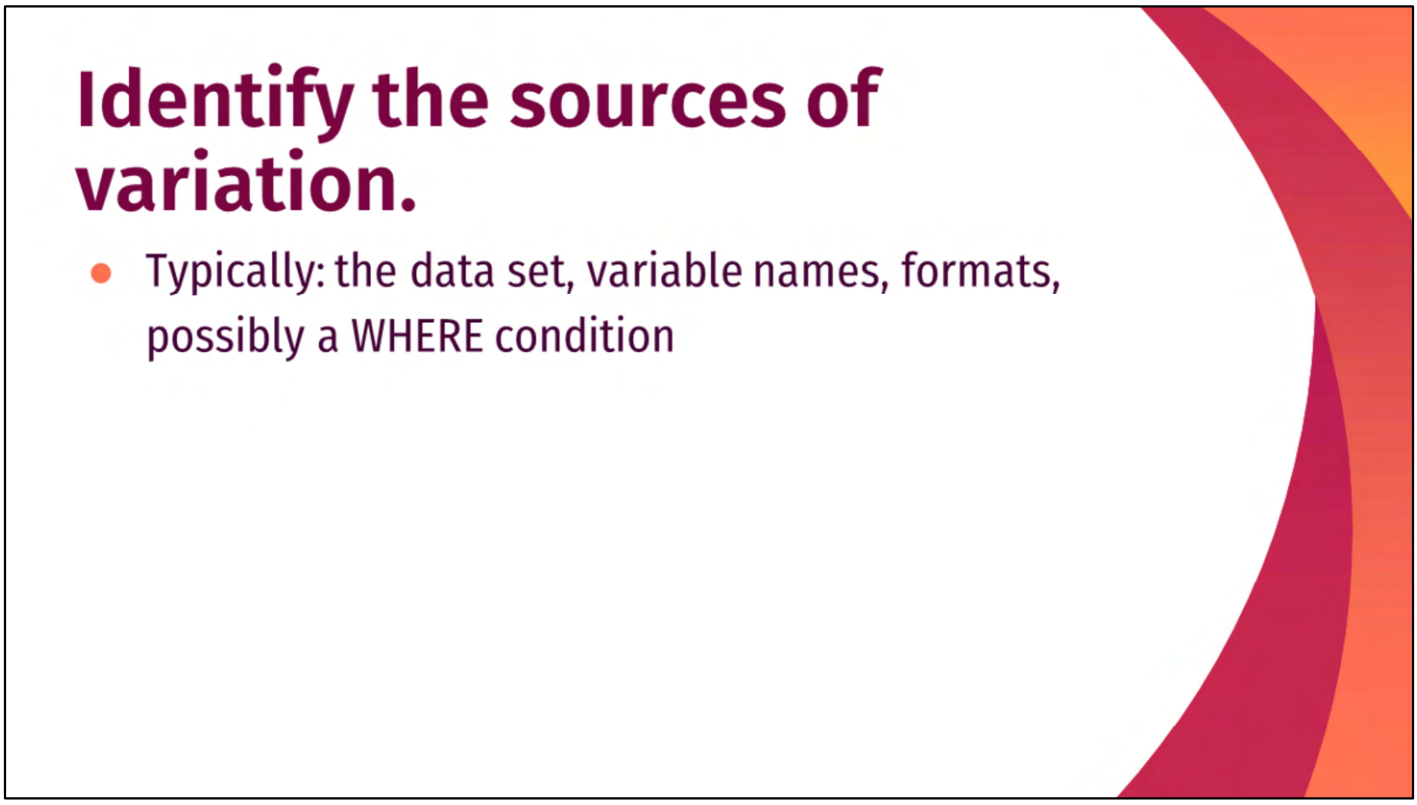

A macro sounds like a good idea. Now what?

Start small.

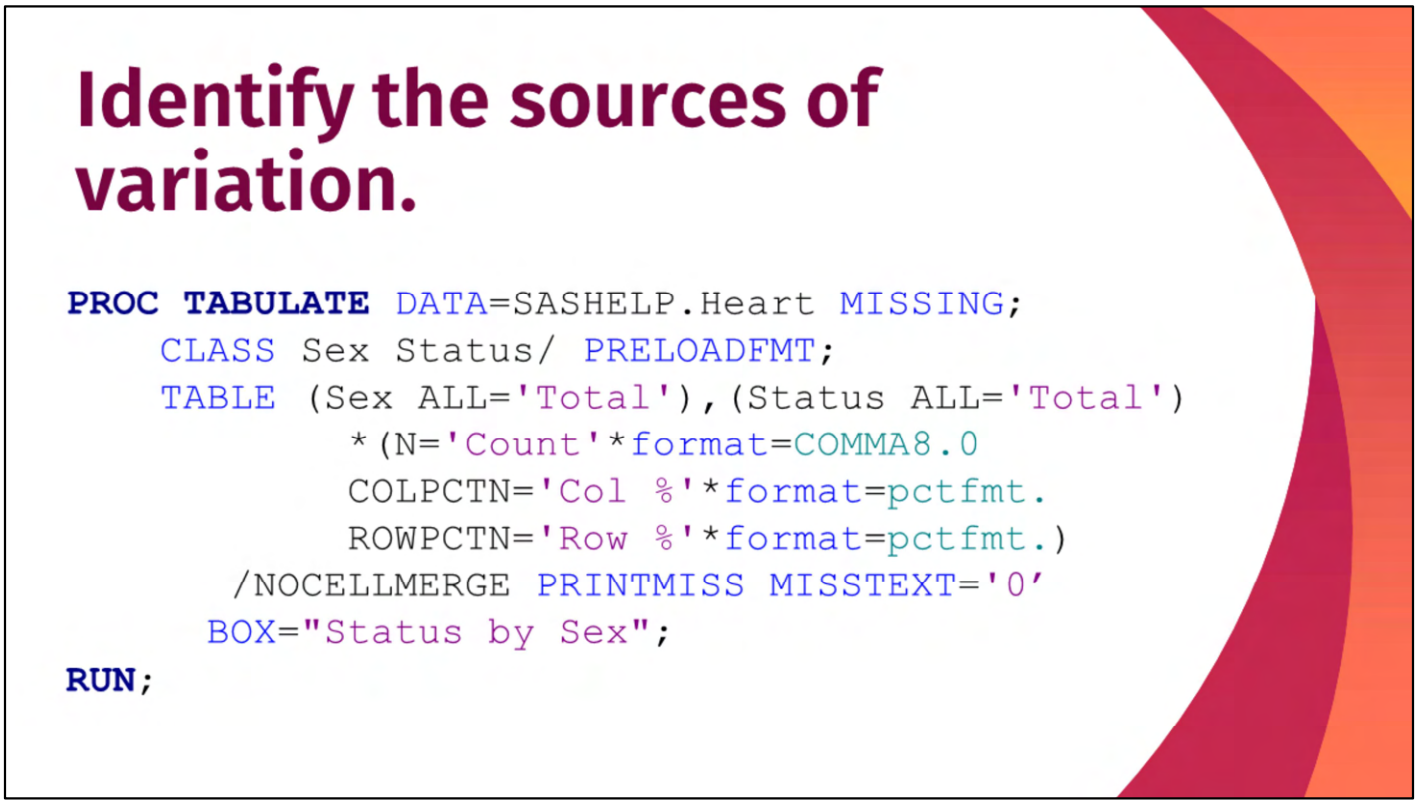
- Especially if the end product is big.
- Take it one piece at a time.

Status by Sex	Status						Total		
	Alive			Dead					
	Count	Col %	Row %	Count	Col %	Row %	Count	Col %	Row %
Sex									
Female	1,977	61%	69%	896	45%	31%	2,873	55%	100%
Male	1,241	39%	53%	1,095	55%	47%	2,336	45%	100%
Total	3,218	100%	62%	1,991	100%	38%	5,209	100%	100%

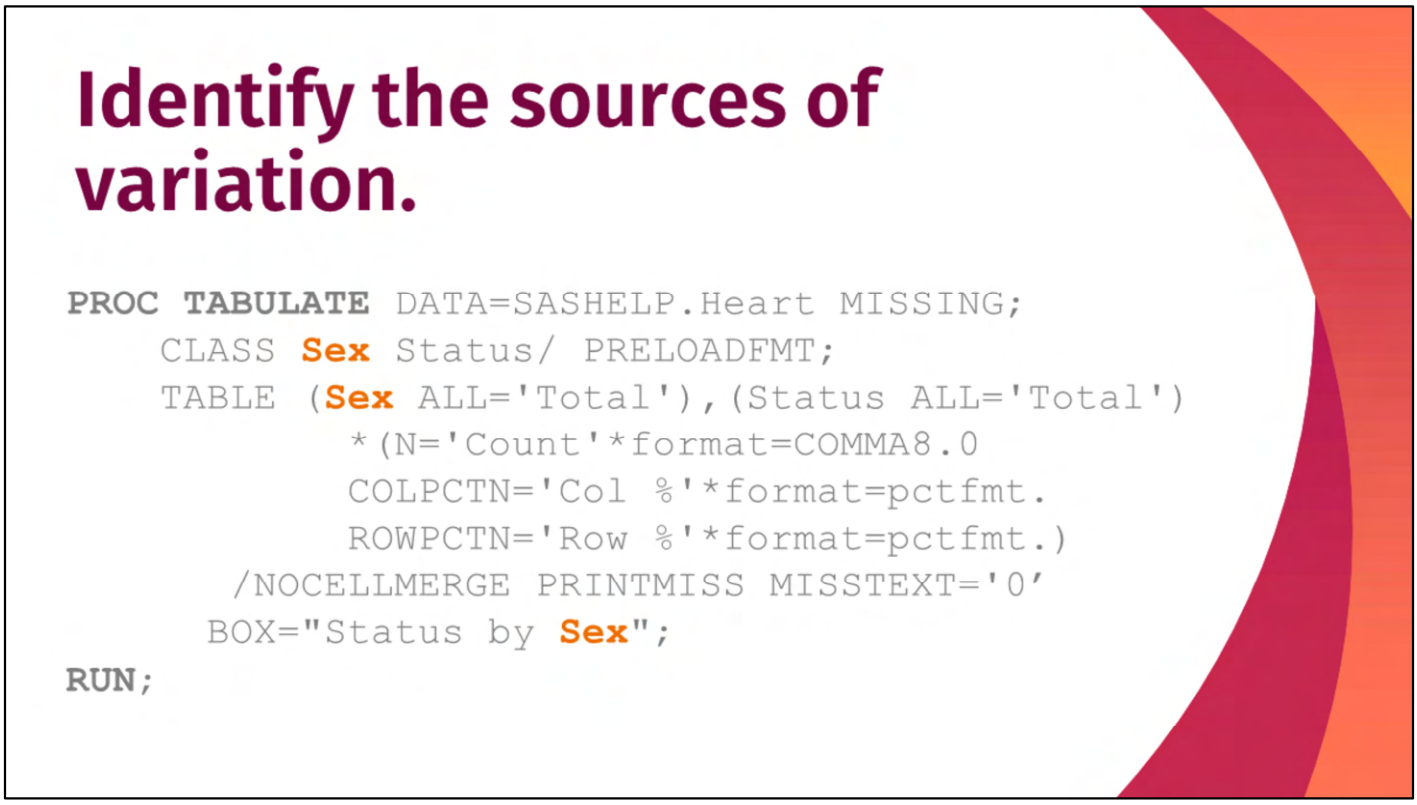
We're only going to work on automating this table, so we are starting small. I'll show a different example later that is a much larger project.



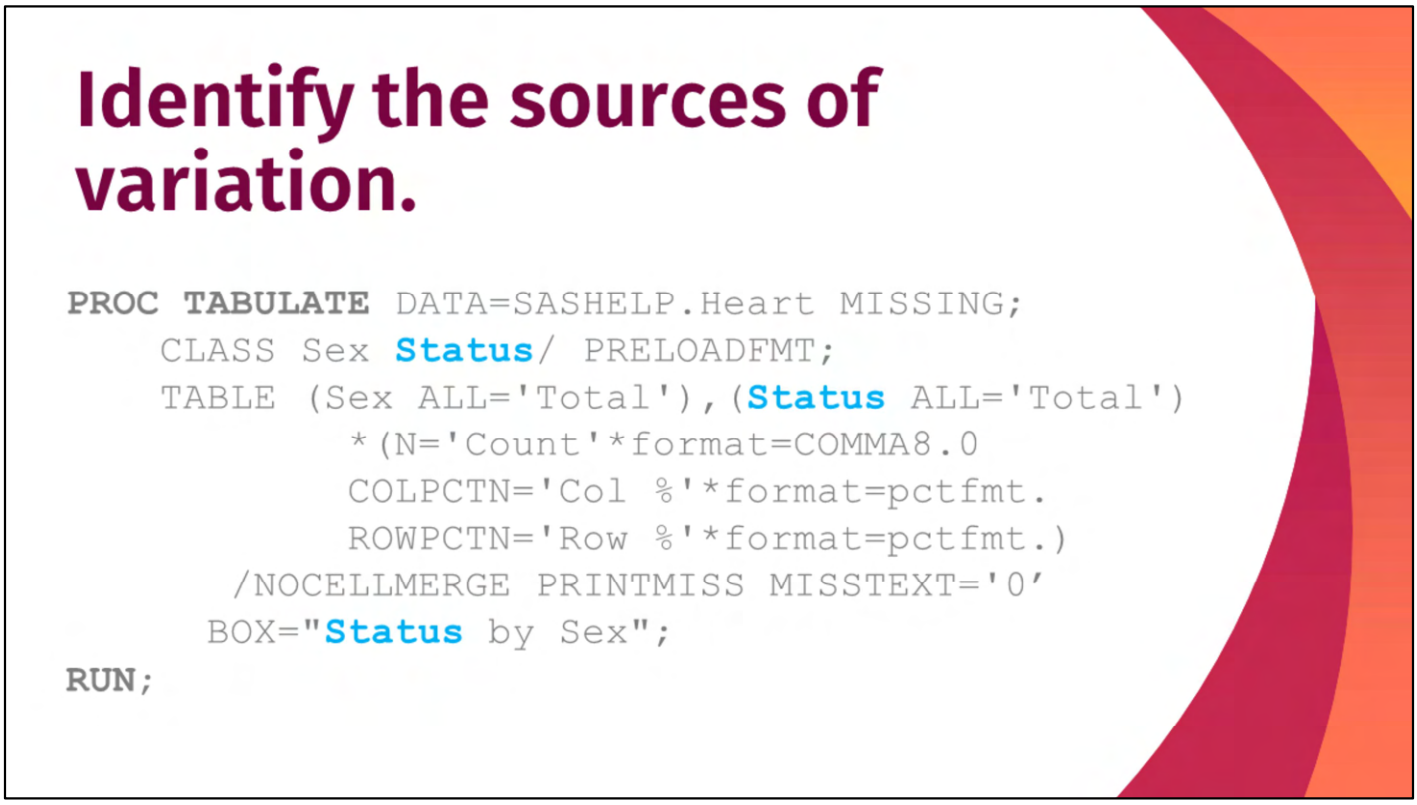
When I say source of variation, I mean what the dynamic pieces of code? What things in the code are you editing each time you copy/paste it? What things might you need to change in the future (e.g. dataset name, outcome variable, etc.) Those pieces that you edit will be the parameters of the macro.



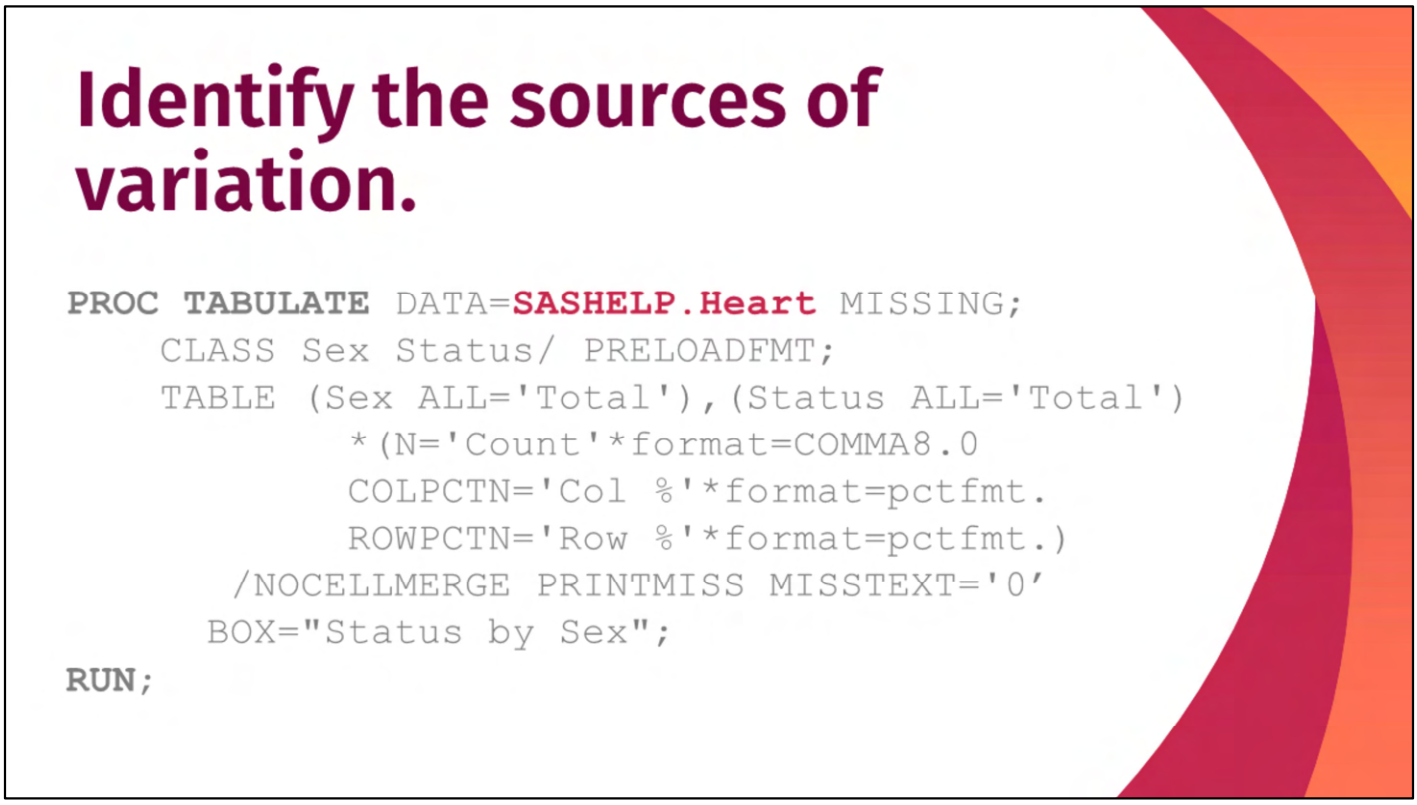
Let's take the PROC TABULATE example from the beginning and identify the dynamic code/sources of variation.



When I copied/pasted, I was only changing the variable name for the row variable, sex, in these 3 places. All of the other code stayed the same. So this is one source of variation that will need



In addition to sex, I could see a situation where I need to look at associations with other outcomes. That could be another source of variation – another



I could also envision a future where I need a different dataset, so we might want to account for that in our parameters. We may need to: add a where statement, add other formats, etc. The more robust the macro is, the more parameters generally need to be defined, and the more complex it is to build and troubleshoot.

Identify the sources of variation.

```
PROC TABULATE DATA=SASHELP.Heart MISSING;
    CLASS Sex Status/ PRELOADFMT;
```

Definitely need parameters for the row variable. Possibly need them for the column variable and dataset name.

```
    BOX="Status by Sex";
RUN;
```

Don't jump straight to a macro. Instead, substitute macro variables.

```
PROC TABULATE DATA=SASHELP.Heart MISSING;
    CLASS Sex Status/ PRELOADFMT;
    TABLE (Sex ALL='Total'),(Status ALL='Total')
            *(N='Count'*format=COMMA8.0
            COLPCTN='Col %'*format=pctfmt.
            ROWPCTN='Row %'*format=pctfmt.)
       /NOCELLMERGE PRINTMISS MISSTEXT='0'
      BOX="Status by Sex";
RUN;
```

```
%LET RowVar = Sex;


PROC TABULATE DATA=SASHELP.Heart MISSING;
    CLASS &RowVar. Status / PRELOADFMT;
    TABLE &RowVar. ALL='Total'),(Status ALL='Total')
          *(N='Count'*format=COMMA8.0
          COLPCTN='Col %'*format=pctfmt.
          ROWPCTN='Row %'*format=pctfmt.)
      /NOCELLMERGE PRINTMISS MISSTEXT='0'
     BOX="Status by &RowVar.";
RUN;
```

I'm going to define macro variables for each of the dynamic pieces of code/sources of variation. I'll start with the row variable, sex. I'll define the macro variable RowVar to be Sex and then substitute &RowVar. Everywhere Sex used to be.

Note that in the BOX option, that's a label. I can substitute the variable name because it's also descriptive, but there may be times where I want that to be something different (more descriptive or formatted differently)…. So in a future iteration, I may want the labels to be separate parameters. But for now, we're going simple so we'll leave it as is.

```
%LET RowVar = Sex;
%LET ColVar = Status;


PROC TABULATE DATA=SASHELP.Heart MISSING;
    CLASS &RowVar. &ColVar. / PRELOADFMT;
    TABLE (&RowVar. ALL='Total'), &ColVar. ALL='Total')
          *(N='Count'*format=COMMA8.0
          COLPCTN='Col %'*format=pctfmt.
          ROWPCTN='Row %'*format=pctfmt.)
      /NOCELLMERGE PRINTMISS MISSTEXT='0'
     BOX="&ColVar. by &RowVar.";
RUN;
```

Now we do the same thing with the column variable, Status. We call its macro variable ColVar.

```
%LET RowVar = Sex;
%LET ColVar = Status;
%LET DatasetName = SASHelp.Heart;

PROC TABULATE DATA=&DatasetName. MISSING;
    CLASS &RowVar. &ColVar. / PRELOADFMT;
    TABLE (&RowVar. ALL='Total'),(&ColVar. ALL='Total')
            *(N='Count'*format=COMMA8.0
            COLPCTN='Col %'*format=pctfmt.
            ROWPCTN='Row %'*format=pctfmt.)
       /NOCELLMERGE PRINTMISS MISSTEXT='0'
      BOX="&ColVar. by &RowVar.";
RUN;
```

Last, I do it for the dataset name.

```
%LET RowVar = Sex;
%LET ColVar = Status;
%LET DatasetName = SASHelp.Heart;

PROC TABULATE DATA=&DatasetName. MISSING;
    CLASS &RowVar. &ColVar. / PRELOADFMT;
    TABLE (&RowVar. ALL='Total'),(&ColVar. ALL='Total')
            *(N='Count'*format=COMMA8.0
            COLPCTN='Col %'*format=pctfmt.
            ROWPCTN='Row %'*format=pctfmt.)
       /NOCELLMERGE PRINTMISS MISSTEXT='0'
      BOX="&ColVar. by &RowVar.";
RUN;
```

Now I have code that looks like this. I'll run this chunk of code and make sure that the log runs clean and that the output matches the output from before. I might even test another RowVar to make sure that runs correctly.

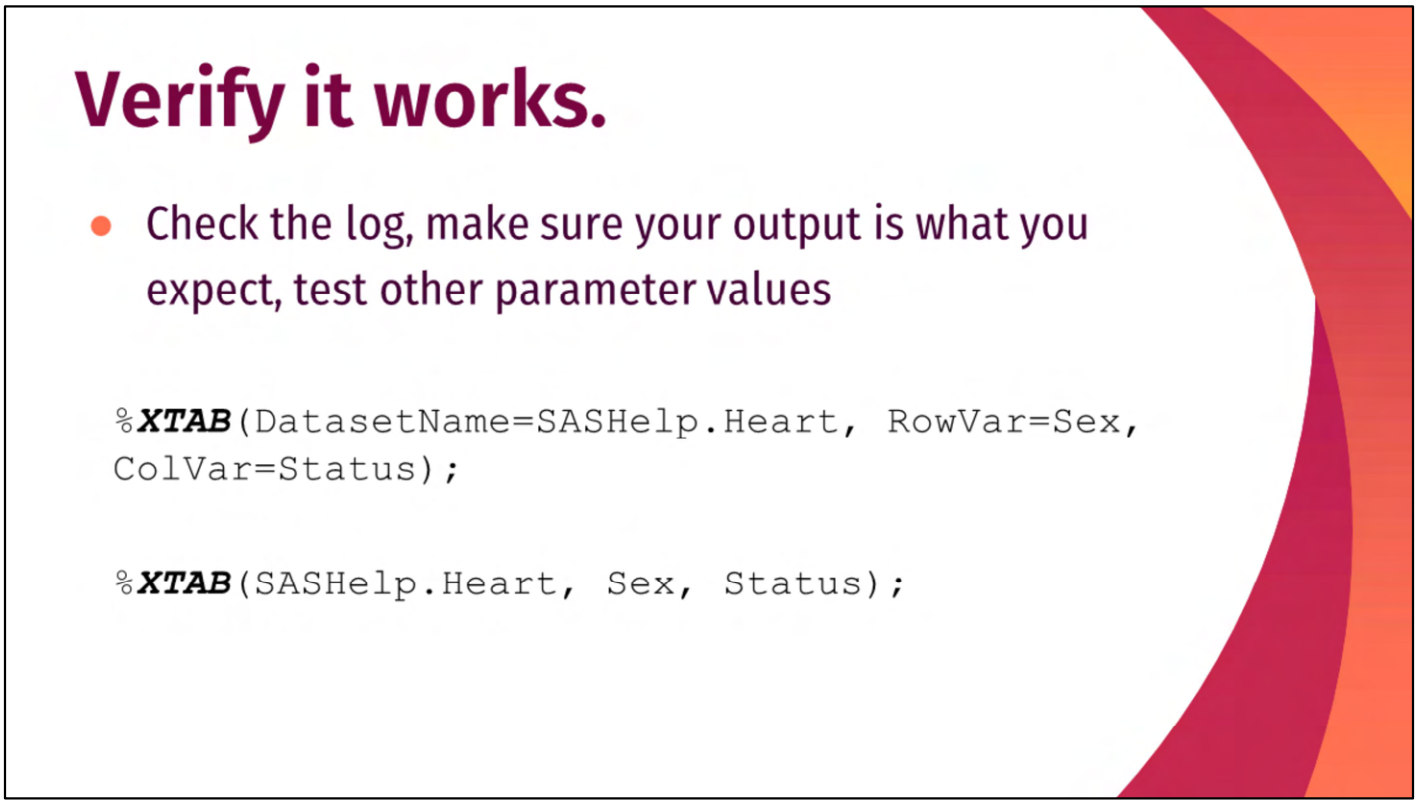
After it runs with macro variables, construct the %macro.

- The macro variables are going to be your macro parameters.

```
%LET RowVar=Sex;
%LET ColVar=Status;
%LET DatasetName=SASHelp.Heart;
```

```
%MACRO XTab(DatasetName, RowVar, ColVar);
PROC TABULATE DATA=&DatasetName. MISSING;
    CLASS &RowVar. &ColVar./ PRELOADFMT;
    TABLE (&RowVar. ALL='Total'),(&ColVar. ALL='Total')
          *(N='Count'*format=COMMA8.0
          COLPCTN='Col %'*format=pctfmt.
          ROWPCTN='Row %'*format=pctfmt.)
      /NOCELLMERGE PRINTMISS MISSTEXT='0'
     BOX="&ColVar. by &RowVar.";
RUN;
%MEND XTab;
```

I've just wrapped that same PROC TABULATE code that I was just testing in a Macro. I used %MACRO to define a macro called Xtab with 3 parameters. The parameter names are the names I just used for the macro variables. At the end of the tabulate, I end the macro with a %MEND. I wouldn't need that last "Xtab" on there, but it's good practice when you're just starting. So now I have my macro!



Now I call the macro. Note that I have to define the parameters in the macro call…. I'm explicitly telling SAS what I want it to substitute for those macro variables.

I don't need to use the explicit calls… I can just give a list of arguments, but SAS will take the arguments and assign them in order, so if you're skipping a parameter or get them out of order, you may not get the results you want.

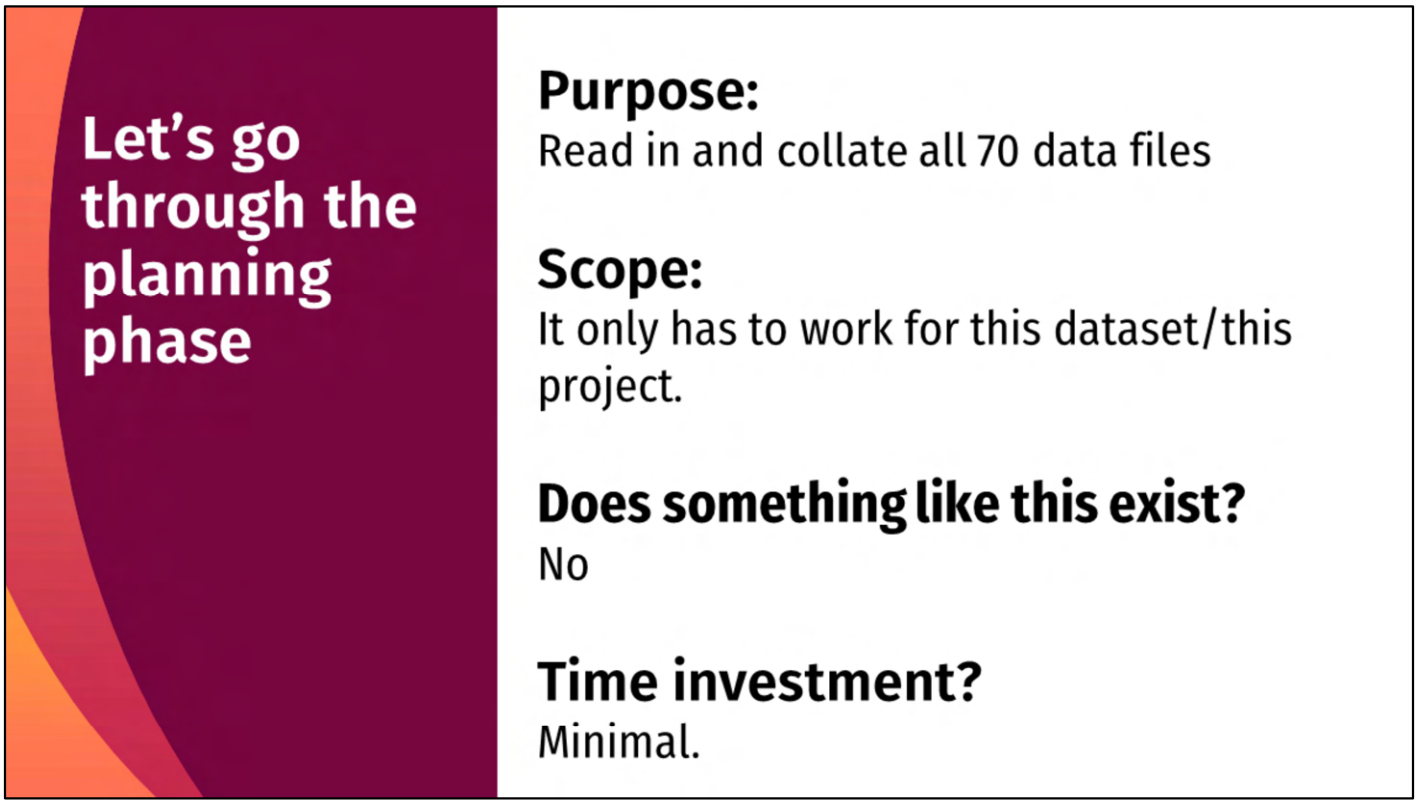
That's it. Once you get it working, you can keep refining it, tweaking it, expanding it, the same way you would with non-macro code.

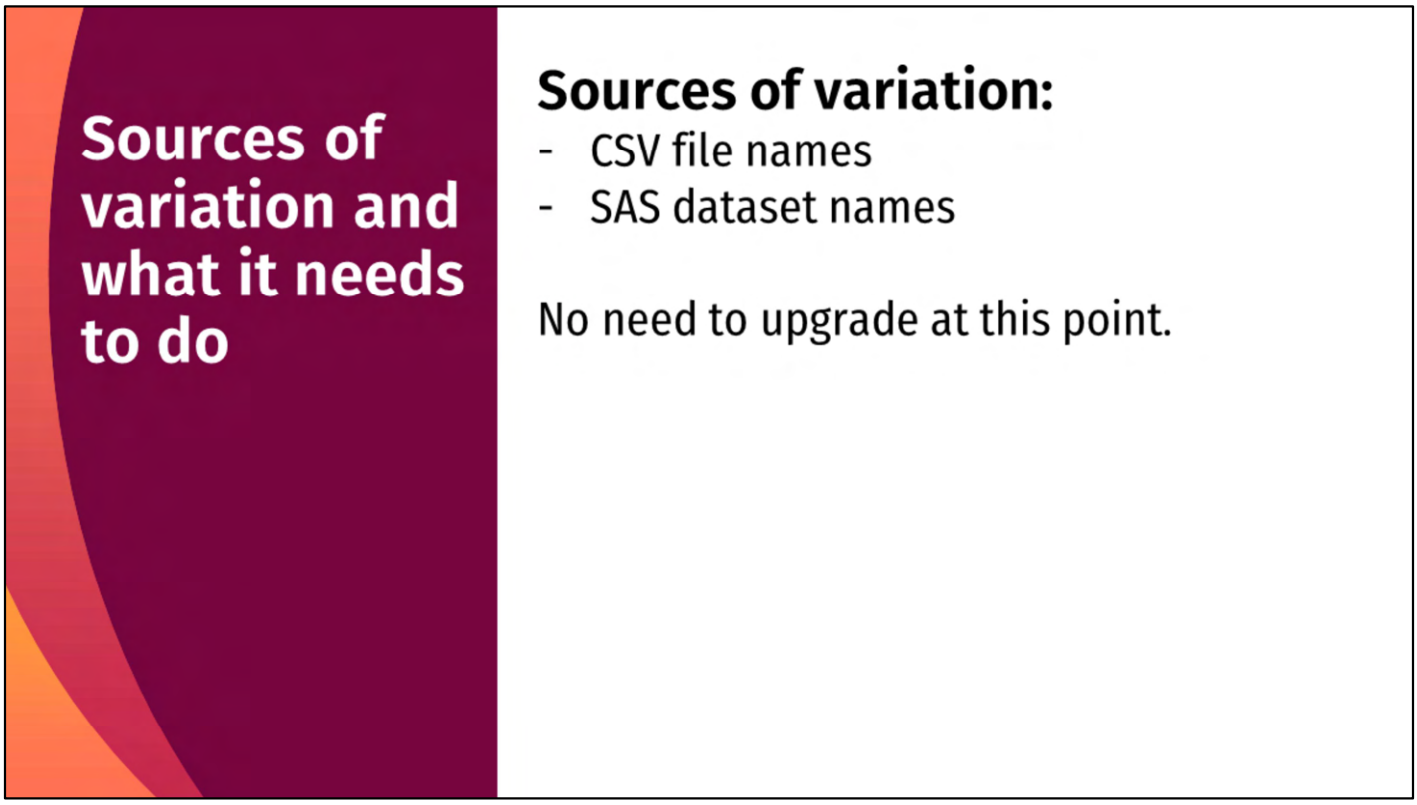
Example: Reading in LOTS of CSV files...

The problem:
Client sent 70 CSV files for me to process/work with.

All files have the same format and the same 5 variables.

I don't want to copy/paste/edit 70 chunks of code.


Let's go through the planning phase
Purpose:
Read in and collate all 70 data files
Scope:
It only has to work for this dataset/this project.
Does something like this exist?
No
Time investment?
Minimal.

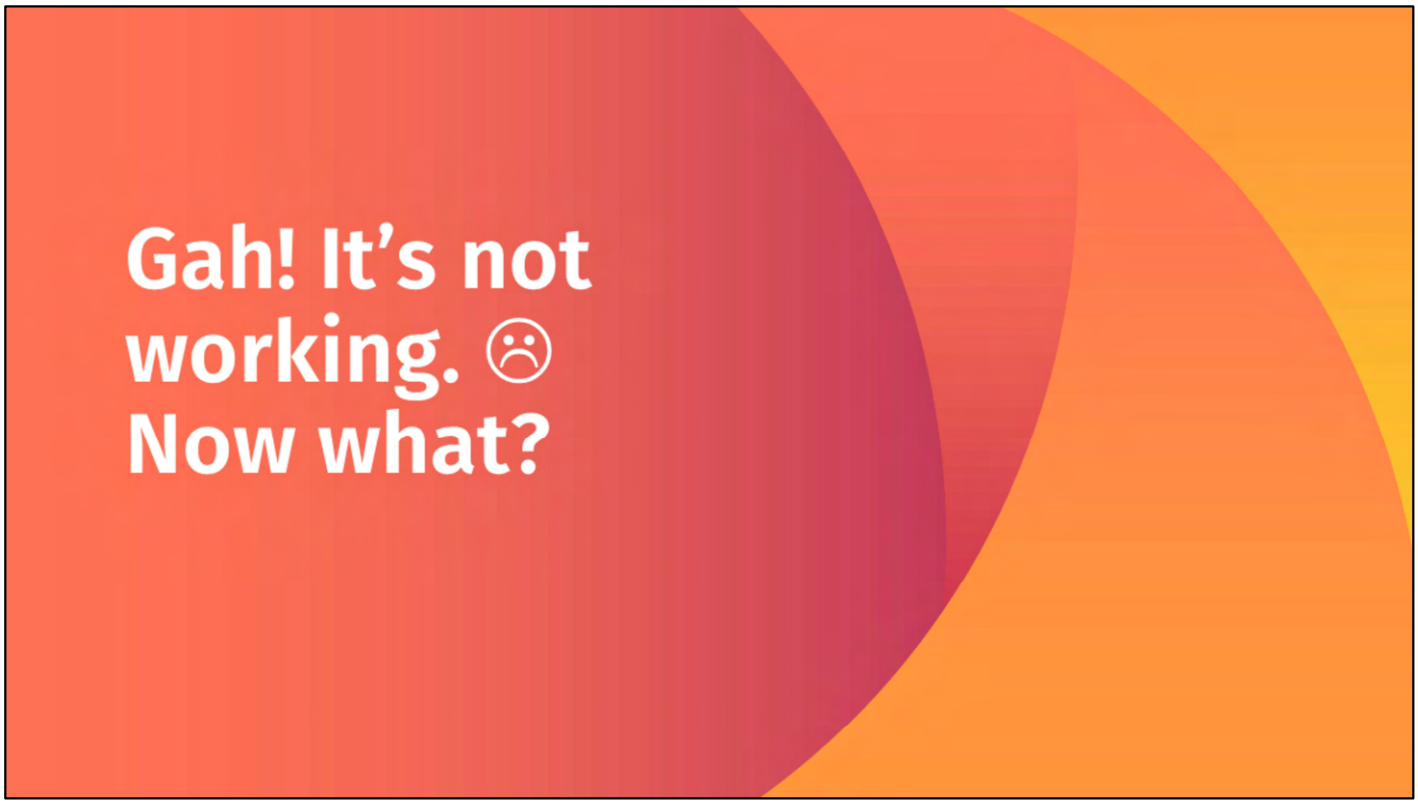

Sources of variation and what it needs to do
Sources of variation:
- CSV file names
- SAS dataset names
No need to upgrade at this point.

```
* Get list of all files;
data steroid_hcpcs (keep=ss_name dataname);
       rc=filename("mydir","&hcpcsinpath.");
       did=dopen("mydir");
       if did>0 then
              do i=1 to dnum(did);
                     ss_name=dread(did,i);
                     dataname=SCAN(ss_name,1,'.');
                     output;
              end;
       rc=dclose(did);
run;
* Index obseravtions;
data steroid_hcpcsv2;
       set steroid_hcpcs;
       obsid=_n_;
run;
```

```
* Read in first one just to make sure it works. ;
%let sasds=data_0_2_0;
%let ssfile=data_0_2_0.csv;
DATA &sasds.;
      INFILE "&hcpcsinpath.\&sasds..csv" DSD FirstObs=2;
      INPUT FORIAN_ID:$16. DATE_OF_SERVICE:$12.
      D_PROCEDURE_CODE:$7. SHORTDESCRIPTION:$27.;
RUN;
```

```
%macro process_csv(ssfile, sasds);
/* Read in file;*/
DATA &sasds.;
      INFILE "&hcpcsinpath.\&sasds..csv" DSD FirstObs=2;
      INPUT FORIAN_ID:$16. DATE_OF_SERVICE:$12.
      D_PROCEDURE_CODE:$7. SHORTDESCRIPTION:$27.;
RUN;
/* Append;*/
PROC DATASETS NOLIST;
      APPEND BASE=fi.steroid_hcpcs DATA=&sasds. FORCE;
RUN;
QUIT;
%mend;
```

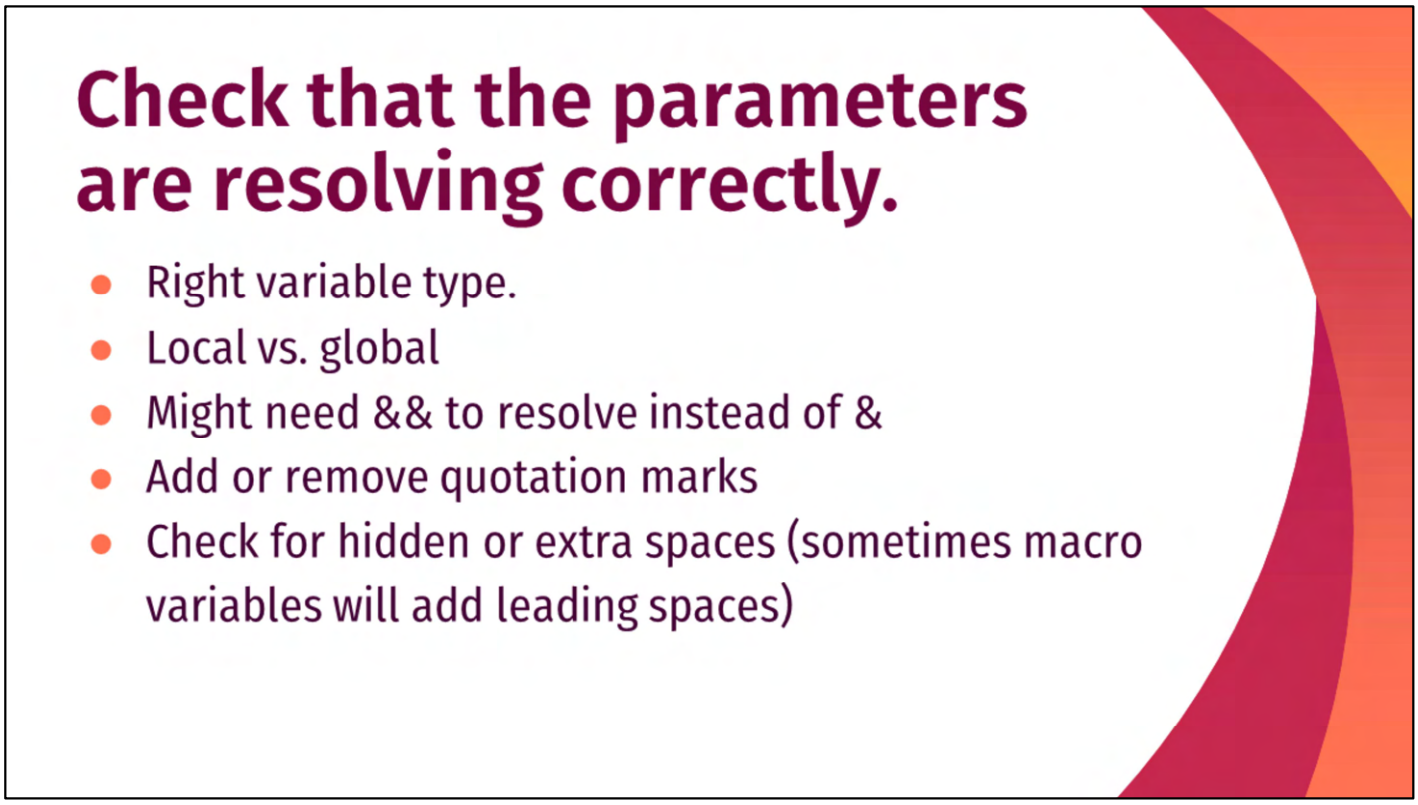
```
* Call the macro to import data;
proc sort data=steroid_hcpcsv2;
      by obsid;
run;
data _null_;
      set steroid_hcpcsv2 (where=(obsid>2));
      by obsid;
      if first.obsid then
      rc = dosubl('%process_csv(ssfile='||ss_name||',
            sasds='||dataname||')');
run;
```

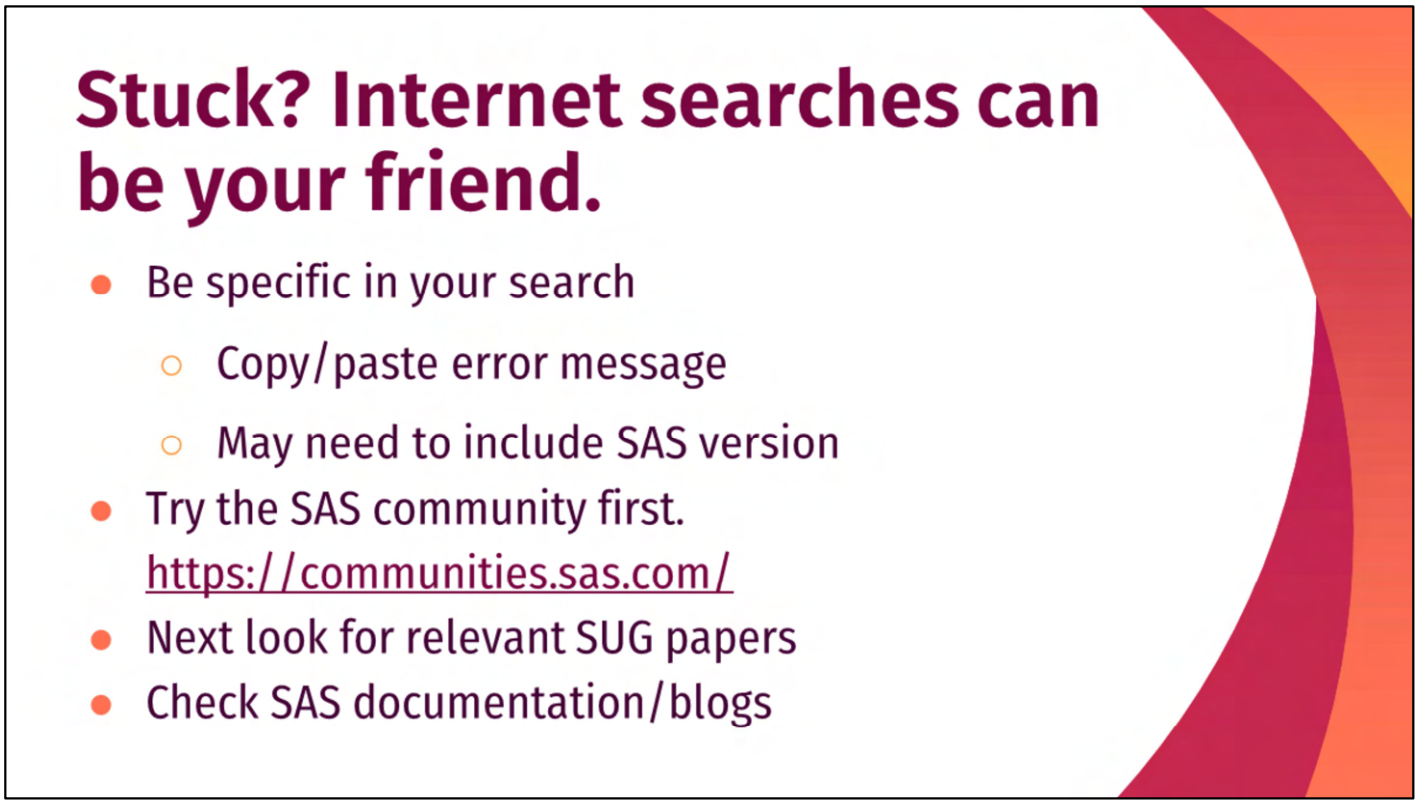

Gah! It's not working. ☹
Now what?


Try using one of the macro de-bugging tools.
SAS has built in tools that can help including:
MERROR
MLOGIC
MPRINT

Verify parameters are correct.

- That means....
  - ... spelled correctly
  - ... listed in the correct order
  - ... correct variable type (numeric vs. character vs. date)
  - ... correctly specified (single vs. multiple entries)
  - ... defined in the macro call


Check that the parameters are resolving correctly.
Right variable type.
Local vs. global
Might need && to resolve instead of &
Add or remove quotation marks
Check for hidden or extra spaces (sometimes macro variables will add leading spaces)


Stuck? Internet searches can be your friend.
Be specific in your search
Copy/paste error message
May need to include SAS version
Try the SAS community first.
https://communities.sas.com/
Next look for relevant SUG papers
Check SAS documentation/blogs


Thanks!
Any questions?
You can find me at:
SRichter@RichterStats.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.