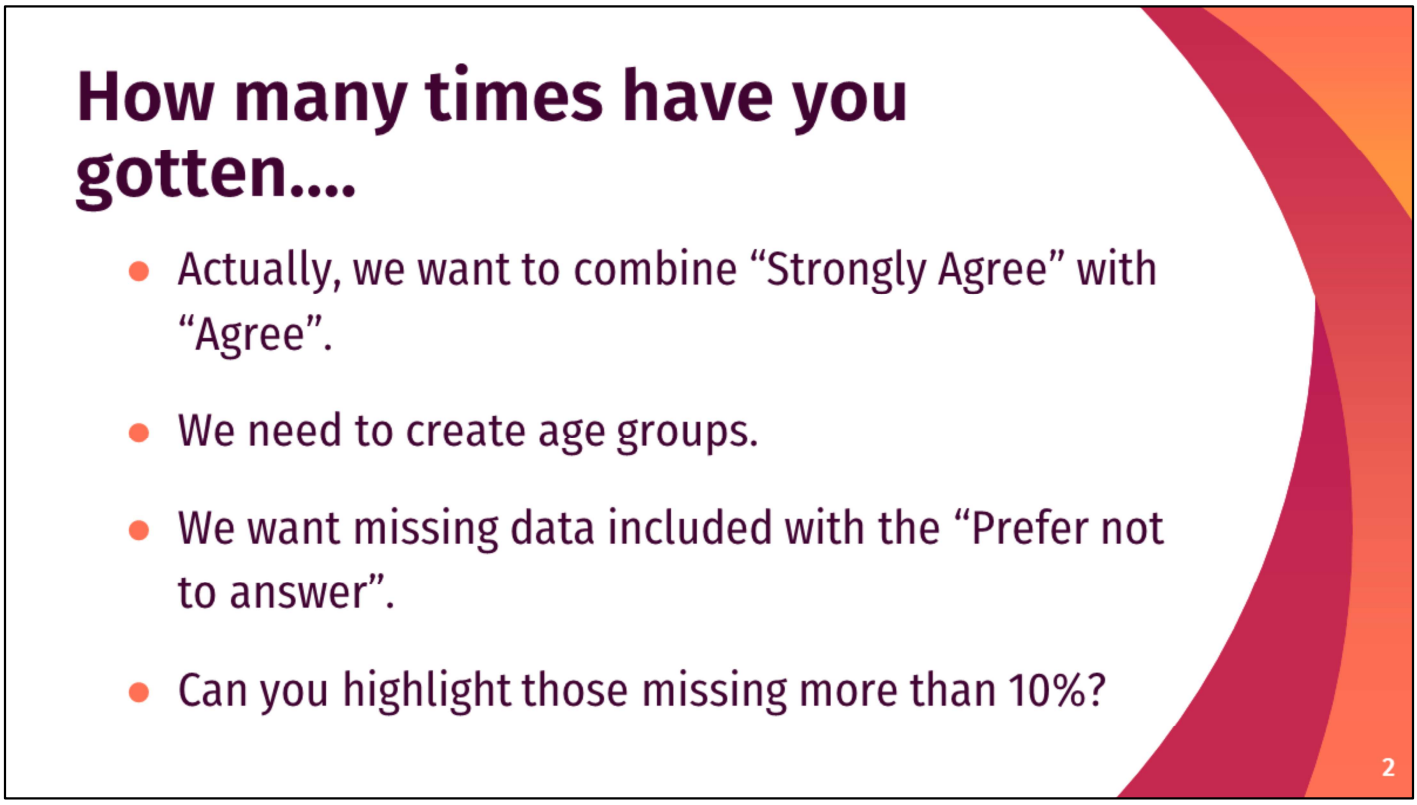
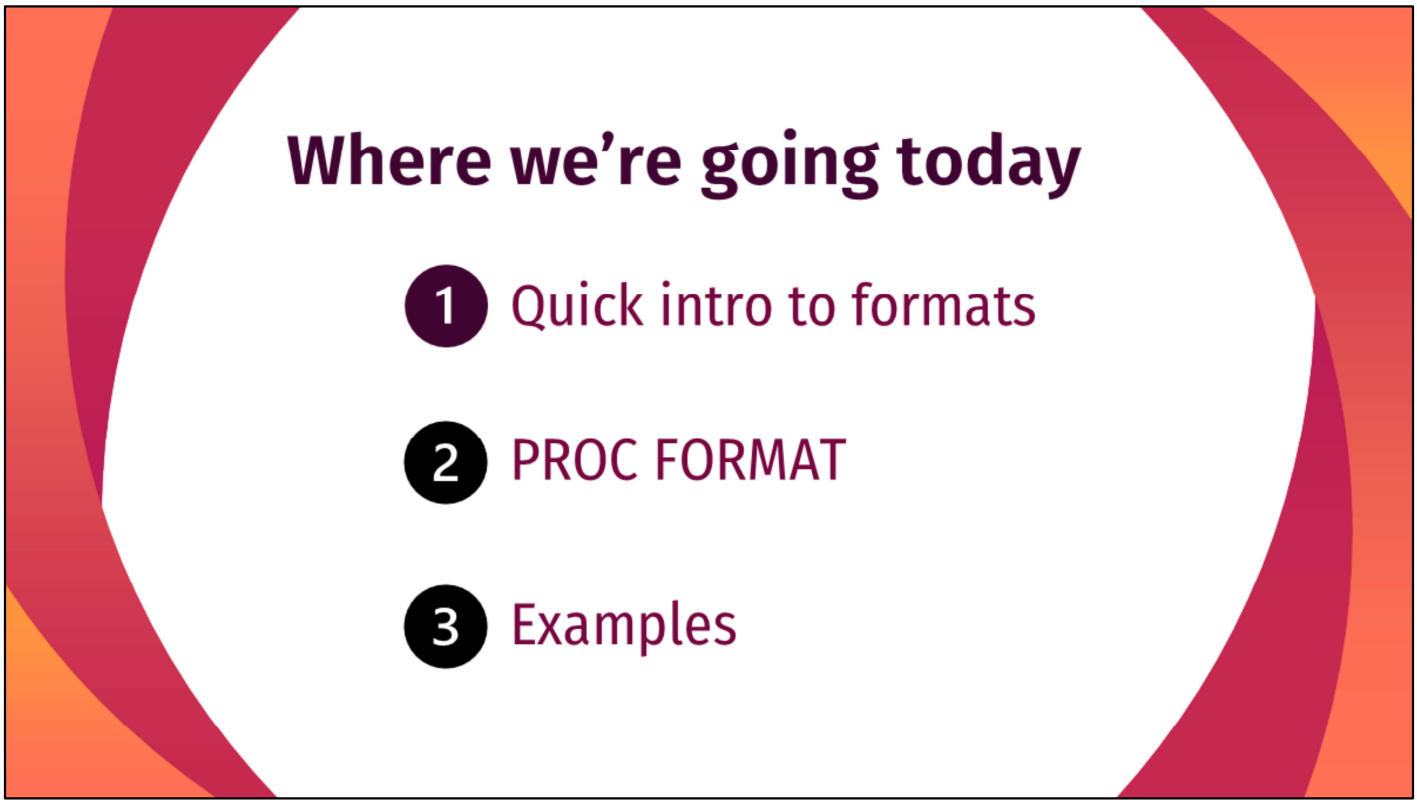
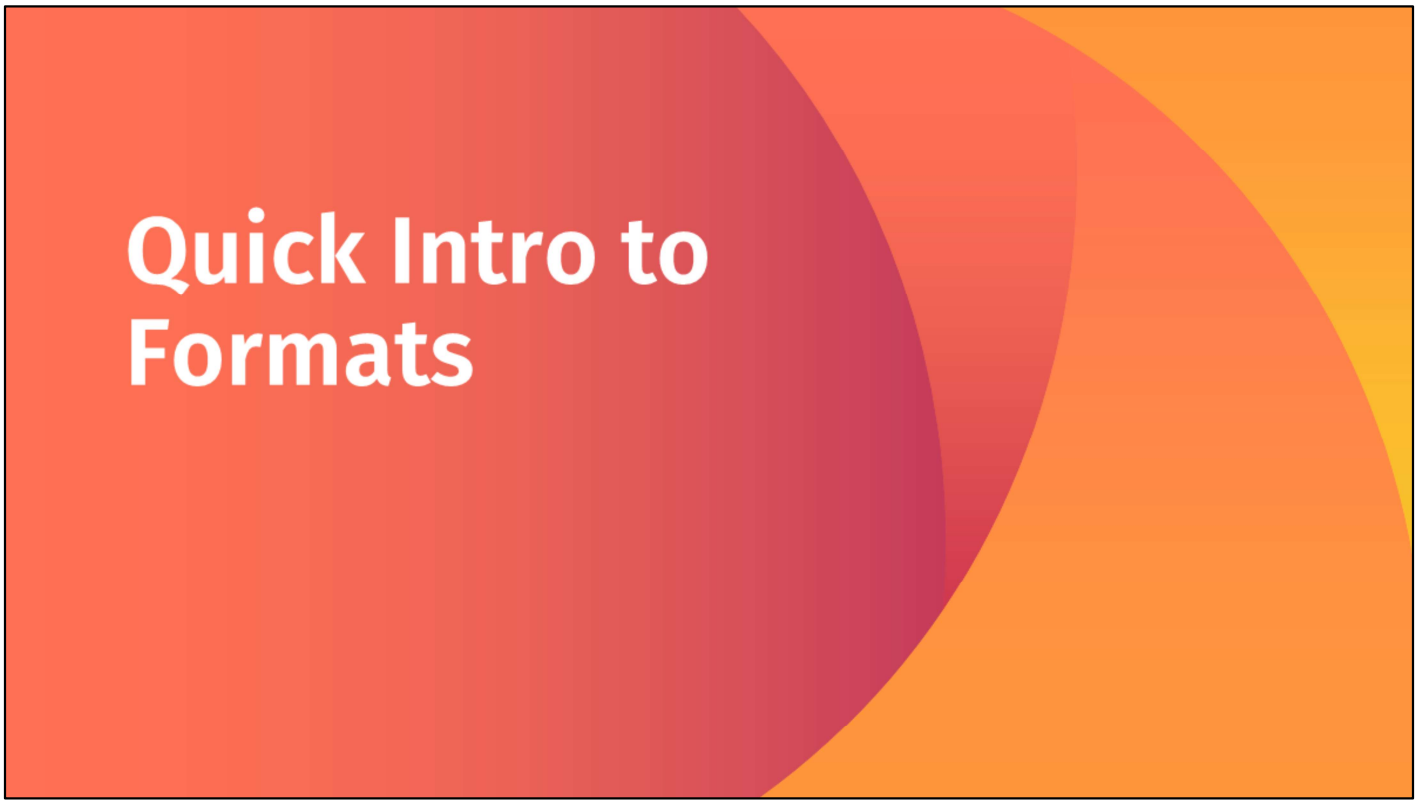
Making the most of user-defined formats

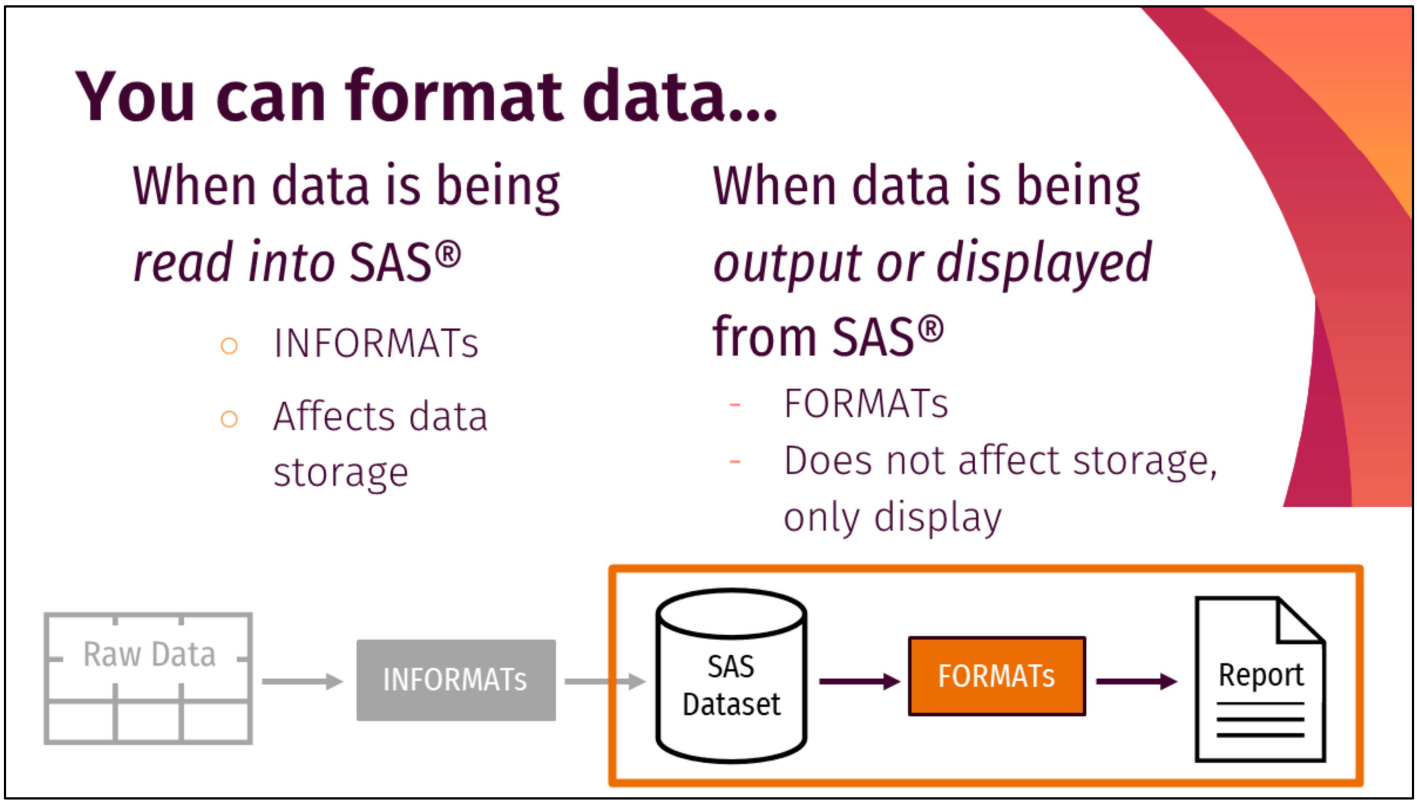
Sara Richter, MS
MWSUG Annual Conference 2024
#BB104



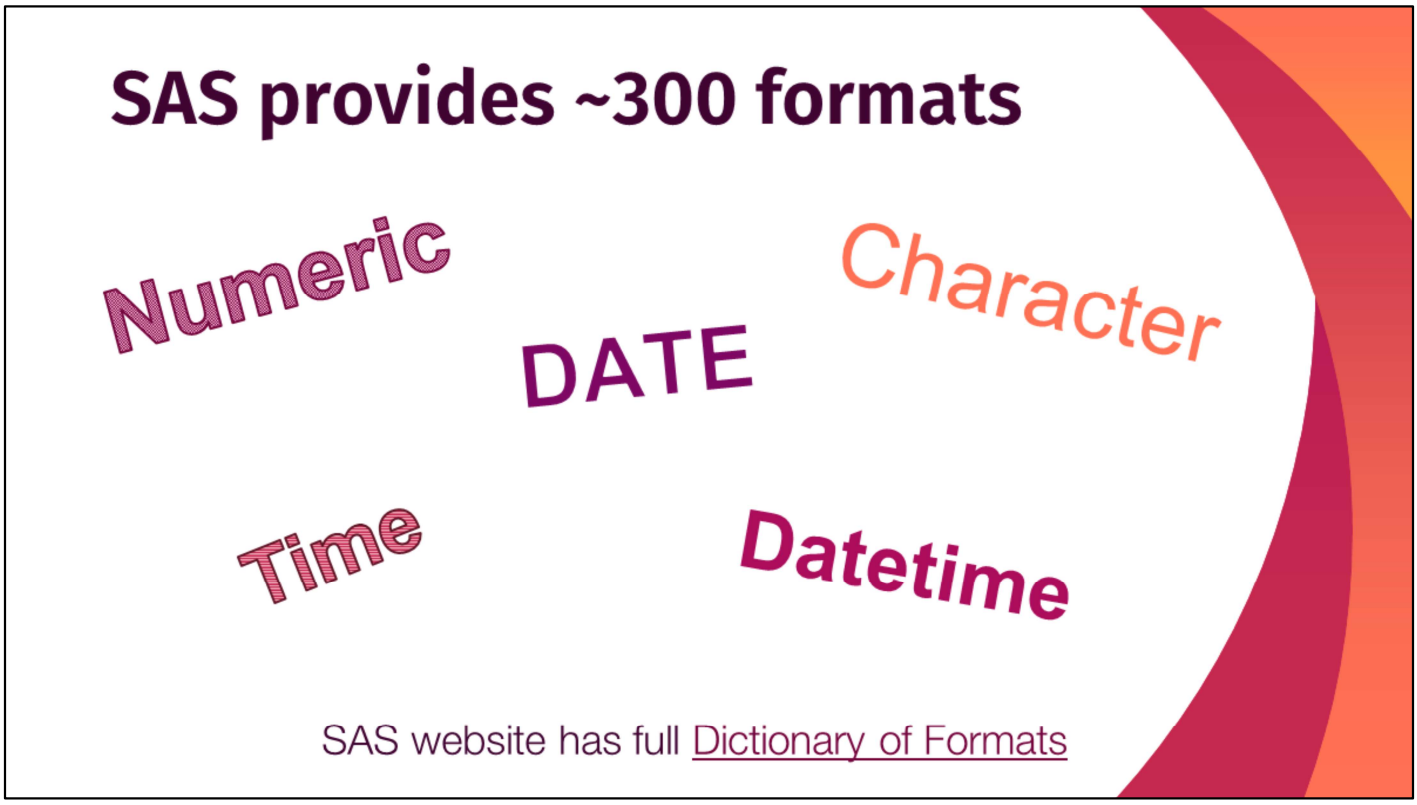
Often I receive requests for results revisions where the timeline is short. Formats offer a way to be very nimble with how results are displayed.


Where we're going today
1 Quick intro to formats
2 PROC FORMAT
3 Examples

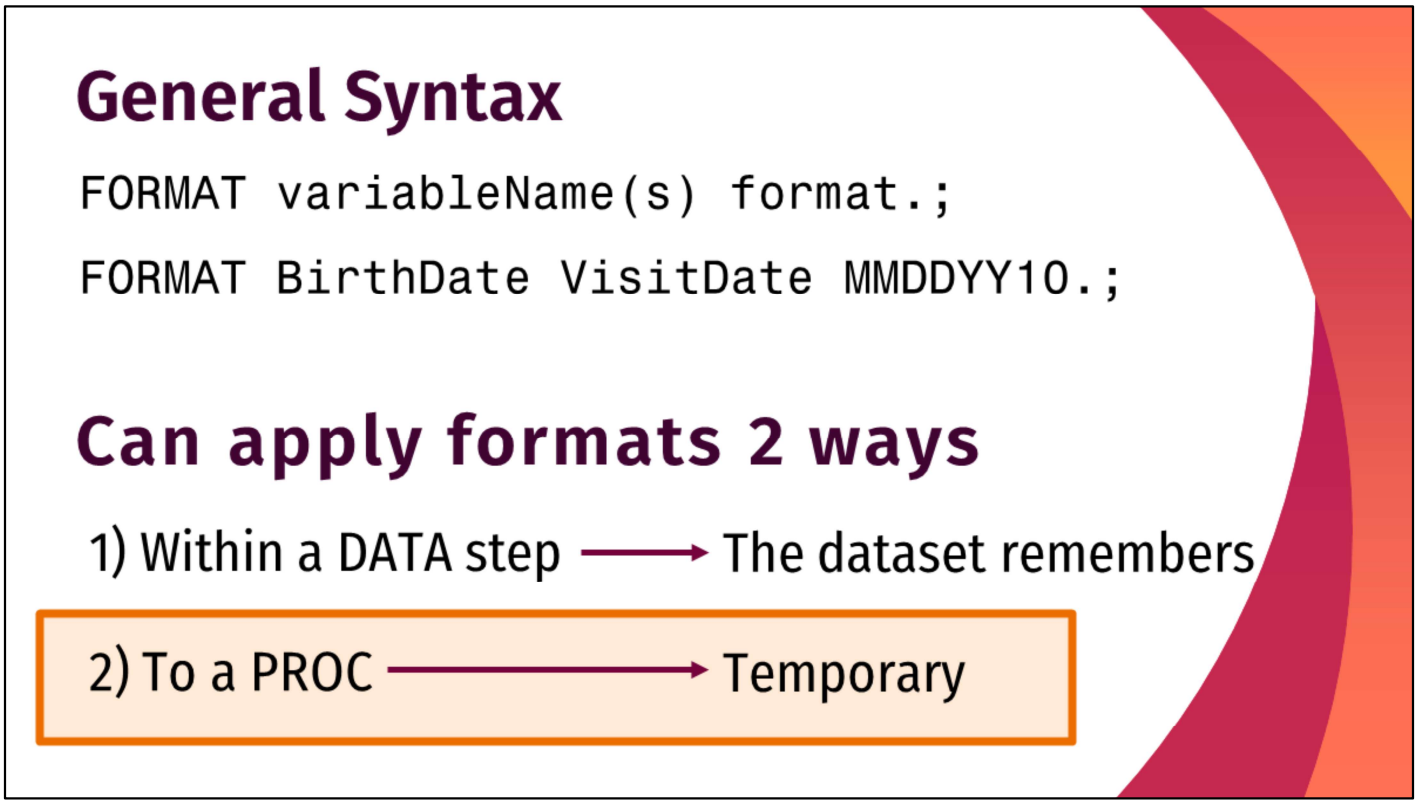

Quick Intro to
Formats



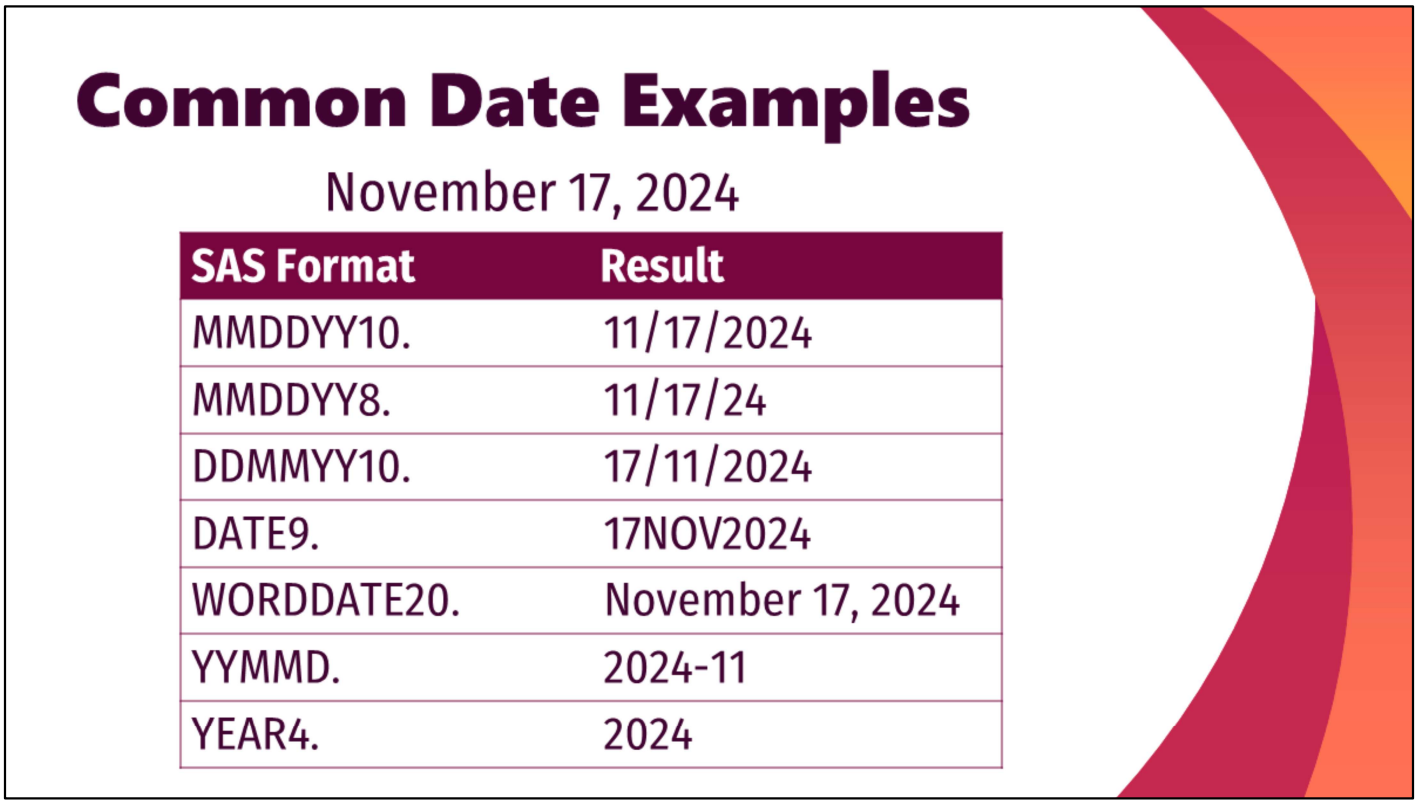
You can format data when it's read into SAS®, which affects how data is stored, or you can format data when it's being output or displayed. When you format it on the way out, it does not affect storage. The examples that I'm showing today focus on an efficient/quick solution to making changes to data, so we're focusing on formatting data on displays using FORMATS.



SAS provides almost 300 built-in formats for all types of variables – numeric, character, date, time, and date-time. We'll look at some examples in a minute.

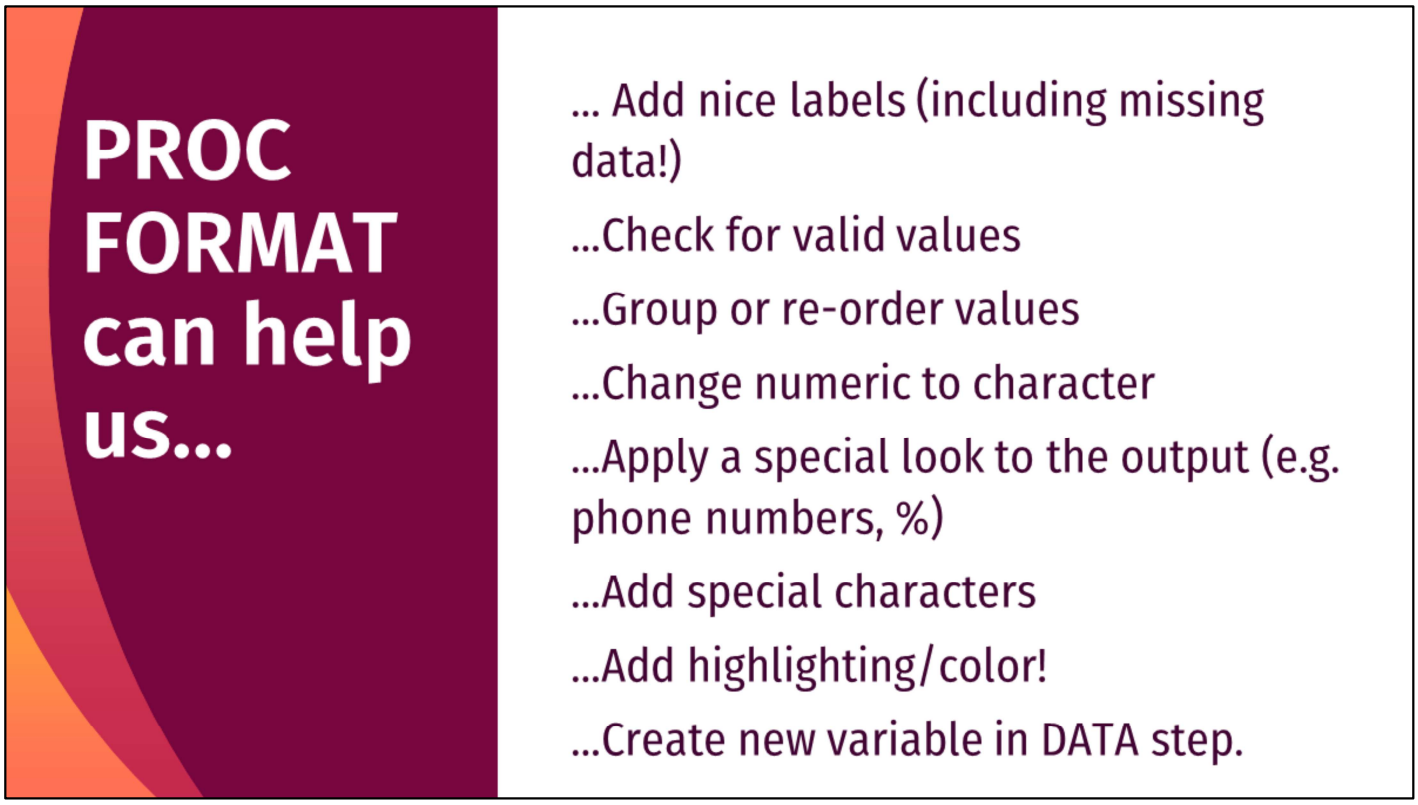


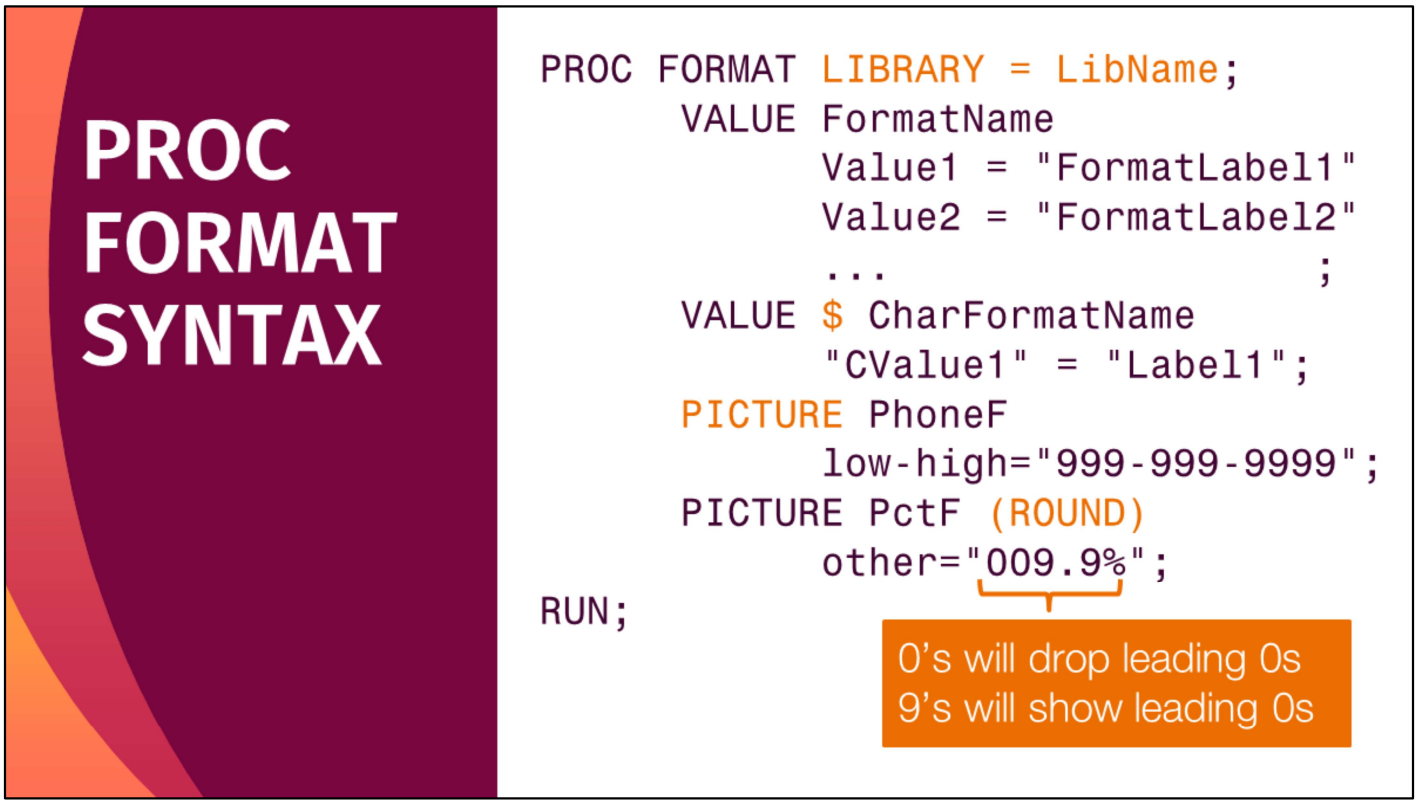
The general syntax involves a FORMAT keyword, the name of the variable or variables you want to be formatted, and then the format you want to apply. It's important to end the format with a period (.) to tell SAS it's a format. Then the semi-colon. An example is that I want Birthdate and Visit date to have the MM/DD/YYYY format (month, day, 4-digit year separated by slashes). You can apply that FORMAT statement in 2 places – 1) within a DATA step. When you apply it here, the dataset remembers that you want that variable to be formatted that way. Whether you open the dataset to browse or use those variables, it will remember you want it to look a certain way. That's not the nimble solution we're looking for, so that's not what we'll talk about. The other way you can apply a FORMAT statement is within a procedure. When you use it in a procedure, it's temporary – SAS will only use that format for that one procedure. That's the nimble solution we want!



Let's look at some of the different date formats that SAS provides. This table shows today's date displayed with these various formats. You could also apply a format to display the quarter, the month, the week, the weekday, and on and on. These are really helpful for dates, but there are lots of times when SAS does not provide what you need. In those cases, we can use PROC FORMAT to define our own format.


PROC FORMAT


PROC FORMAT can help us...
... Add nice labels (including missing data!)
...Check for valid values
...Group or re-order values
...Change numeric to character
...Apply a special look to the output (e.g. phone numbers, %)
...Add special characters
...Add highlighting/color!
...Create new variable in DATA step.



Here is the general syntax. You can opt to store your formats in a SAS Library with the LIBRARY option. I generally do not take that approach. Instead, I put all of my formats in a separate syntax file and read it in each session using an %INCLUDE. To define a format to be used with numeric variables, use VALUE then the format name (I try to end the format name in "f" or "fmt" to remind myself it's a format), then the specific mappings of values to my format labels. That's really all a format is – a mapping. We can define formats for character variables using a $ and quoting the values to be mapped. SAS also offers PICTURE formats which can change how variables look, such as turning a 10 digit number into a phone number. You can also create a PICTURE format to help make your percentage outputs nicer/have the percent sign. It's especially cool because you can tell the format to round to a certain place and you can even specify a multiplier (not shown). You can specify how many places to hold with 0's and 9's (any number actually, but 0 and 9 are most common). 0's will drop the leading 0s and 9s will not.

Picture Format

Value	Picture format	Formatted value
8.5	"999.9%"	008.5%
8.5	"009.9%"	8.5%
8.5	"000.0%"	8.5%
0.09	(ROUND) "009.9%"	0.1%
0.09	(ROUND) "000.0%"	1%

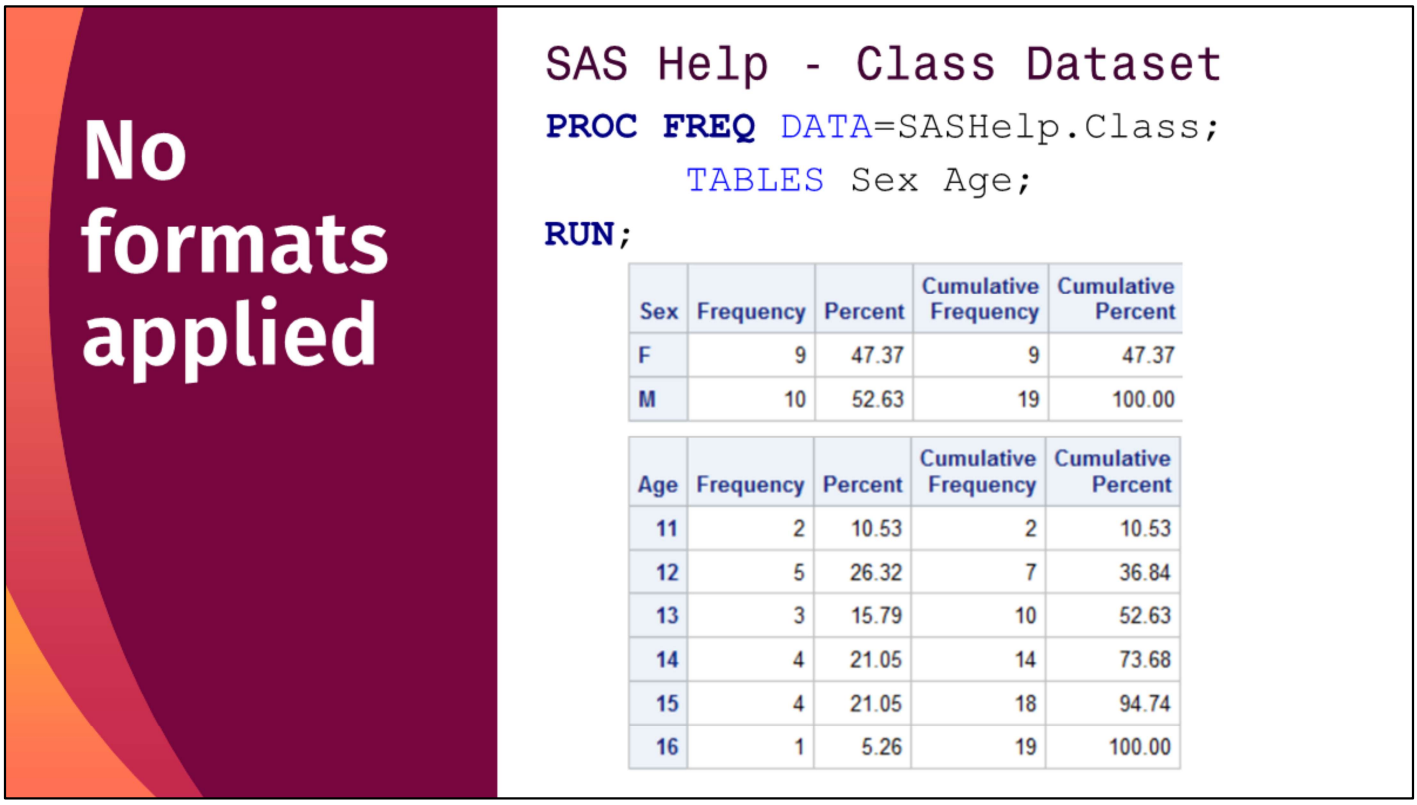
Let's look at some examples of how the 0's and 9's do different things. In the first example, using all 9's leads to funny looking output. On the other hand, using all 0's can also lead to weird things happening. A good rule-of-thumb is to always use 9's after a decimal point. If you don't, the format might do weird things like in the last example.

So many ways to assign formats

You can assign....	Value	Formatted Value
Single value	1	= "Yes"
Multiple values	"SA", "A"	= "Agree or strongly agree"
Range of values	13-19	= "Teenager"
Exclusive range	1- <100	= "Less than 100"
Exclusive range	19.99 < - 29.99	= "20s"
Out of range	OTHER	= "Miscoded or missing"
Extreme low	LOW – 31.99	= "Freezing temperature"
Extreme high	95-HIGH	= "Too hot – stay in the AC"

Here are some ways formats can be assigned. Note the keywords towards the end – OTHER (catches all values not already mapped, including missing), LOW (which includes the lowest non-missing number though the upper end you specify), and HIGH (which captures the highest value).

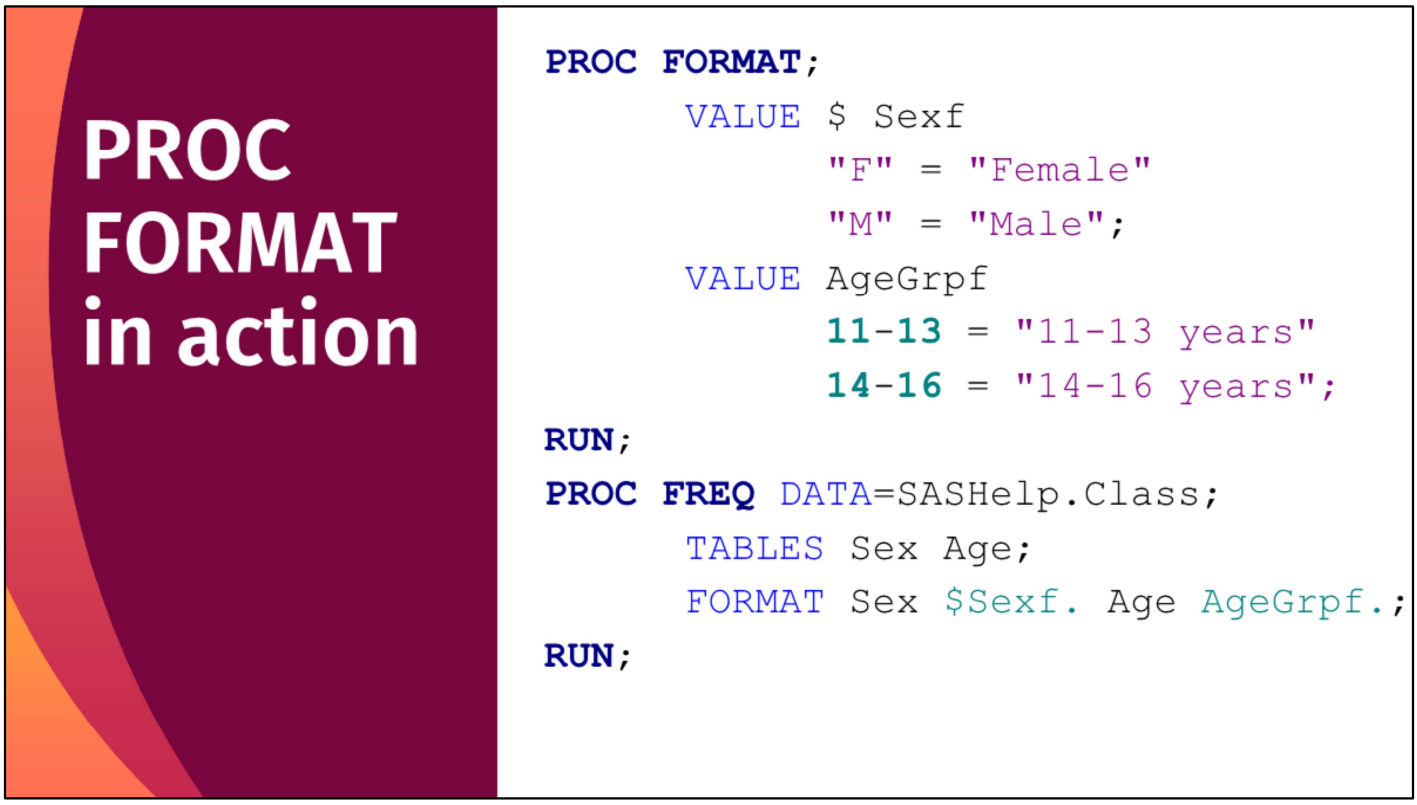

The Examples!
{Finally!}
14



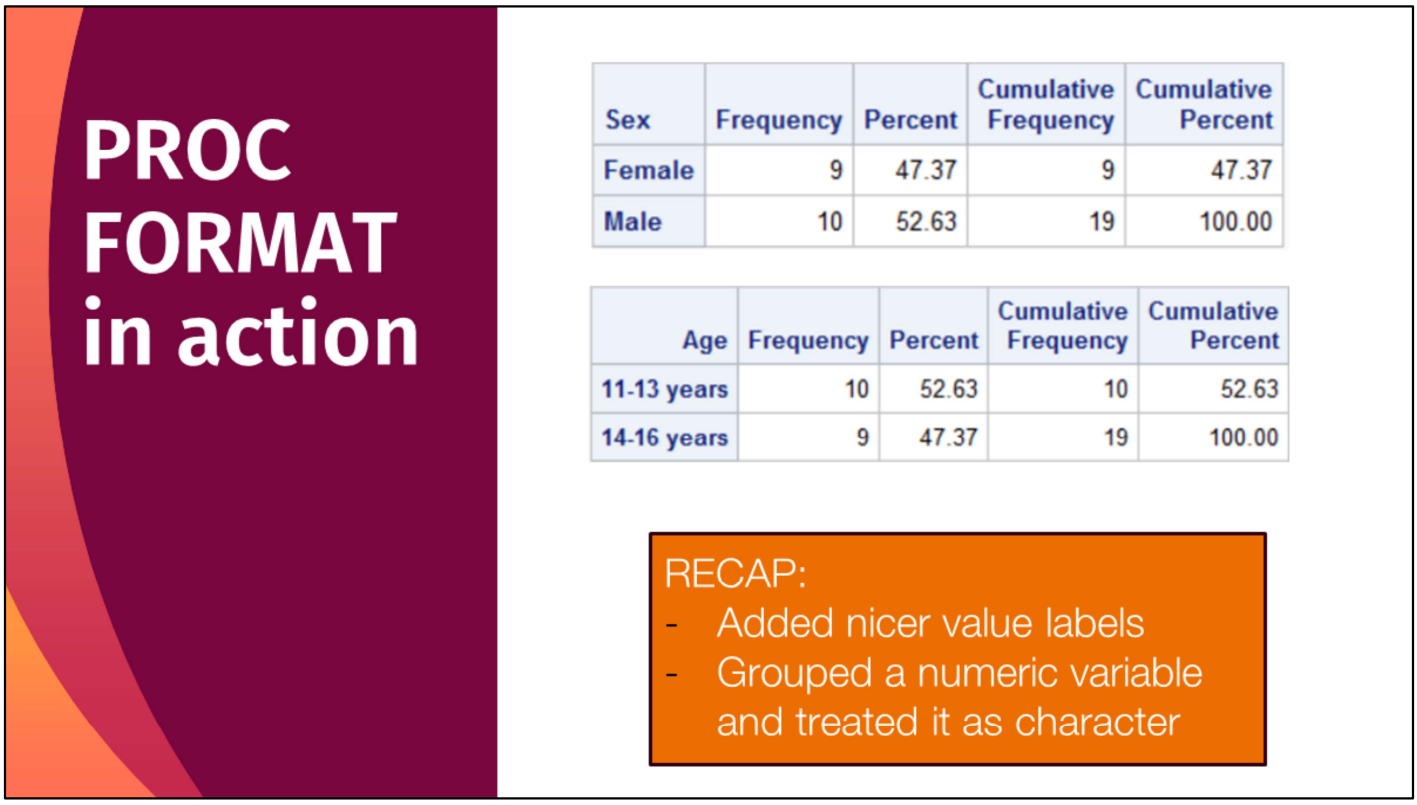
Sex	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	9	47.37	9	47.37
M	10	52.63	19	100.00

Age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
11	2	10.53	2	10.53
12	5	26.32	7	36.84
13	3	15.79	10	52.63
14	4	21.05	14	73.68
15	4	21.05	18	94.74
16	1	5.26	19	100.00

The examples use the SAS Help Class dataset. Here are FREQ output tables for 2 variables – sex and age. We might want nicer labels on sex variable and might want to group ages. PROC FORMAT can do both of these things.



Here we create a format to add better labels on Sex – it's character, so use the $. Also define a numeric format for age that creates 2 age groups. Then we run the same PROC FREQ and apply our new formats to the respective variables.



And here is that output – nicer labels and summarized age groups! Even better, SAS now treats these groupings as the analysis level without having to create such a variable in our dataset. For instance, we could run a chi square test on sex*age, and it will be run on these 2 age categories.

PROC FORMAT in action

```
PROC FORMAT;
        VALUE AgeGrpf
                11-13 = "11-13 years"
                14-16 = "14-16 years";
RUN;
PROC MEANS DATA=SASHelp.Class MAXDEC=1;
        VAR Height;
        CLASS Age;
        FORMAT Age AgeGrpf.;
RUN;
```

Analysis Variable : Height

Age	N Obs	N	Mean	Std Dev	Minimum	Maximum
11-13 years	10	10	59.0	4.3	51.3	65.3
14-16 years	9	9	66.0	3.1	62.5	72.0

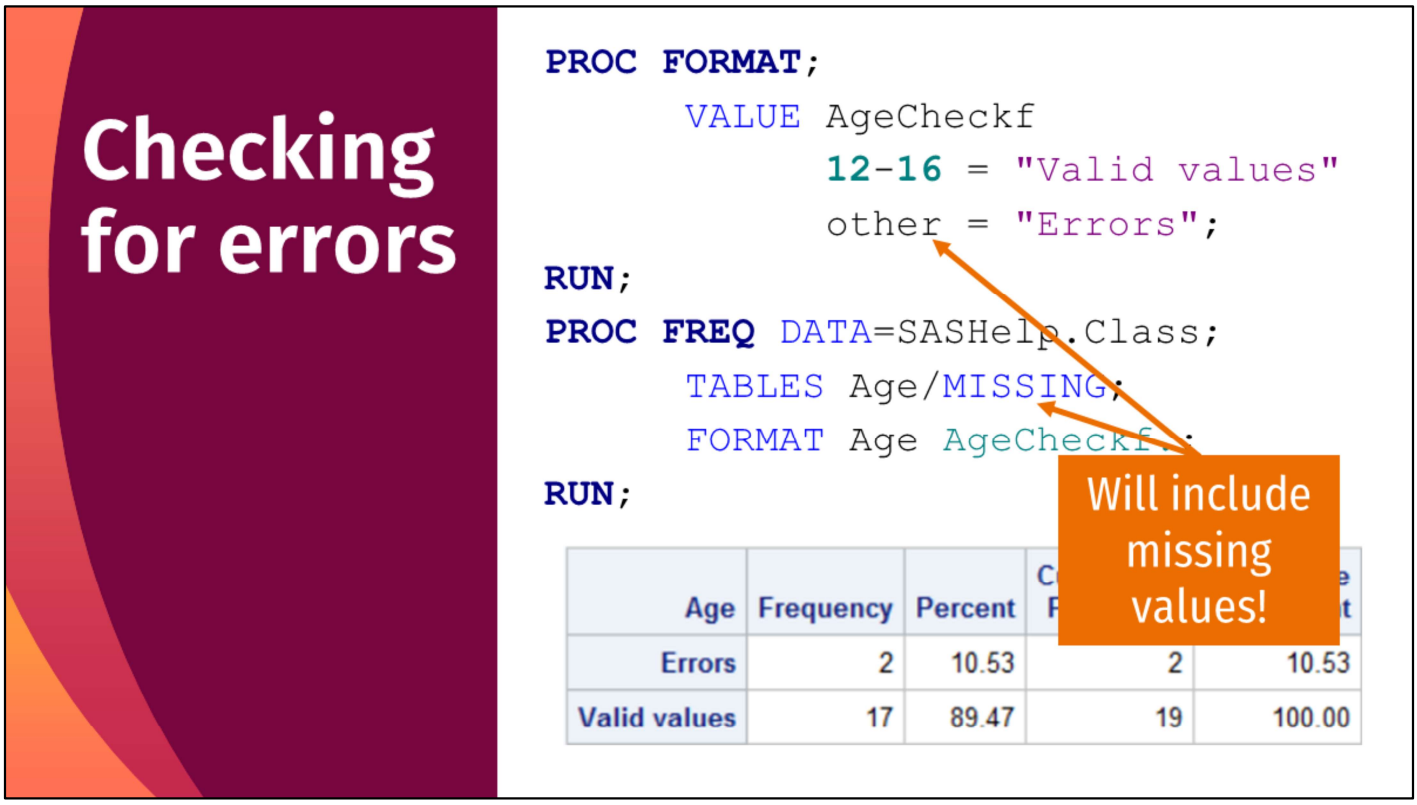
In this example, we use the age format to get summaries of height by age group. We haven't changed the underlying data – it's still a numeric age field, but SAS is treating it as categorical. Easy way to get summaries without having to add variables to the dataset.

Checking for errors

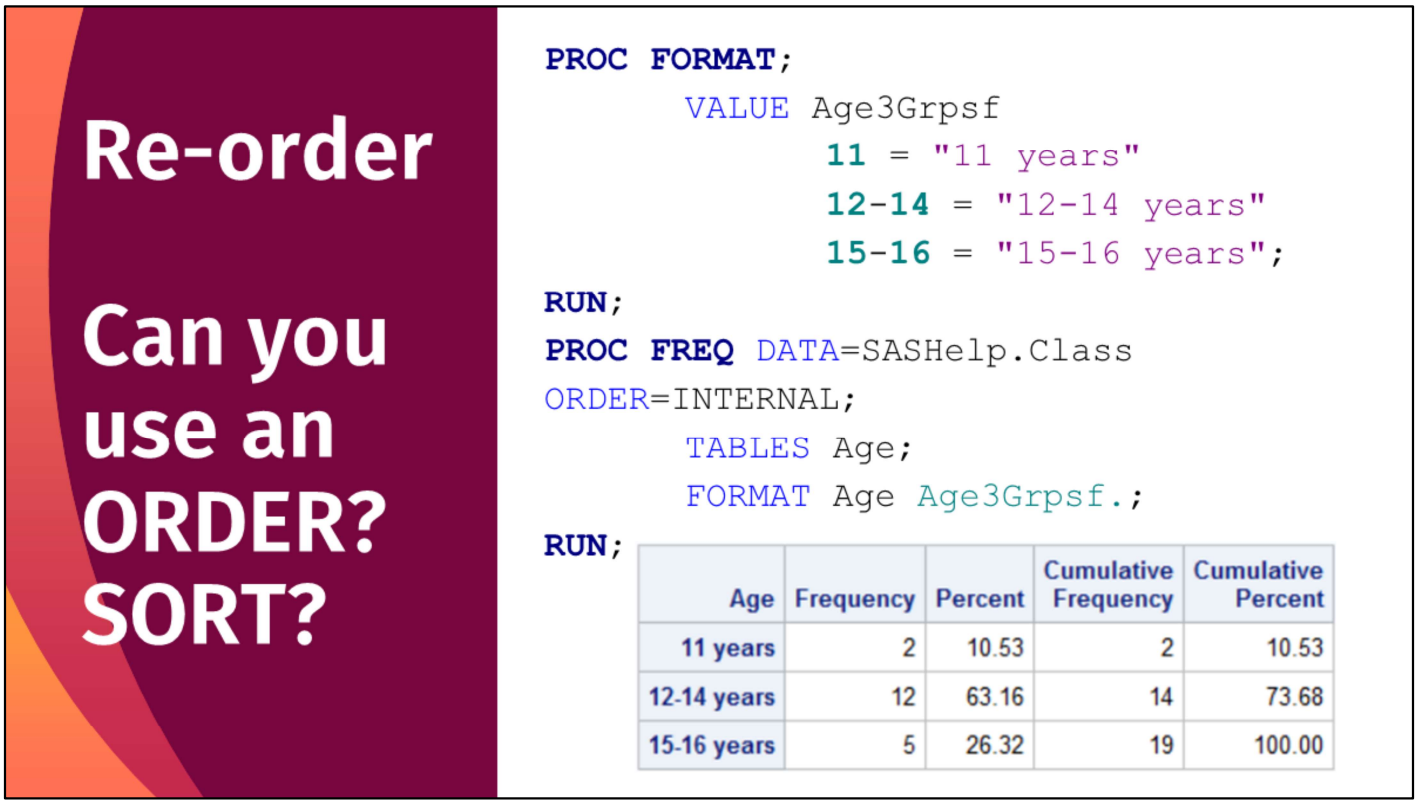
```
PROC FORMAT;
      VALUE Age2Grpsf
            13-14 = "13-14 years"
            15-16 = "15-16 years";
RUN;
PROC FREQ DATA=SASHelp.Class;
      TABLES Age;
      FORMAT Age Age2Grpsf.;
RUN;
```

Age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
11	2	10.53	2	10.53
12	5	26.32	7	36.84
13-14 years	7	36.84	14	73.68
15-16 years	5	26.32	19	100.00

We can also use formats to check for errors. In this case, ages 11 and 12 are not mapped and show up as their own levels in the output table. This could indicate a data collection issue (if 11 and 12 year olds were not supposed to be in the data) or a programming issue (where 11 and 12 were not mapped). Either way, it's clear that 11 and 12 need some attention.



We can also use formats to more systematically find errors by using the “OTHER” to see how many values there are outside of the expected range. This approach will include missing values in the “Errors” category as long as the MISSING option is specified in the PROC FREQ.



Age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
11 years	2	10.53	2	10.53
12-14 years	12	63.16	14	73.68
15-16 years	5	26.32	19	100.00

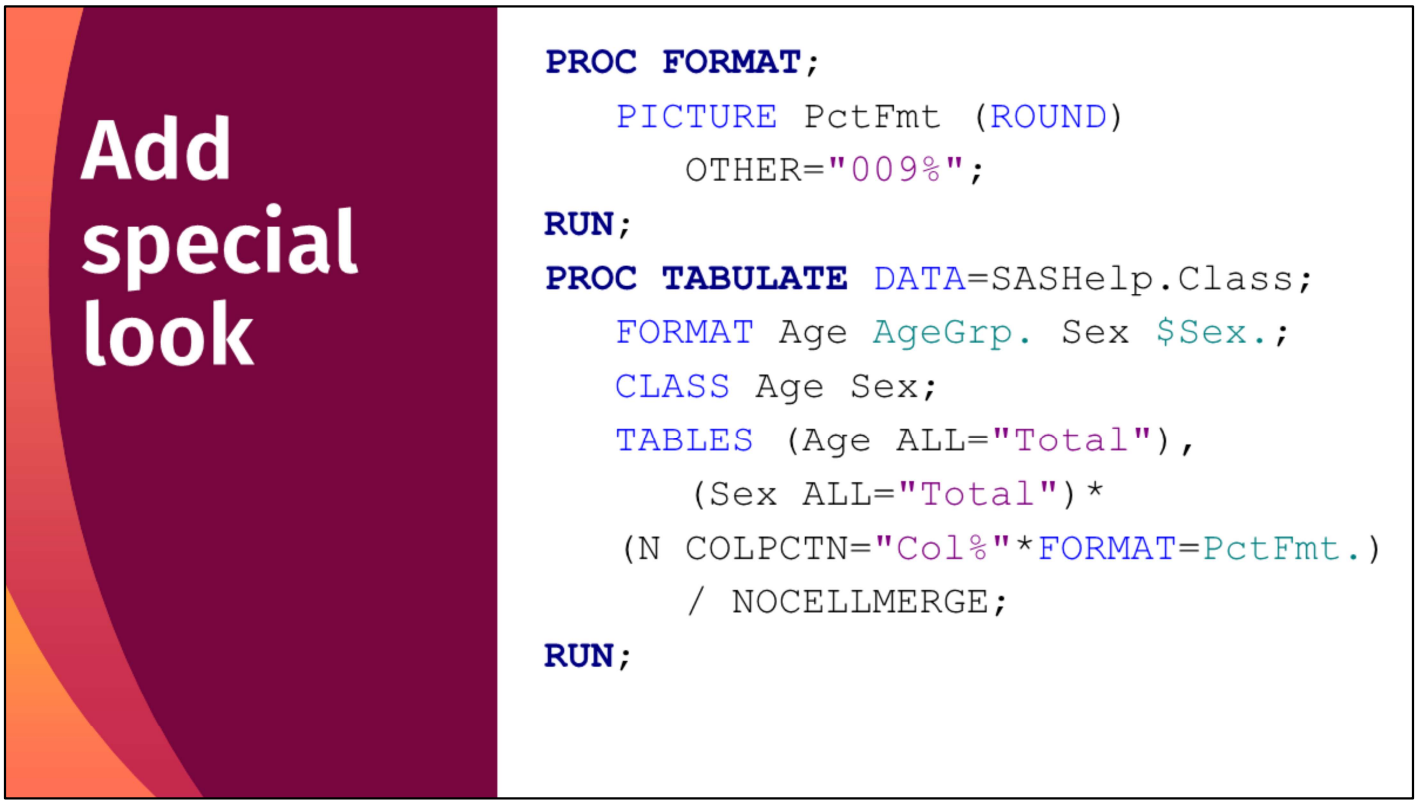
You can also use PROC FORMAT to re-order output. You might not need that – you might be able to use an ORDER option or a PROC SORT with the ORDER option. Here we see the default INTERNAL order.

Re-order

```
PROC FORMAT;
   VALUE Age3Grpsf
   11 = "2) Middle risk (11 years)"
   12-14 = "1) Most at risk (12-14 years)"
   15-16="3) Least at risk (15-16 years)";
RUN;
PROC FREQ DATA=SASHelp.Class
ORDER=FORMATTED;
       TABLES Age;
       FORMAT Age Age3Grpsf.;
RUN;
```

Age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1) Most at risk (12-14 years)	12	63.16	12	63.16
2) Middle risk (11 years)	2	10.53	14	73.68
3) Least at risk (15-16 years)	5	26.32	19	100.00

If those quick fixes (SORT/ORDER) don't work, an easy way to have PROC FORMAT help with re-ordering is to just add a number to start each format label…. Kind of like making your own numbered list. The output isn't gorgeous, but it's not terrible. The bottom line: it's a pretty easy way to get custom ordering in your output and allows me to get this request done and move on with my day. And yes, this example is totally contrived just to demonstrate the ordering. ☺



Here is an example showing a PICTURE format – adding a percent formatting to a PROC TABULATE.

```
PROC FORMAT;
      PICTURE PctFmt (ROUND)
            OTHER="009%";
RUN;
```

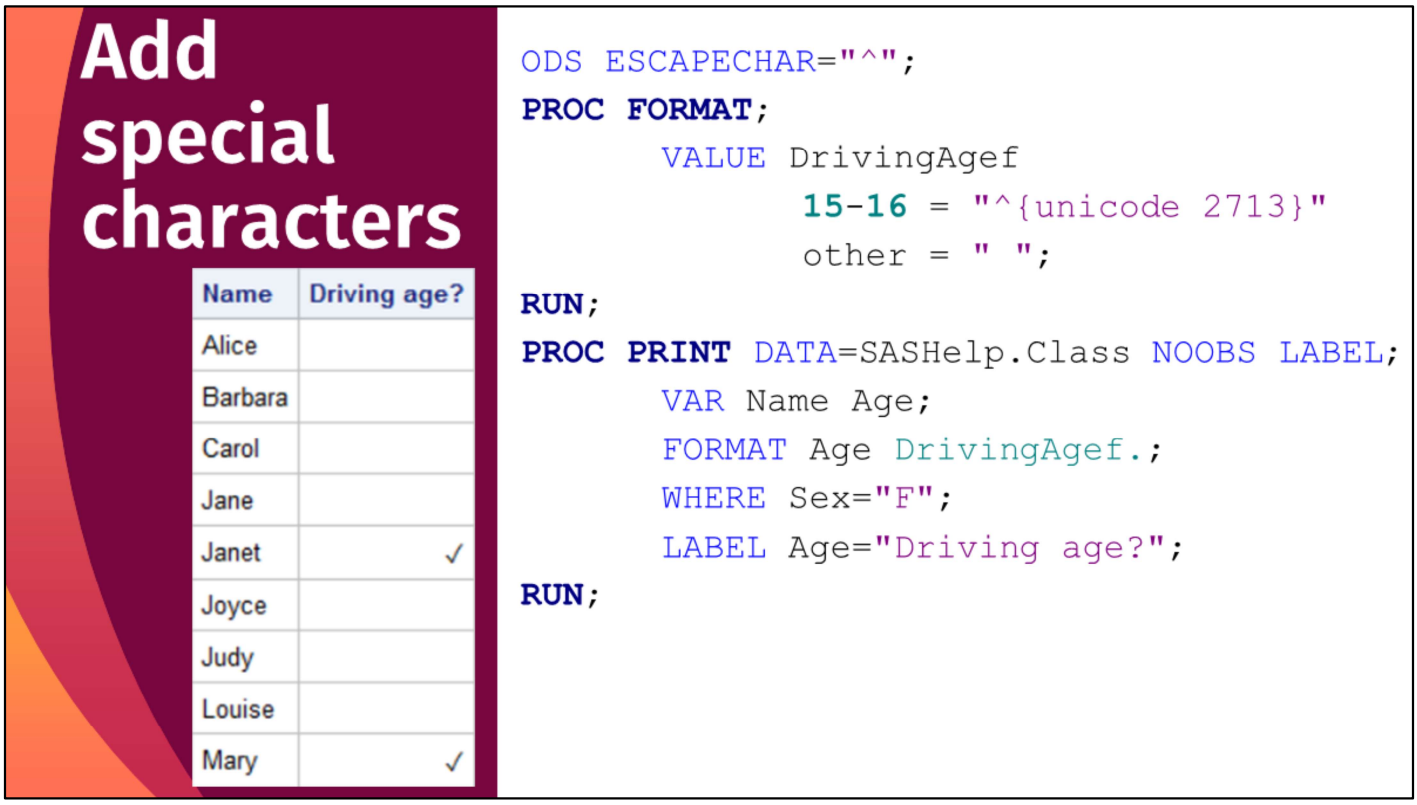
No format applied to percentages

	Sex				Total	
	Female		Male			
	N	Col%	N	Col%	N	Col%
Age						
11-13 years	5	55.56	5	50.00	10	52.63
14-16 years	4	44.44	5	50.00	9	47.37
Total	9	100.00	10	100.00	19	100.00

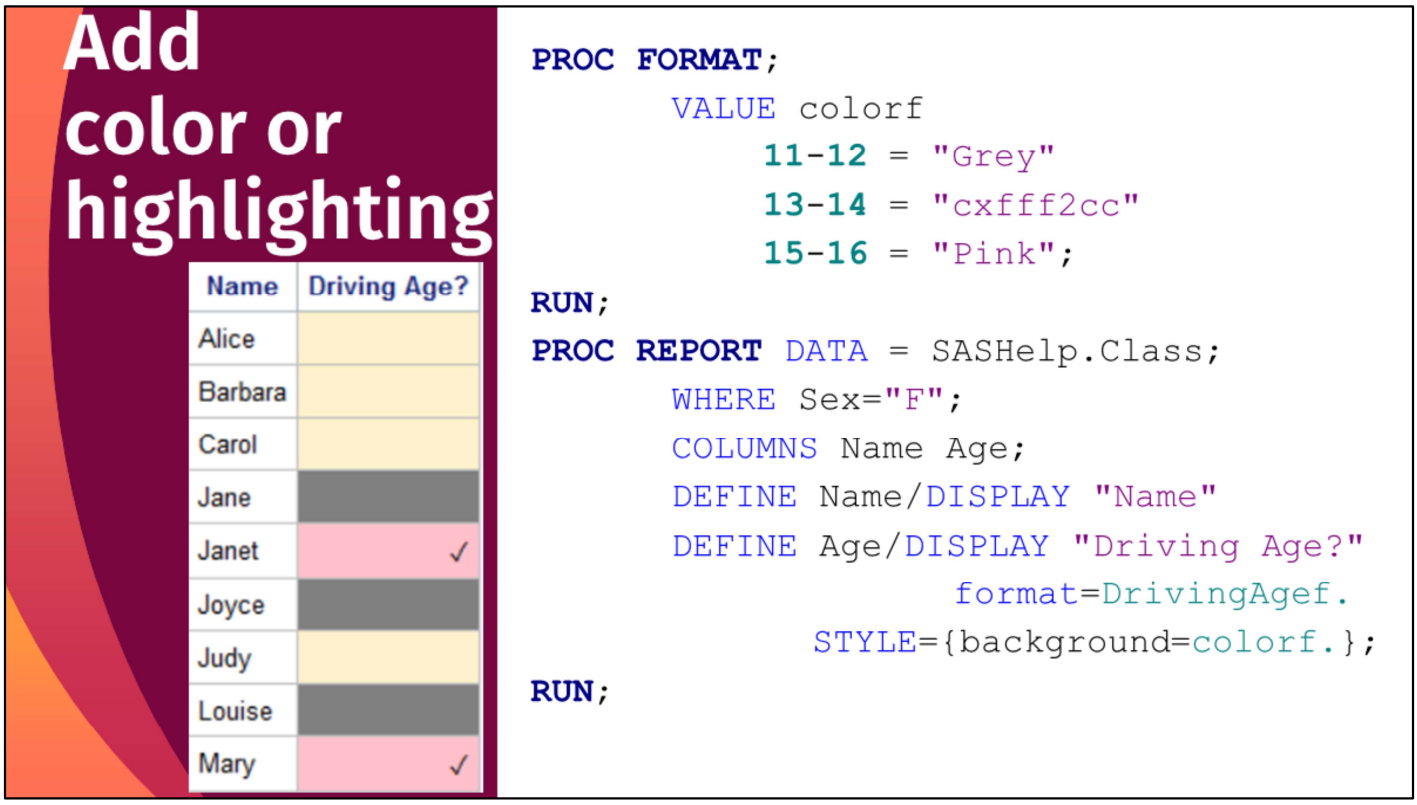
Format applied

	Sex				Total	
	Female		Male			
	N	Col%	N	Col%	N	Col%
Age						
11-13 years	5	56%	5	50%	10	53%
14-16 years	4	44%	5	50%	9	47%
Total	9	100%	10	100%	19	100%

Here we see the unformatted and formatted versions.

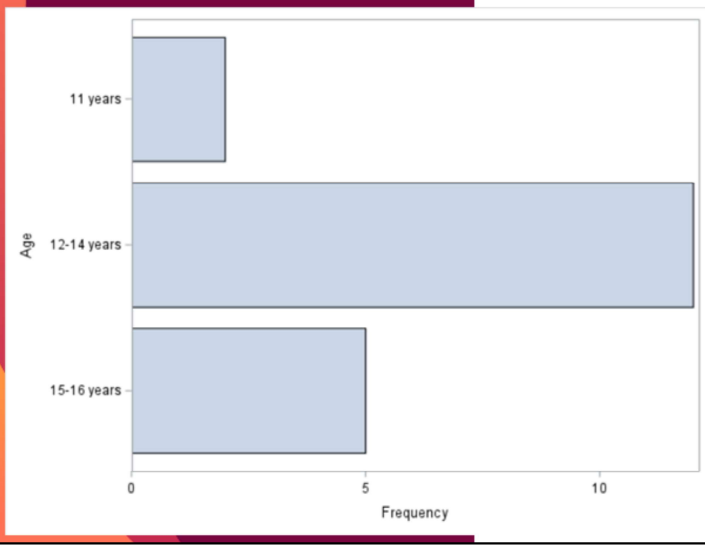


Another fun way to use PROC FORMAT is to add special characters, like Unicode characters. In this case we're translating age groups to check marks. We're creating a more visual list of females of driving age. This works because we're pairing the FORMAT with the ODS ESCAPECHAR. This escape character (the ^), tells SAS to pay attention because we want to do a special formatting. In this case, we're assigning a Unicode character. Unicode characters and their 4-digit codes can be found online through an internet search. Depending on the output destination, the escape character and Unicode character might not resolve, though HTML, PDF, Excel, and many other destinations are fine. You could take this to the next level by applying other style elements (like a different font size, color, justification, etc).



Here we take that same example and shade the cells based on age. That's right – PROC FORMAT can be used to assign cell colors in a PROC REPORT (and a PROC TABULATE). Even though I picked ugly colors, the concept is cool because I'm applying 2 formats to the same variable. Doing it this way, only the cells in the column being DEFINEd will be colored. If you want to color the whole row, you need to use a COMPUTE block with a row style element. This FORMAT approach is meant to be a quick work-around.

Graphing!




```
PROC FORMAT;
        VALUE Age3Grpsf
                11 = "11 years"
                12-14 = "12-14 years"
                15-16 = "15-16 years";
RUN;
PROC SGPLOT DATA=SASHelp.Class;
        HBAR Age;
        FORMAT Age Age3Grpsf.;
RUN;
```

You can also apply formats to graphing procedures. This graph isn't pretty or overly meaningful, but hopefully demonstrates the proof of concept that we can do it.

Add new variable to dataset

```
DATA tempClass;
    SET SASHelp.Class;

    FORMAT AgeGroup $15.;
    AgeGroup=PUT(Age,AgeGrpf.);
RUN;
```

Name	Sex	Age	Height	Weight	AgeGroup
Alfred	M	14	69	112.5	14-16 years
Alice	F	13	56.5	84	11-13 years
Barbara	F	13	65.3	98	11-13 years
Carol	F	14	62.8	102.5	14-16 years
Henry	M	14	63.5	102.5	14-16 years
James	M	12	57.3	83	11-13 years
Jane	F	12	59.8	84.5	11-13 years
Janet	F	15	62.5	112.5	14-16 years
Jeffrey	M	13	62.5	84	11-13 years

One last example is creating a new variable in a dataset using a format. Here we define a format called AgeGrpF. Then we use a PUT function to assign and store the formatted age group in a variable called AgeGroup. No If-Then-Else coding needed. It's not always a time saver to do it this way, but it can be.

Thanks!

Any questions?

You can find me at:

- SRichter@RichterStats.com

29