



Better (Prompt) Engineering: Leveraging ChatGPT to get started with SAS Coding SAS Guided Demo

Purpose

This SAS Guided Demo introduces programmers to the practical use of large language models (LLMs), such as ChatGPT, as a partner in code development. We walk through three main cases where LLMs can be particularly useful for those learning new code: (1) in generating a first draft of code, which gets you 80-90% of the way to a working solution, (2) understanding complex code written by someone else, and (3) supporting code documentation, to make your work clearer and easier to maintain. Moreover, we start with best practices on how to interact with LLMs to optimize the output generated.

Software

This SAS Guided Demo uses SAS Viya for Learners, version 2025.03LTS, and the free version of ChatGPT.

Prerequisites

While not mandatory, it is useful if participants have some exposure to SAS coding in SAS Model Studio and in working with ChatGPT.

Table of Contents

Better (Prompt) Engineering:.....	1
Leveraging ChatGPT to get started with SAS Coding	1
Purpose	1
Software.....	1
Prerequisites	1
Part 1: LLM best practices for better prompt engineering.....	4
High-Level Notes (from a human)	4
Detailed Advice from ChatGPT5.....	4
Part 2: SAS Guided Demo.....	6
Learning Objectives (aka – what I’m trying to trick you into learning):	6
GitHub Links	6
Disclaimers	6
Start the adventure in SAS Studio – and setup the project	6
Shortcuts to assigning libraries	8
ChatGPT Prompt.....	11
ChatGPT Response	11
ChatGPT Prompt.....	11
ChatGPT Code	12
ChatGPT Notes	12
Read in a file from GitHub.....	13
ChatGPT Prompt.....	13
ChatGPT Code	14
ChatGPT Notes	14
Next Step: Better understand your data	16
ChatGPT Prompt.....	16
ChatGPT Code	16
ChatGPT Notes	19
Revised ChatGPT Prompt	19
Revised ChatGPT Code	20
Revised ChatGPT Details	21
Next Step: Collapse the data using SQL. Then ask ChatGPT what the heck happened.....	23
ChatGPT Prompt.....	23

ChatGPT Response	24
Summarize our newly created data set	27
ChatGPT Prompt.....	27
ChatGPT Code	27
ChatGPT Notes	29
Nearing the end: let's plot state-level trends over time!.....	29
ChatGPT Prompt.....	29
ChatGPT Code	29
ChatGPT Additional Details	30
Revised ChatGPT Prompt	33
Revised ChatGPT Response	34
Revised ChatGPT Code	34
Revised (2) ChatGPT Prompt	35
Revised (2) ChatGPT Code.....	35
Revised (2) ChatGPT Details	36
Final Trick: Documentation!.....	37
ChatGPT Prompt.....	38
ChatGPT Code	38
ChatGPT Notes	40
Recap.....	41

Part 1: LLM best practices for better prompt engineering

High-Level Notes (from a human)

- Be nice to the machines
 - Seriously. Why? <https://www.linkedin.com/pulse/nice-ai-seriously-heres-why-amy-kauffman-cjpfc/>
 - And be conversational.
- Take time to explain the context to the LLMs
 - Why?
 - Like modeling → garbage in = garbage out
 - In other words, bad question → bad answers
- You'll likely get a slightly different answer each time from ChatGPT.
 - And that's ok.
- Be ok with iterating.
 - Most of us don't get things perfect the first time we do them.
 - Why should the machines?
 - And the code may never be perfect.
 - But, it can get you about 80-90% of the way there.

Detailed Advice from ChatGPT5

- 1. Be Kind and Conversational**
 - **Kindness pays off.** You don't need to flatter the machine — but a friendly, clear tone helps the model stay cooperative and creative.
 - **Conversational prompts** ("Let's try this," "Can you explain why...") work better than commands ("Do this," "Write code now").
 - Think of ChatGPT as a *teammate*, not a search engine.
- 2. Explain the Context**
 - LLMs don't know your intent unless you tell them. The more you explain your project or goal, the better your results.
 - For example: "I'm teaching students how to use PROC LOGISTIC in SAS Viya" yields far better code than "write PROC LOGISTIC."
 - Remember: like modeling, **garbage in → garbage out**.
- 3. Be Clear and Specific**
 - Define your **goal** ("I want a SAS program to summarize numeric variables and visualize the results").
 - Define your **scope** ("Just the code, no commentary" or "Include comments explaining each step").
 - Mention your **environment** (SAS Viya 4, SAS OnDemand for Academics, Workbench, etc.) since syntax can differ slightly.
- 4. Use Concrete Examples**

- Share **sample data** (even 5 rows) or describe the structure: “Dataset HHS.HHS_OTJ_Raw with variables like FIPS, year, and UE_Women.”
- If possible, describe **what success looks like**: “I want a heat map showing unemployment rates by state and year.”

5. Build in Steps, Not Walls

- Break large requests into smaller ones: “First, import the data. Next, summarize. Then visualize.”
- It’s often better to **iterate** than to ask for a full project at once.
- Think of ChatGPT as helping you debug or refine one stage at a time.

6. Expect Variability — and Iterate

- Each response may be slightly different — and that’s okay.
- Use the first answer as a *starting point*, not a verdict.
- Refining your prompt (“Good start, but add PROC UNIVARIATE and label the axes”) usually leads to the perfect version.

7. Reference and Reuse

- Point back to earlier parts of your conversation: “Use the dataset from above” or “Modify the previous PROC SQL example.”
- LLMs perform best when the conversation stays connected and logically sequenced.

8. Respect Boundaries

- Don’t share personal, proprietary, or confidential data.
- When possible, use **mock or generalized examples**.
- This keeps both your organization and your model safe.

9. Review and Reflect

- Always read the output carefully before running code.
- LLMs can make small syntax errors or assumptions — especially in SAS code.
- Treat ChatGPT as a **co-pilot**, not an autopilot.

10. Learn from Documentation (and Let ChatGPT Guide You There)

- Combine ChatGPT’s explanations with official sources (e.g., SAS Help Docs, Viya Learn Portal, SAS Communities).
- You can even ask: “Show me how this compares to the SAS documentation.”
- The best workflows pair **LLM + official docs** for understanding *and* correctness.

Bonus Principle: Co-Design, Don’t Copy

Use ChatGPT as a creative partner. You’re not just asking for code — you’re co-designing ideas, lessons, and workflows.

Think: “Let’s build this together,” not “Do my homework.”

Part 2: SAS Guided Demo

Learning Objectives (aka – what I’m trying to trick you into learning):

- How easy it is to create a basic SAS program using code generated from ChatGPT
- How ChatGPT won’t always give you a perfect answer – and you’ll need to iterate
- How ChatGPT can help you better understand complex code
 - There’s nothing wrong with asking it to explain it to you like you’re a 10-year-old.
- How ChatGPT can serve as a powerful coding partner – helping you write code you already know and guiding you through code you’re learning.

GitHub Links

- SAS On-the-Job HHS Analyst Project: <https://github.com/lincolngroves/SAS-OTJ-HHS>
- Repo for this workshop: <https://github.com/lincolngroves/BetterPromptEngineering>
 - This is where you can directly copy SAS Code used in the program we’ll create together.

Disclaimers

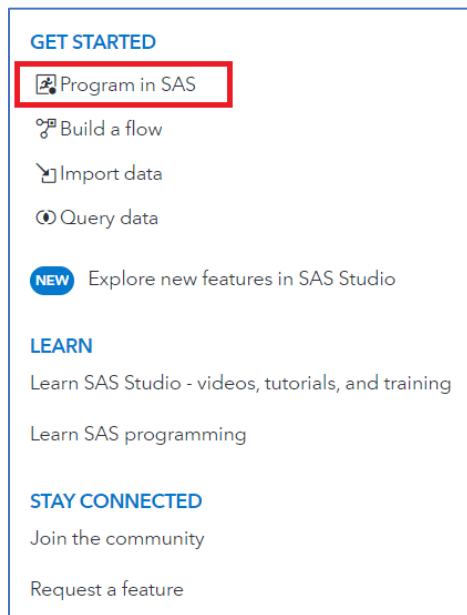
- This SAS Guided Demo was written in SAS Viya 4, version 2025.03 LTS.
- Moreover, I used ChatGPT5 to write this script. But, you can use ChatGPT4-turbo, as it’s now free to use for everyone without an account – and should use similar responses.
Learn more [here](#).

Start the adventure in SAS Studio – and setup the project

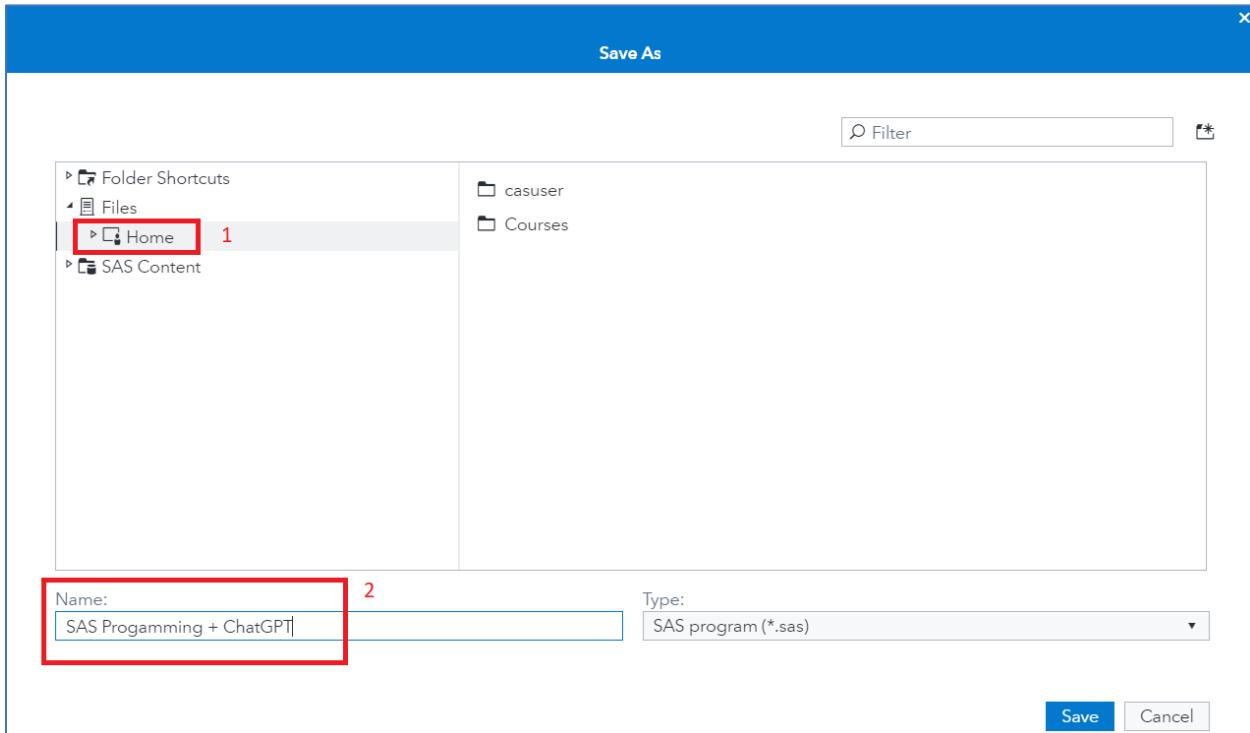
- From the **Applications menu**, select **Develop Code and Flows** under Analytics Life Cycle:



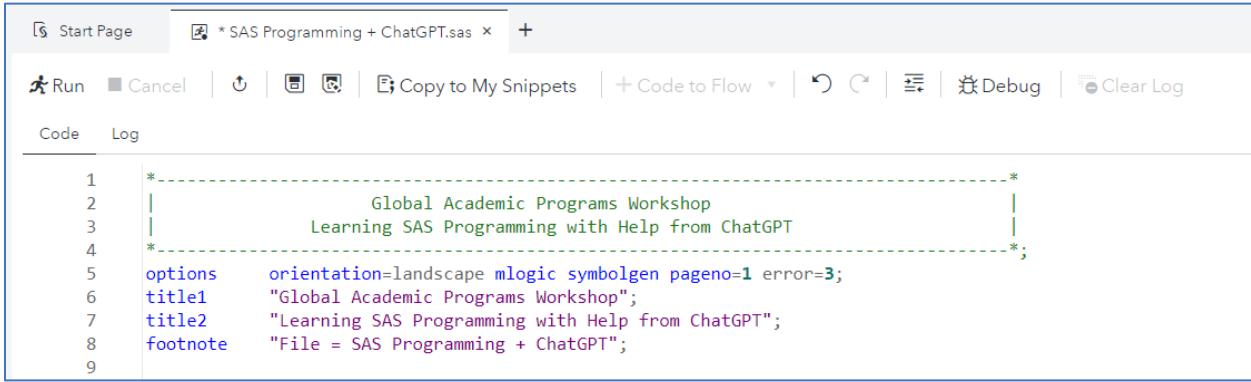
- Under **Get Started**, choose **Program in SAS**



- Click to save your new program in your **Home** directory. And let's call it **SAS Programming + ChatGPT**:



- And I'll say it just one time: *save – and save often* – so you don't lose your work!
- Any good program starts with a nice header and some titles to describe the project and objective. So, create something akin to the following – or simply mirror the example below:

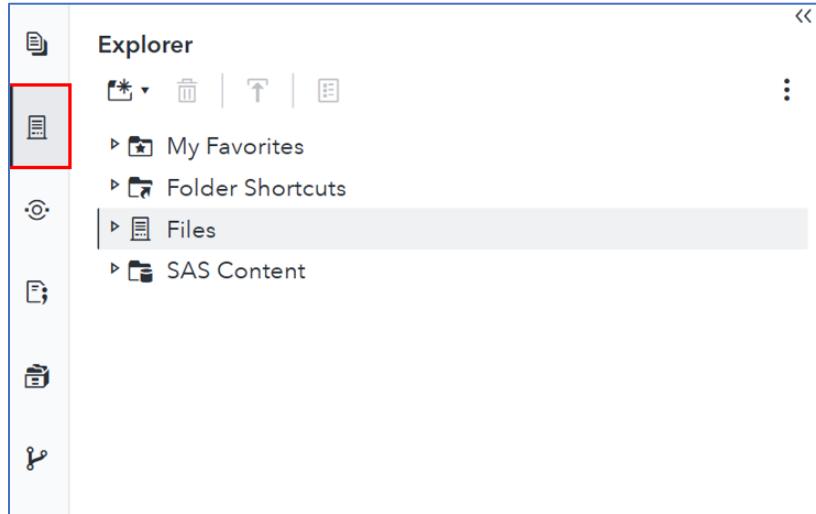


The screenshot shows the SAS Studio interface with a code editor window titled "SAS Programming + ChatGPT.sas". The code is as follows:

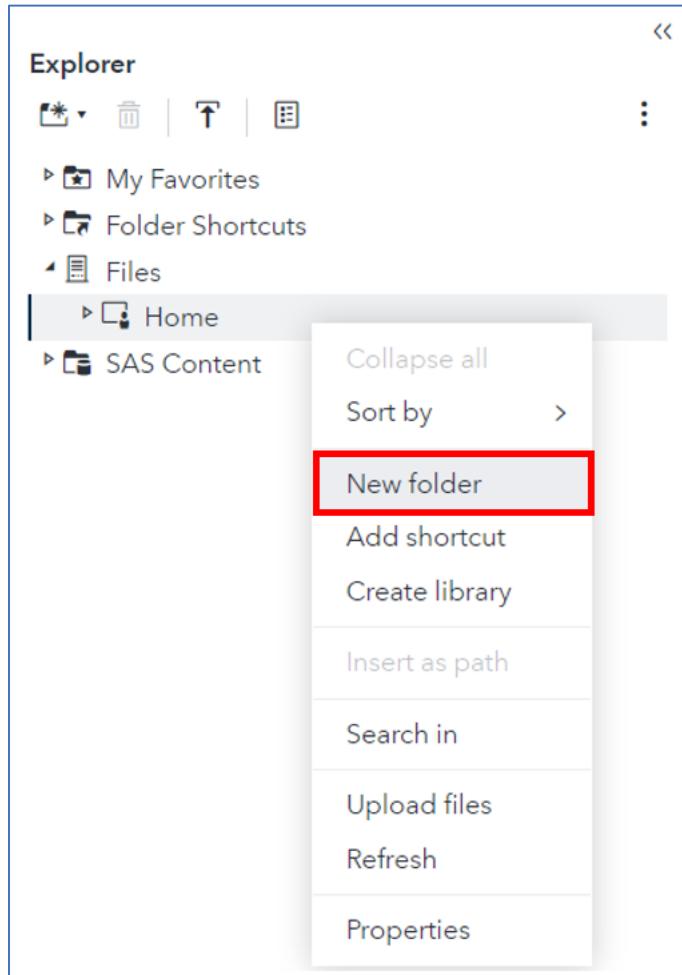
```
1 *-----*
2 |          Global Academic Programs Workshop
3 |          Learning SAS Programming with Help from ChatGPT
4 *-----*
5 options orientation=landscape mlogic symbolgen pageno=1 error=3;
6 title1 "Global Academic Programs Workshop";
7 title2 "Learning SAS Programming with Help from ChatGPT";
8 footnote "File = SAS Programming + ChatGPT";
9
```

Shortcuts to assigning libraries

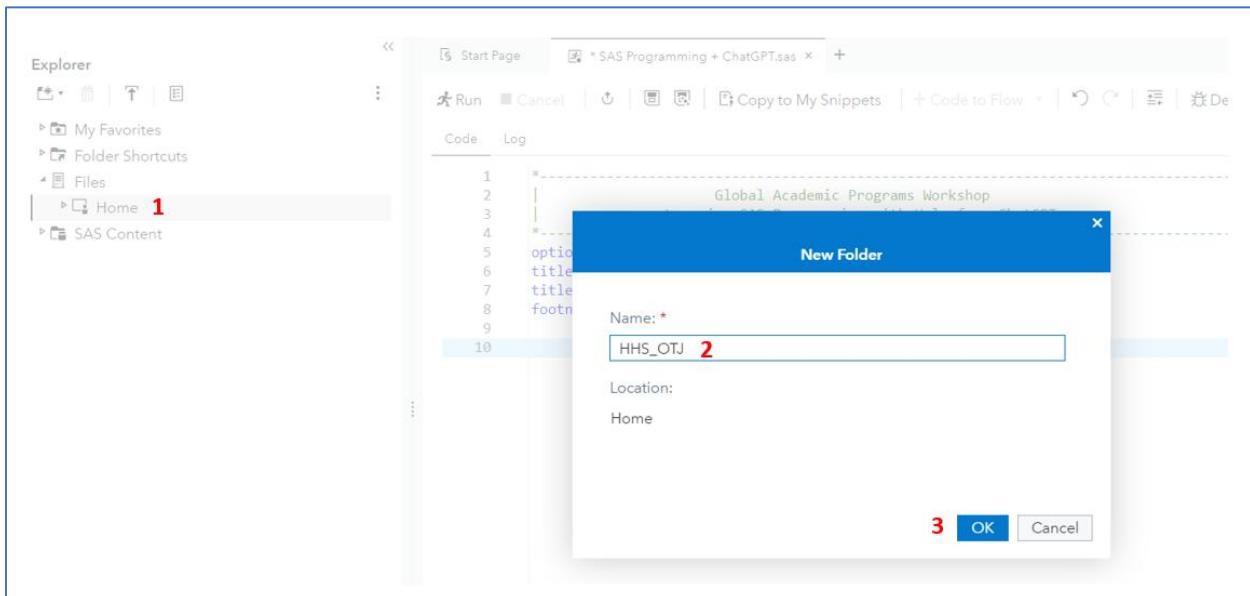
- The next step is to tell SAS where we'd like to save our data. Start by expanding **Explorer** on the left navigation pane:



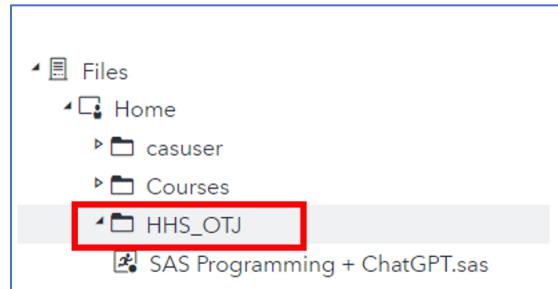
- For this project, let's create a new folder under **Files >> Home** called **HHS_OTJ**. To do so, expand the **Files** folder, then find **Home**. By right-clicking on **Home**, you can select **New folder**, as follows:



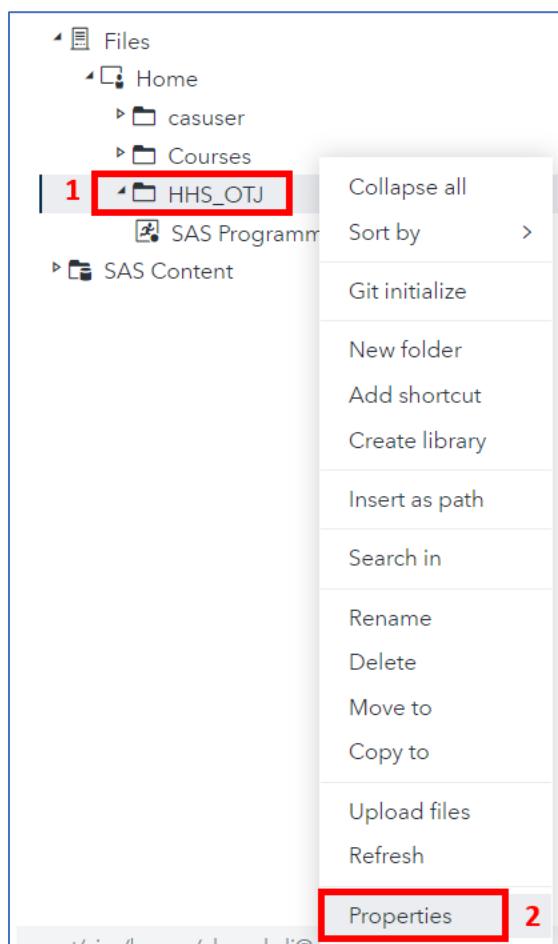
- Then create the **New Folder** titled **HHS_OTJ**:



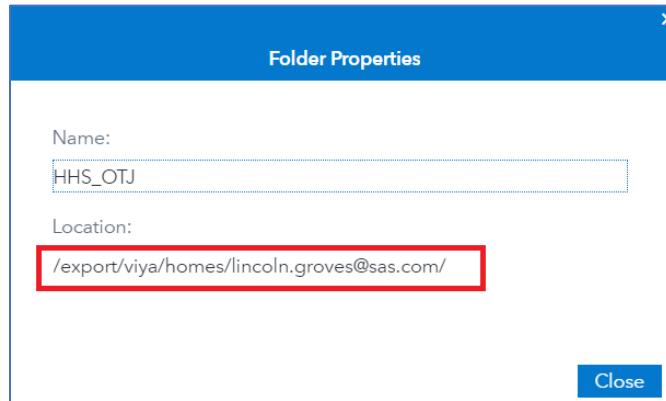
- Check under **Home** to see that a new folder was created. There it is!



- Fun fact: we're all different users – and people. So, our paths on the VFL system will be different. Want proof? Well, right-click on **HHS_OTJ**, then select **Properties**:



- Can you match this?



- Nope. Didn't think so. Because you're not me 😊 So, this could be a useful case for a relative file reference – and a nice opportunity for you to meet your soon-to-be-dear friend, the “~”. Let's ask ChatGPT a bit more about the tilde.

ChatGPT Prompt

I have a question about assigning SAS libraries. Instead of writing out the full path to a library, such as libname HHS
"/export/viya/homes/lincoln.groves@sas.com/HHS_OTJ", how can I use a ~ to shorten the path?

ChatGPT Response

In SAS (especially in UNIX/Linux environments), you generally cannot use ~ as a shortcut for the home directory in a LIBNAME statement like you can in a shell (e.g., Bash). SAS does not automatically expand ~ the way the shell does.

✖ This does NOT work:

```
sas
libname HHS "~/HHS_OTJ";
```

Copy code

This will result in an error like:

```
makefile
ERROR: Invalid physical name.
```

Copy code

- Ummm. Not off to a great start. Because you can use the “~” in SAS Viya for Learners. So, let's refine the prompt a bit.

ChatGPT Prompt

But, you can use the ~ in SAS Viya for Learners. Can you explain to me how it works?

ChatGPT Code

```
libname HHS "~/HHS_OTJ";
```

ChatGPT Notes

You're absolutely right — and thank you for pointing that out! 🍀

In SAS Viya for Learners, you *can* use the ~ (tilde) in file paths like this:

SAS

Copy code

```
libname HHS "~/HHS_OTJ";
```

And it **does work correctly** by resolving ~ to your SAS session's home directory. 🎉

This behavior is specific to **SAS Viya for Learners** (and some Viya environments), where the SAS compute server is configured to **recognize and expand** ~ just like a Unix shell would.

So in your case:

SAS

Copy code

```
libname HHS "~/HHS_OTJ";
```

is valid, and it maps to:

SAS

Copy code

```
/export/viya/homes/lincoln.groves@sas.com/HHS_OTJ
```



- So, that's a sweet hack! Add that line of code to your program. For example, here:

```
1 *-----*  
2 |          Global Academic Programs Workshop  
3 |          Learning SAS Programming with Help from ChatGPT  
4 *-----*  
5 options orientation=landscape mlogic symbolgen pageno=1 error=3;  
6 title1 "Global Academic Programs Workshop";  
7 title2 "Learning SAS Programming with Help from ChatGPT";  
8 footnote "File = SAS Programming + ChatGPT";  
9  
10 libname HHS "~/HHS_OTJ";  
11
```

- You now have a SAS library where you can store data SAS data generated in your program. Submit that program using the **Run** icon and make sure it all runs smoothly:

The screenshot shows the SAS Studio interface. On the left, the 'Code' tab is selected, displaying SAS code. The code includes options for orientation, titles, and a libname statement. A red box labeled '1' highlights the 'Run' button at the top of the code editor. On the right, the 'Log' tab is selected, showing the execution results. A red box labeled '2' highlights the log output area. The log shows the generated preamble, the successful assignment of libref HHS, and the postamble. It also notes that libref HHS was successfully assigned to the physical path /export/viya/homes/lincoln.groves@sas.com/HHS_OTJ.

```

1   *----- Global Academic Programs Workshop
2   | Learning SAS Programming with Help from ChatGPT
3   |
4   *
5   options orientation=landscape mlogic symbolgen pageno=1 error=3;
6   title1 "Global Academic Programs Workshop";
7   title2 "Learning SAS Programming with Help from ChatGPT";
8   footnote "File = SAS Programming + ChatGPT";
9
10 libname HHS "/~/HHS_OTJ";
11

```

Log

- Errors (0)
- Warnings (0)
- Notes (9)

There are no messages.

```

1 /* region: Generated preamble */
79
80 *
81 *----- Global Academic Programs Workshop
82 | Learning SAS Programming with Help from ChatGPT
83 *
84 options orientation=landscape mlogic symbolgen pageno=1 error=3;
85 title1 "Global Academic Programs Workshop";
86 title2 "Learning SAS Programming with Help from ChatGPT";
87 footnote "File = SAS Programming + ChatGPT";
88
89 libname HHS "/~/HHS_OTJ";
NOTE: Libref HHS was successfully assigned as follows:
  Engine:      V9
  Physical Name: /export/viya/homes/lincoln.groves@sas.com/HHS_OTJ
90
91
92 /* region: Generated postamble */
103

```

- All good on my end! Hopefully it is for you too!

Read in a file from GitHub

- Now we are ready to pull in data from a GitHub repo. For this guided demo, the data come from a SAS On-the-Job asset that I created which explores the labor supply by women before, during, and after-COVID. The longer SAS On-the-Job storytelling is that students pretend that they are starting their first day as Public Policy analysts and are asked to examine whether there are any lingering negative effects from COVID on labor supply by prime aged women, aged 25-54.
 - Spoiler alert: despite doomsday reporting by the U.S. media, the labor market continues to be very strong in the U.S.
- The general GitHub repo for the original SAS On-the-Job project is here: <https://github.com/lincolngroves/SAS-OTJ-HHS>
 - And the .CSV file that we'd like to pull and convert to a SAS dataset is here: https://github.com/lincolngroves/SAS-OTJ-HHS/blob/main/CPS_2015_2023_ltd.csv
 - As the .CSV file name suggests, the data spans from 2015 to 2023.
- Now let's see how ChatGPT can help us write some code to read in the data.

ChatGPT Prompt

I'd now like to write some SAS code that will help me download a .CSV file from a GitHub page. The data file is found here: https://github.com/lincolngroves/SAS-OTJ-HHS/blob/main/CPS_2015_2023_ltd.csv. Can you please help me write this code so I can load the data into SAS Studio?

ChatGPT Code

```
filename cps url  
"https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/CPS_2015_2023_ltd.csv";  
  
proc import datafile=cps  
    out=work.cps_2015_2023  
    dbms=csv  
    replace;  
    guessingrows=max;  
run;
```

ChatGPT Notes

Absolutely! 🍏

You can download a `.csv` file directly from GitHub into SAS Viya (or SAS Studio) by using the file's `raw` URL, not the webpage link.

Here's how to do it step-by-step:

💡 Step 1: Get the `raw` GitHub link

Your link:

```
ruby Copy code  
https://github.com/lincolngroves/SAS-OTJ-HHS/blob/main/CPS\_2015\_2023\_ltd.csv
```

Change it to the `raw` version:

```
bash Copy code  
https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/CPS\_2015\_2023\_ltd.csv
```

That's the direct file path SAS can read.

Step 2: Use a `filename` statement with the URL access method

Here's the complete SAS code you can paste into SAS Studio:

```
sas                                         ⌂ Copy code

filename cps url "https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/CPS_2015_2023_1.csv";

proc import datafile=cps
    out=work.cps_2015_2023
    dbms=csv
    replace;
    guessingrows=max;
run;
```

What this does

- `filename cps url ...;` tells SAS to fetch the file directly from the web.
- `proc import` reads it as a CSV file.
- `out=work.cps_2015_2023` stores it in your temporary **WORK** library.
- `guessingrows=max` ensures SAS checks all rows to correctly assign data types.

- Boom! Let's make a couple of further refinements because I have preferences.
 - To start, add that code to our SAS program in SAS Studio.
 - Next change the output data file in the proc export statement to **HHS.HHS_OTJ_Raw** – so that we can save the data to that newly created SAS library.
 - Then adjust **guessingrows=100000**, rather than max (this will make the import faster).
- Original code – and those cumulative changes – should yield the following:

```
*-----*
|          Import Data from GitHub           |
*-----*;
filename cps url "https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/CPS_2015_2023_ltd.csv";

proc import datafile=cps
    out=hhs.hhs_otj_raw
    dbms=csv
    replace;
    guessingrows=100000;
run;
```

- Mash that **Run** icon:



- With some starter code from ChatGPT and a few refinements, you now have a data set to work with! And you can find it in the **Explorer** pane, in your **HHS_OTJ** folder:

	@ State_FIP	△ State_Name	□ YearQuarter	△ Race_Ethnic	△ EDUC_LTD	△ Child_Status
1	1	Alabama	2015Q1	Black, Non-Hispanic	< HS	No Children
2	1	Alabama	2015Q1	Black, Non-Hispanic	< HS	No Children
3	1	Alabama	2015Q1	Black, Non-Hispanic	< HS	Older Children
4	1	Alabama	2015Q1	Black, Non-Hispanic	< HS	Older Children
5	1	Alabama	2015Q1	Black, Non-Hispanic	< HS	Older Children
6	1	Alabama	2015Q1	Black, Non-Hispanic	High School Diploma	No Children
7	1	Alabama	2015Q1	Black, Non-Hispanic	High School Diploma	No Children
8	1	Alabama	2015Q1	Black, Non-Hispanic	High School Diploma	No Children
9	1	Alabama	2015Q1	Black, Non-Hispanic	High School Diploma	Older Children
10	1	Alabama	2015Q1	Black, Non-Hispanic	High School Diploma	Older Children
11	1	Alabama	2015Q1	Black, Non-Hispanic	High School Diploma	Older Children
12	1	Alabama	2015Q1	Black, Non-Hispanic	High School Diploma	Child < 5
13	1	Alabama	2015Q1	Black, Non-Hispanic	High School Diploma	Child < 5
14	1	Alabama	2015Q1	Black, Non-Hispanic	High School Diploma	Child < 5
15	1	Alabama	2015Q1	Black, Non-Hispanic	Some College	No Children
16	1	Alabama	2015Q1	Black, Non-Hispanic	Some College	No Children
17	1	Alabama	2015Q1	Black, Non-Hispanic	Some College	No Children
18	1	Alabama	2015Q1	Black, Non-Hispanic	Some College	Older Children
19	1	Alabama	2015Q1	Black, Non-Hispanic	Some College	Older Children
20	1	Alabama	2015Q1	Black, Non-Hispanic	Some College	Older Children
21	1	Alabama	2015Q1	Black, Non-Hispanic	Some College	Child < 5

- Happy data exploring!

Next Step: Better understand your data

- You likely still don't know what this project is all about yet. And that's ok because we're just getting started! So, let's ask ChatGPT to help us better understand our new HHS_OTJ_Raw data set.

ChatGPT Prompt

I'd like to better understand my new SAS data set HHS.HHS_OTJ_Raw. Can you help me write some SAS code which summarizes all the numeric and non-numeric variables in the dataset? Bonus points for graphs - and thanks!

ChatGPT Code

```
/* Point to your dataset */
%let DS = HHS.HHS_OTJ_Raw;

ods graphics on;
options nosource nonotes;
title; footnote;

/* 0) Quick inventory: names, types, labels, formats, order */
proc contents data=&DS out=_contents(keep=libname memname name type
length format label varnum) noprint; run;
proc sort data=_contents; by varnum; run;

title "Variable Inventory for &DS";
```

```

proc print data=_contents label noobs; run;

/* Build macro lists of numeric and character variables */
proc sql noprint;
  select name into :numvars separated by ' ' from _contents where type=1 order
  by varnum;
  select name into :charvars separated by ' ' from _contents where type=2 order
  by varnum;
  select count(*) into :_N from &DS;
quit;

/* 1) Numeric summary: N, Missing, Mean, Std, Percentiles */
%if %length(&numvars) %then %do;
  title "Numeric Summary for &DS";
  proc means data=&DS n nmiss mean std min p25 median p75 max maxdec=3;
    var &numvars;
    ods output summary=_num_summary;
  run;

  title "Distributions for Numeric Variables in &DS";
  proc univariate data=&DS;
    var &numvars;
    histogram &numvars / normal;
    inset n mean std median min max / position=ne;
  run;
%end;

/* 2) Character summary: Top-k levels per variable (counts, show missings) */
%macro charfreq(maxlevels=25);
  %if %length(&charvars) %then %do;
    %local i n var;
    %let n=%sysfunc(countw(&charvars,%str( )));
    %do i=1 %to &n;
      %let var=%scan(&charvars,&i);

      title "Top &maxlevels Values for &var (including missing)";
      proc freq data=&DS order=freq;
        tables &var / missing nocum nopercent maxlevels=&maxlevels;
        ods output OneWayFreqs=_freq_&i;
      run;
    %end;
  %end;

```

```

%mend;
%charfreq(maxlevels=25);

/* 3) Compact “data profiler” table: type, #missing, #non-missing, distinct count
*/
data _profiles; length variable $32 type $5; stop; run;

%macro add_profiles(list, vtype);
%local i k var miss distinct;
%let k=%sysfunc(countw(&list,%str( )));
%do i=1 %to &k;
%let var=%scan(&list,&i);
proc sql noprint;
/* COUNT(DISTINCT) works for both char and numeric */
select sum(missing(&var)), count(distinct &var) into :miss, :distinct from &DS;
quit;

data _profiles; set _profiles end=_e;
length variable $32 type $5;
variable=&var; type=&vtype";
n=&_N; n_missing=&miss; n_nonmissing=n - n_missing;
distinct=&distinct; pct_missing=100*n_missing/n;
run;
%end;
%mend;

%if %length(&numvars) %then %add_profiles(&numvars,NUM);
%if %length(&charvars) %then %add_profiles(&charvars,CHAR);

proc sort data=_profiles; by type descending pct_missing variable; run;

title "Data Profiler: Missingness & Cardinality for &DS";
proc print data=_profiles label noobs;
var variable type n n_missing pct_missing n_nonmissing distinct;
label variable="Variable" type="Type" n="Rows"
      n_missing="Missing" pct_missing="% Missing"
      n_nonmissing="Non-Missing" distinct="Distinct Values";
format pct_missing 8.2;
run;

/* 4) Optional quick visuals:
   - Missingness bar by variable

```

- Cardinality (distinct count) by variable
Tip: Comment out if too many variables. */

```
%if %length(&numvars) or %length(&charvars) %then %do;  
title "Percent Missing by Variable";  
proc sgplot data=_profiles;  
    vbar variable / response=pct_missing datalabel;  
    xaxis discreteorder=data fitpolicy=thin;  
    yaxis label="% Missing";  
run;  
  
title "Distinct Values by Variable";  
proc sgplot data=_profiles;  
    vbar variable / response=distinct datalabel;  
    xaxis discreteorder=data fitpolicy=thin;  
    yaxis label="Distinct Count";  
run;  
%end;  
  
title; footnote; options source notes;
```

ChatGPT Notes

What you get

- Inventory table of all variables (name/type/label/format).
- Numeric summary (N, missing, mean, SD, min/percentiles/max).
- Character summaries with top 25 values per variable (includes missing).
- A compact data profiler table: rows, missing %, and distinct counts for every variable.
- Graphs: percent missing and cardinality per variable, plus histograms for numerics.

If you'd like, I can add:

- An ODS Excel export (one sheet per section),
- A CAS version using proc casutil and simple.summary,
- Or a toggle to limit histograms to, say, the top 12 numeric variables by variance.

- Yikes. That got real quick. And I'm getting the following error:

ERROR: Expected %DO not found.

ERROR: Skipping to next %END statement.

- Let's refine the prompt to simplify.

Revised ChatGPT Prompt

I'm a beginning SAS user and that code got much too complex. And it's giving me the following errors:

ERROR: Expected %DO not found.

ERROR: Skipping to next %END statement.
Can you please simplify?

Revised ChatGPT Code

```
/* Point to your table */
%let DS = HHS.HHS_OTJ_Raw;

/* 1) Quick inventory of variables (names, types, labels)
*/
proc contents data=&DS
    out=work._contents(keep=name type length label format
varnum)
    noprint;
run;

proc sort data=work._contents; by varnum; run;

title "Variable Inventory: &DS";
proc print data=work._contents label noobs; run;

/* 2) Numeric variables: summary stats + histograms */
title "Numeric Summary: &DS";
proc means data=&DS n nmiss mean std min p25 median p75 max
maxdec=3;
    var _numeric_;
run;

title "Numeric Distributions (Histograms): &DS";
proc univariate data=&DS noprint;
    var _numeric_;
    histogram _numeric_ / normal;
    inset n mean std median min max / position=ne;
run;

/* 3) Character variables: levels + one-way frequencies
   Tip: Change MAXLEVELS=25 if you want more/fewer
   categories shown */
title "Character Variables: Number of Distinct Levels";
proc freq data=&DS nlevels;
    tables _character_ / noprint;
run;

title "Top Values for Character Variables (includes
missing)";
proc freq data=&DS order=freq;
    tables _character_ / missing nocum nopercent /* limit
long lists: */ maxlevels=25;
run;
```

```
title;
```

Revised ChatGPT Details

What this does

- PROC CONTENTS → clean inventory of all variables.
- PROC MEANS → quick numeric summary (N, missing, mean, percentiles, etc.).
- PROC UNIVARIATE → histograms for all numeric variables (simple visuals).
- PROC FREQ (NLEVELS) → how many distinct values each character variable has.
- PROC FREQ (TABLES CHARACTER) → top categories for each character variable (caps lists at 25 to keep output readable).

If you want the output saved (e.g., to Excel) or want to cap the number of histograms printed, say the word and I'll add a tiny switch—still beginner-friendly.

- Replace your old “Explore Data Set” code with the simplified stuff. Like so:

```
*-----*
|           Explore Data Set           |
*-----*;
/* Point to your table */
%let DS = HHS.HHS_OT3_Raw;

/* 1) Quick inventory of variables (names, types, labels) */
proc contents data=&DS
  out=work._contents(keep=name type length label format varnum)
  noplay;
run;

proc sort data=work._contents; by varnum; run;

title "Variable Inventory: &DS";
proc print data=work._contents label noobs; run;

/* 2) Numeric variables: summary stats + histograms */
title "Numeric Summary: &DS";
proc means data=&DS n nmiss mean std min p25 median p75 max maxdec=3;
  var _numeric_;
run;

title "Numeric Distributions (Histograms): &DS";
proc univariate data=&DS noplay;
  var _numeric_;
  histogram _numeric_ / normal;
  inset n mean std median min max / position=ne;
run;

/* 3) Character variables: levels + one-way frequencies
   Tip: Change MAXLEVELS=25 if you want more/fewer categories shown */
title "Character Variables: Number of Distinct Levels";
proc freq data=&DS nlevels;
  tables _character_ / noplay;
run;

title "Top Values for Character Variables (includes missing)";
proc freq data=&DS order=freq;
  tables _character_ / missing nocum nopercent /* limit long lists: */ maxlevels=25;
run;

title;
```

- And then **Run!** When it's finished, check for errors and then examine the **Results**:

Log Results Output Data (1)

Variable Inventory: HHS.HHS_OTJ_Raw

Variable Name	Variable Type	Variable Length	Variable Number	Variable Label	Variable Format
State_FIP	1	8	1		BEST
State_Name	2	20	2		\$
YearQuarter	1	8	3		YYQ
Race_Ethnic	2	21	4		\$
EDUC_LTD	2	19	5		\$
Child_Status	2	14	6		\$
Unemp	1	8	7		BEST
in_LF	1	8	8		BEST
WTFINL	1	8	9		BEST

Numeric Summary: HHS.HHS_OTJ_Raw

The MEANS Procedure

Variable	N	Miss	Mean	Std Dev	Minimum	25th Pctl	Median	75th Pctl	Maximum
State_FIP	157785	0	28.471	15.680	1.000	15.000	29.000	41.000	56.000
YearQuarter	157785	0	21648.682	946.299	20089.000	20820.000	21640.000	22462.000	23284.000
Unemp	157785	0	0.177	0.381	0.000	0.000	0.000	0.000	1.000
in_LF	157785	0	0.619	0.486	0.000	0.000	1.000	1.000	1.000
WTFINL	157785	0	43753.888	103258.748	123.303	4122.790	11952.131	37107.798	2366665.817

- Do you now understand your data a bit better? Anything you'd change or update from the code provided?
 - Note that the data are disaggregated by demographic groups, so a lot of the values likely don't make much sense without properly aggregated population weights. Don't fret, we'll get there soon.
 - Caveats aside, this isn't a bad starting point.
- Please refine the code if you feel so inclined.
 - For example: for the first PROC MEANS step, change the MAXDEC=2 option to simplify the first table
 - This is also a sneaky way for me to show you the dynamic program editor (aka autocomplete) in SAS Studio

```
/* 2) Numeric variables: summary stats + histograms */
title "Numeric Summary: &DS";
proc means data=&DS n nmiss mean std min p25 median p75 max maxdec;
  var _numeric_;
run;

title "Numeric Distributions (Histograms): &DS";
proc univariate data=&DS noprint;
  var _numeric_;
  histogram _numeric_ / normal;
  inset n mean std median min max / position=ne;
run;

/* 3) Character variables: levels + one-way frequencies
   Tip: Change MAXDEC=25 if you want more/fewer digits */

```

File = SAS Programming +

PROCEDURE OPTIONS

MAXDEC=

Keyword: MAXDEC=

Context: [PROC MEANS] MAXDEC= option

[Syntax: MAXDEC=number]

Specifies the maximum number of decimal places to display the statistic or displayed output.

Search: [Product Documentation](#) [Samples and SAS Notes](#) [Papers](#)

Next Step: Collapse the data using SQL. Then ask ChatGPT what the heck happened.

- To expedite my storytelling, pretend you inherited some SQL code to help with the next step: which is to aggregate the data from the demographic cell level to statewide averages over time. And if it helps, pretend you inherited this code from me (actually, you did).
 - This code has a LOT in it – so simply pull it from the *HHS + OTJ - State Level Estimates using SQL.sas* file in [the GitHub repo](#) for this project. You can download the file in the GitHub repo [here](#):

- Paste that code into your SAS Program. But before you simply hit **Run**, let's unpack it a bit... and highlight another nice feature of ChatGPT. ChatGPT can easily explain complex code to us... particularly complex code that we inherit from others. So, let's have ChatGPT do exactly that.

ChatGPT Prompt

I'm struggling to understand some complex proc SQL code in SAS. Allegedly, this code takes a very long data set and collapses the data to a much small number of observations, while creating a bunch of new variables. Can you help explain it to me like I'm a 10 year old? Here is the code: [paste the SQL code]

ChatGPT Response

Perfect — let's unpack that code slowly and clearly, as if you're 10 years old 🤓

✳️ What This Code Is Doing in Plain English

Imagine you have a **giant spreadsheet** where every row represents **one person** — her job status, education, and whether she has kids.

The code's goal is to **shrink** that big spreadsheet down so that instead of millions of people, you just have **one row per state per year** — with a bunch of **summary statistics** (like unemployment rates and labor force participation rates).

⚙️ Step-by-Step Breakdown

1 `proc sql;`

This starts an SQL step inside SAS.

SQL is a way to ask questions about data — like "Show me the average income by state."

2 `create table hhs.covid_labor_supply as`

This makes a **new dataset** (a smaller table) called `HHS.COVID_LABOR_SUPPLY`.

Everything inside the parentheses that follows defines what goes *into* that new table.

3 `select distinct state_fip, state_name, year(yearquarter) as Year`

This means:

👉 "For every unique combination of state and year, show me one row."

`year(yearquarter)` pulls out just the **year** from something like "2020Q1".

4 All those `sum(...)/sum(...)` lines

This is where the *magic math* happens.

Each line computes a **weighted percentage** — things like unemployment or labor force participation rates — for different groups of women.

Let's take one as an example:

sas

 Copy code

```
sum( (unemp=1) * WTFINL ) / sum( (in_LF=1) * WTFINL ) as UE_Women
```

Translation:

- `(unemp=1)` turns into 1 if the person is unemployed, 0 otherwise.
- `WTFINL` is a **weight** that tells SAS how much each person "counts" (to make the data nationally representative).
- `sum((unemp=1)*WTFINL)` adds up the weights of all unemployed women.
- `sum((in_LF=1)*WTFINL)` adds up the weights of all women in the labor force.
- Dividing one by the other gives you the **unemployment rate**.



5 The same logic repeats

There are groups for:

- **Education** (`educ_ltd="High School Diploma"`, `"Some College"`, `"College +"`)
- **Child status** (`"No Children"`, `"Older Children"`, `"Child < 5"`)

Each of those blocks calculates:

- **Unemployment rate** (`UE_...`)
- **Labor force participation rate** (`LFP_...`)

6 `from hhs.hhs_otj_raw`

That's the **source dataset** — the giant spreadsheet of person-level data.

7 group by 1,2,3

This means:

"Group the data by the first three columns in the SELECT list"
(`state_fip`, `state_name`, and `year`).

So, for every `state` and `year`, SAS calculates all those summary stats.

8 order by 1,2,3

This sorts the results so the output appears neatly by state and year.

8 order by 1,2,3

This sorts the results so the output appears neatly by state and year.

▀▀▀ The End Result

Your new dataset (`hhs.covid_labor_supply`) will look something like this:

state_fip	state_name	year	UE_Women	LFP_Women	UE_Women_HS	UE_Women_SCollege
1	Alabama	2019	0.045	0.59	0.07	0.05
1	Alabama	2020	0.09	0.56	0.11	0.08
2	Alaska	2019

Each number is a **weighted percentage** summarizing many thousands of people.

▀▀▀ Simple Summary

You started with:

"One row per person."

You ended with:

"One row per state per year, with new columns that summarize how women are doing in the labor market."

Would you like me to show you a **visual sketch** (like a before/after diagram) to go with this explanation? It's great for teaching or including in slides.

- Gotta admit, that was a pretty cool summary and very useful. And now we understand our complex SQL code a bit better. Go ahead and **Run** that SQL step in your program, and ensure that it executes without error:

The screenshot shows a SAS log window with the following details:

- Log** tab selected.
- Output Data (1)** tab selected.
- Errors (0)**, **Warnings (0)**, and **Notes (10)** buttons.
- There are no messages.**
- SQL Code:**

```

115  ****LFP ****
116  sum( ( child_status="No Children" ) * ( in_LF=1 ) * WTFINL ) / sum( ( child_stat
117! LFP_Women_NoKidslabel="No Children" format percent9.1 ,
118  sum( ( child_status="Older Children" ) * ( in_LF=1 ) * WTFINL ) / sum( ( child_s
118! LFP_Women_OlderKidslabel="Older Children" format percent9.1 ,
119  sum( ( child_status="Child < 5" ) * ( in_LF=1 ) * WTFINL ) / sum( ( child_status
119! LFP_Women_YoungKidslabel="Young Children"format percent9.1
120
121
122  from hhs.hhs_otj_raw
123  groupby 1,2,3
124  orderby 1,2,3 ;
NOTE: Table HHS.COVID LABOR_SUPPLY created, with 459 rows and 17 columns.
125  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.14 seconds
      cpu time           0.19 seconds

SYMBOLGEN: Macro variable GRAPHTERM resolves to options noaccessible;
126
127
④ 128 /* region: Generated postamble */
129

```
- Execution Notes:**
 - NOTE: Table HHS.COVID LABOR_SUPPLY created, with 459 rows and 17 columns.
 - NOTE: PROCEDURE SQL used (Total process time):
 - real time 0.14 seconds
 - cpu time 0.19 seconds
- Symbolgen:** Macro variable GRAPHTERM resolves to options noaccessible;
- Postamble:** /* region: Generated postamble */

Summarize our newly created data set

- “Just” one item in this section. Let’s summarize our newly created data set!

ChatGPT Prompt

Can you please create some SAS code that creates a nice summary table of the new variables in the hhs.covid_labor_supply data set?

ChatGPT Code

```

/* Step 1: Quick overview of variables */
title "Variable Inventory for HHS.COVID LABOR_SUPPLY";
proc contents data=HHS.COVID LABOR_SUPPLY varnum;
run;

```

```

/* Step 2: Summary statistics for all numeric variables */
title "Summary Statistics for Key Labor Supply Variables";
proc means data=HHS.COVID_LABOR_SUPPLY n nmiss mean std min
max maxdec=2;
  var _numeric_;
run;

/* Step 3: (Optional) Average rates by year, nicely
formatted */
title "Average Labor Supply Measures by Year";
proc tabulate data=HHS.COVID_LABOR_SUPPLY
format=percent8.1;
  class year;
  var UE_Women LFP_Women
    UE_Women_HS UE_Women_SCollege UE_Women_CollegeP
    LFP_Women_HS LFP_Women_SCollege LFP_Women_CollegeP
    UE_Women_NoKids UE_Women_OlderKids UE_Women_YoungKids
    LFP_Women_NoKids LFP_Women_OlderKids
    LFP_Women_YoungKids;
  table year,
    (UE_Women LFP_Women
    UE_Women_HS UE_Women_SCollege UE_Women_CollegeP
    LFP_Women_HS LFP_Women_SCollege LFP_Women_CollegeP
    UE_Women_NoKids UE_Women_OlderKids
    UE_Women_YoungKids
    LFP_Women_NoKids LFP_Women_OlderKids
    LFP_Women_YoungKids)*mean=' ';
run;

/* Step 4: Bonus – quick visualization of unemployment
trends by year */
title "Average Unemployment Rate by Year (All Women)";
proc sgplot data=HHS.COVID_LABOR_SUPPLY;
  vbar year / response=UE_Women stat=mean datalabel;
  yaxis label="Unemployment Rate" grid;
run;

title;

```

ChatGPT Notes

Here's an easy, clean SAS script that:

1. Lists your variables and their types,
 2. Shows summary stats for all the numeric columns, and
 3. Creates a formatted "nice table" of averages by year.
-

💡 What This Code Does

- `PROC CONTENTS` → Lists all your new variables, their labels, and types.
 - `PROC MEANS` → Gives descriptive statistics for everything numeric.
 - `PROC TABULATE` → Builds a clean, *publication-style* summary table by year.
 - `PROC SGLOT` → Adds a quick, easy bar chart of the unemployment trend.
-

Nearing the end: let's plot state-level trends over time!

- For our next-to-last trick, let's get ChatGPT to help us plot the data over time. Because who doesn't love a good time trend!
- In particular, let's simplify the analysis and just look at the unemployment rates for 4 categories – *All women*, *women without children*, *women with children < 5*, and *women with children >=5* – by state and over time. These would be interesting trends for public policy experts to explore.

ChatGPT Prompt

Alright, one more trick. Can you please help me write some SAS code that will use the `hhs.covid_labor_supply` data set and examine `UE_Women`, `UE_Women_NoKids`, `UE_Women_OlderKids`, and `UE_Women_YoungKids` by state over time? The time variable is `year` and the state variable is named `state_name`. Thank you!

ChatGPT Code

```
title "Unemployment Rates for Women by State and Year";
proc sort data=HHS.COVID_LABOR_SUPPLY
out=work.covid_sorted;
by state_name year;
run;

proc print data=work.covid_sorted noobs label;
var state_name year UE_Women UE_Women_NoKids
UE_Women_OlderKids UE_Women_YoungKids;
format UE_Women--UE_Women_YoungKids percent8.1;
```

```

run;

ods graphics on;
title "Unemployment Trends for Women by State (Overall and
by Child Status)";
proc sgplot data=HHS.COVID_LABOR_SUPPLY;
    series x=year y=UE_Women / group=state_name
lineattrs=(pattern=solid) legendlabel="All Women";
    series x=year y=UE_Women_NoKids / group=state_name
lineattrs=(pattern=shortdash) legendlabel="No Children";
    series x=year y=UE_Women_OlderKids / group=state_name
lineattrs=(pattern=dot) legendlabel="Older Children";
    series x=year y=UE_Women_YoungKids / group=state_name
lineattrs=(pattern=dashdot) legendlabel="Young Children";
    yaxis label="Unemployment Rate" grid;
    xaxis label="Year";
    keylegend / title="Category" position=bottom across=2;
run;
ods graphics off;

proc sgplot data=HHS.COVID_LABOR_SUPPLY;
    where state_name in ("California", "Texas", "New York");
    series x=year y=UE_Women / group=state_name
lineattrs=(thickness=2);
    yaxis label="Unemployment Rate" grid;
run;

title "Unemployment Rate (All Women) by State and Year";
proc sgplot data=HHS.COVID_LABOR_SUPPLY;
    heatmap x=year y=state_name / colorresponse=UE_Women
colormodel=(white orange red);
    gradlegend label="Unemployment Rate";
    xaxis discreteorder=data;
    yaxis discreteorder=data;
run;

```

ChatGPT Additional Details

You got it — this is a great next step 🎉

You've already got a tidy dataset (`HHS.COVID_LABOR_SUPPLY`) with one row per **state-year**, so you can easily visualize and summarize unemployment trends by state and time.

Here's a clean SAS workflow with **summary tables and graphs**, perfect for both exploration and presentation:

⌚ Summary

Step	Purpose
PROC PRINT	Simple table by state and year
PROC SGPlot (series)	Trend lines over time
WHERE clause	Focus on selected states
HEATMAP	Visual summary across all states

- Copy and paste that code into your program:

```
*-----*
|                               Plots over time by State                         |
*-----*;*-----*
```

```
title "Unemployment Rates for Women by State and Year";
proc sort data=HHS.COVID_LABOR_SUPPLY out=work.covid_sorted;
  by state_name year;
run;

proc print data=work.covid_sorted noobs label;
  var state_name year UE_Women UE_Women_NoKids UE_Women_OlderKids UE_Women_YoungKids;
  format UE_Women--UE_Women_YoungKids percent8.1;
run;

ods graphics on;
title "Unemployment Trends for Women by State (Overall and by Child Status)";
proc sgplot data=HHS.COVID_LABOR_SUPPLY;
  series x=year y=UE_Women / group=state_name lineattrs=(pattern=solid) legendlabel="All Women";
  series x=year y=UE_Women_NoKids / group=state_name lineattrs=(pattern=shortdash) legendlabel="No Children";
  series x=year y=UE_Women_OlderKids / group=state_name lineattrs=(pattern=dot) legendlabel="Older Children";
  series x=year y=UE_Women_YoungKids / group=state_name lineattrs=(pattern=dashdot) legendlabel="Young Children";
  yaxis label="Unemployment Rate" grid;
  xaxis label="Year";
  keylegend / title="Category" position=bottom across=2;
run;
ods graphics off;

proc sgplot data=HHS.COVID_LABOR_SUPPLY;
  where state_name in ("California", "Texas", "New York");
  series x=year y=UE_Women / group=state_name lineattrs=(thickness=2);
  yaxis label="Unemployment Rate" grid;
run;

title "Unemployment Rate (All Women) by State and Year";
proc sgplot data=HHS.COVID_LABOR_SUPPLY;
  heatmap x=year y=state_name / colorresponse=UE_Women colormodel=(white orange red);
  gradlegend label="Unemployment Rate";
  xaxis discreteorder=data;
  yaxis discreteorder=data;
run;
```

- Now Run the code. Did you get any errors? I did!

(X) Errors (2) (A) Warnings (1) (i) Notes (25)

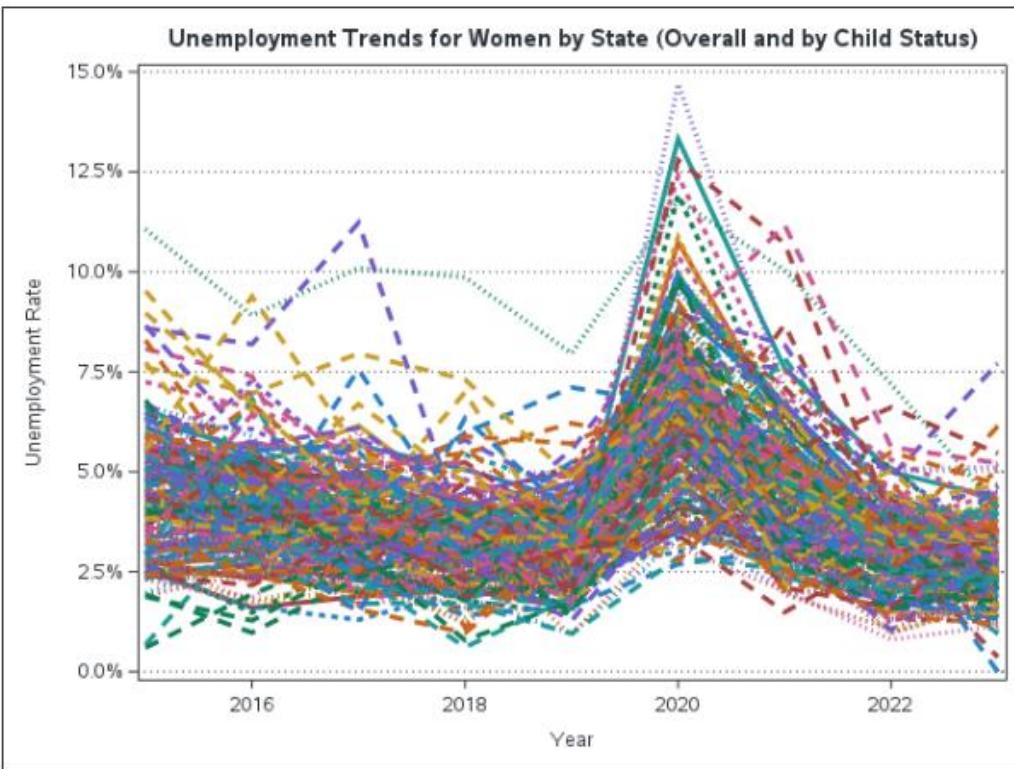
⚠ WARNING 1-322: Assuming the symbol DASH was misspelled as dashdot.
✖ ERROR 22-322: Syntax error, expecting one of the following: a quoted string, ;, /.
✖ ERROR 202-322: The option or parameter is not recognized and will be ignored.

```
108
109 proc sgplot data=HHS.COVID_LABOR_SUPPLY;
110   where state_name in ("California", "Texas", "New York");
111   series x=year y=UE_Women / group=state_name lineattrs=(thickness=2);
112   yaxis label="Unemployment Rate" grid;
113 run;
NOTE: PROCEDURE SGPLOT used (Total process time):
      real time            0.10 seconds
      cpu time             0.04 seconds

NOTE: Listing image output written to
      /opt/sas/viya/config/var/tmp/compsrv/default/ae8e02cf-79fb-418c-ba29-46360e6af0fd/SAS_work2C
      a6ef-cb79-4826-83e3-ca1ac7862e66-41963/SGPlot2.png.
NOTE: There were 27 observations read from the data set HHS.COVID_LABOR_SUPPLY.
      WHERE state_name in ('California', 'New York', 'Texas');
114
115
116 title "Unemployment Rate (All Women) by State and Year";
117 proc sgplot data=HHS.COVID_LABOR_SUPPLY;
118   heatmap x=year y=state_name / colorresponse=UE_Women colormodel=(white orange red);
119   gradlegend label="Unemployment Rate";
      -----
      22
      202
ERROR 22-322: Syntax error, expecting one of the following: a quoted string, ;, /.
ERROR 202-322: The option or parameter is not recognized and will be ignored.
120   xaxis discreteorder=data;
121   yaxis discreteorder=data;
122 run;
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE SGPLOT used (Total process time):
      real time            0.00 seconds
      cpu time             0.00 seconds

SYMBOLGEN:  Macro variable GRAPHTERM resolves to goptions noaccessible;
123
124
④ 125 /* region: Generated postamble */
136
```

- Moreover, what is this?



- Ummmm, yeah. What the heck just happened? Let's ask **ChatGPT** to refine the output, starting with the error first.

Revised ChatGPT Prompt

The heatmap syntax gives me this error:
 title "Unemployment Rate (All Women) by State and Year";
 proc sgplot data=HHS.COVID_LABOR_SUPPLY;
 heatmap x=year y=state_name / colorresponse=UE_Women colormodel=(white
 orange red);
 gradlegend label="Unemployment Rate";

22

202

ERROR 22-322: Syntax error, expecting one of the following: a quoted string, , , /.
 ERROR 202-322: The option or parameter is not recognized and will be ignored.

Revised ChatGPT Response

Excellent catch 🎉 — you just bumped into a tiny SAS syntax quirk!

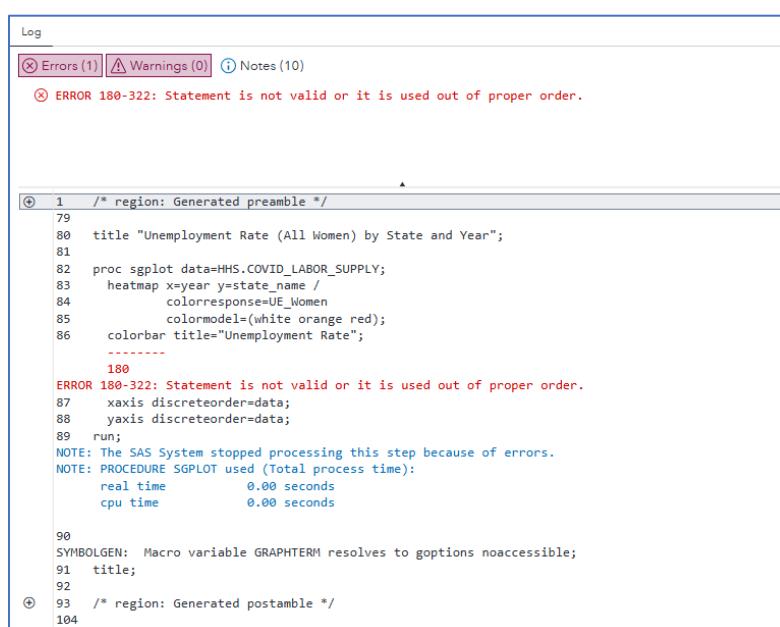
That error happens because in **PROC SGPlot**, the correct statement to control the color legend is `keylegend`, not `gradlegend`. The `gradlegend` keyword is used in **PROC SGMAP**, not **SGPLOT**.

Let's fix it and simplify a bit so it runs cleanly in **SAS Viya** or **SAS Studio** 👍

Revised ChatGPT Code

```
title "Unemployment Rate (All Women) by State and Year";  
  
proc sgplot data=HHS.COVID_LABOR_SUPPLY;  
  heatmap x=year y=state_name /  
    colorresponse=UE_Women  
    colormodel=(white orange red);  
  colorbar title="Unemployment Rate";  
  xaxis discreteorder=data;  
  yaxis discreteorder=data;  
run;  
  
title;
```

- Let's replace that code in your program. And rerun. How did it go? I got this error:

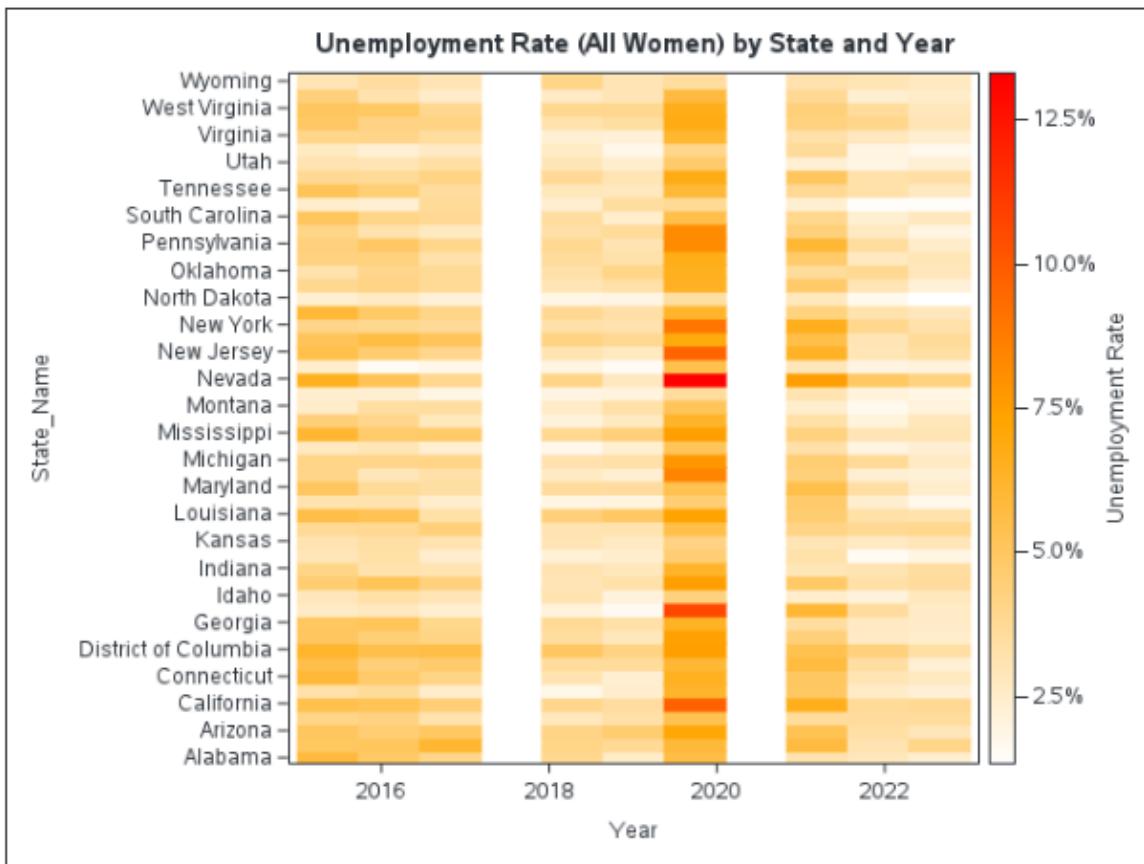


The screenshot shows the SAS Log window with the following details:

- Log**
- Errors (1)**: A red box highlights this tab.
- Warnings (0)**
- Notes (10)**
- ERROR 180-322: Statement is not valid or it is used out of proper order.**
- Code Snippet (Lines 1-94):**

```
1  /* region: Generated preamble */  
79  
80  title "Unemployment Rate (All Women) by State and Year";  
81  
82  proc sgplot data=HHS.COVID_LABOR_SUPPLY;  
83  heatmap x=year y=state_name /  
84    colorresponse=UE_Women  
85    colormodel=(white orange red);  
86  colorbar title="Unemployment Rate";  
87  -----  
88  180  
89  ERROR 180-322: Statement is not valid or it is used out of proper order.  
90  xaxis discreteorder=data;  
91  yaxis discreteorder=data;  
92  run;  
NOTE: The SAS System stopped processing this step because of errors.  
NOTE: PROCEDURE SGPlot used (Total process time):  
      real time          0.00 seconds  
      cpu time          0.00 seconds  
93  SYMBOLGEN: Macro variable GRAPHTERM resolves to goptions noaccessible;  
94  title;  
95  /* region: Generated postamble */  
104
```
- NOTE: The SAS System stopped processing this step because of errors.**
- NOTE: PROCEDURE SGPlot used (Total process time):**
 - real time 0.00 seconds
 - cpu time 0.00 seconds
- SYMBOLGEN: Macro variable GRAPHTERM resolves to goptions noaccessible;**
- 93 /* region: Generated postamble */**

- 0-2. But I can keep sharing the errors with ChatGPT and having it refine the syntax. For now, I'll just comment out the colorbar line so that the code will work. Which produces:



- Well, that's still fun! Let's move on and address the other issue in the code – that overwhelming “Unemployment Trends for Women by State (Overall and by Child Status)” plot. It's just too much.

Revised (2) ChatGPT Prompt

Alright, one final trick. The “Unemployment Trends for Women by State (Overall and by Child Status)” plot is overwhelming. Can you create a series of plots, by state, which show the four time series by child status? Thanks!

Revised (2) ChatGPT Code

```
/* Small multiples: one panel per state, four lines per panel */
ods graphics on;
```

```
proc sgpanel data=HHS.COVID_LABOR_SUPPLY;
  panelby state_name / columns=5 novarname uniscale=column onepanel;
  series x=year y=UE_Women      / legendlabel="All Women";
  series x=year y=UE_Women_NoKids / legendlabel="No Children"
  lineattrs=(pattern=shortdash);
```

```

    series x=year y=UE_Women_OlderKids / legendlabel="Older Children"
    lineatrs=(pattern=dot);
    series x=year y=UE_Women_YoungKids / legendlabel="Young Children"
    lineatrs=(pattern=dashdot);
    colaxis label="Year" integer;
    rowaxis label="Unemployment Rate" valuesformat=percent8.1 grid;
    keylegend / position=bottom across=2;
    title "Unemployment Trends by Child Status — One Panel per State";
    run;

ods graphics off;

```

Revised (2) ChatGPT Details

Absolutely—great call to split this by state. Here are two clean options:

Option A — Small multiples (all states in a grid)

Keeps everything in one view using `PROC SGPANEL`. Tweak `columns=` to fit your screen.

Option B — One state per page (easy to read / export to PDF)

Creates a separate plot for each state using `BY state_name`. Great for sharing.

- Two options → nice! Let's try option 1. Paste it into your program, like so:

```

*-----*-----*-----*
|                               Plots over time by State                         |
*-----*-----*-----*;

title "Unemployment Rates for Women by State and Year";
proc sort data=HHS.COVID_LABOR_SUPPLY out=work.covid_sorted;
  by state_name year;
run;

proc print data=work.covid_sorted noobs label;
  var state_name year UE_Women UE_Women_NoKids UE_Women_OlderKids UE_Women_YoungKids;
  format UE_Women--UE_Women_YoungKids percent8.1;
run;

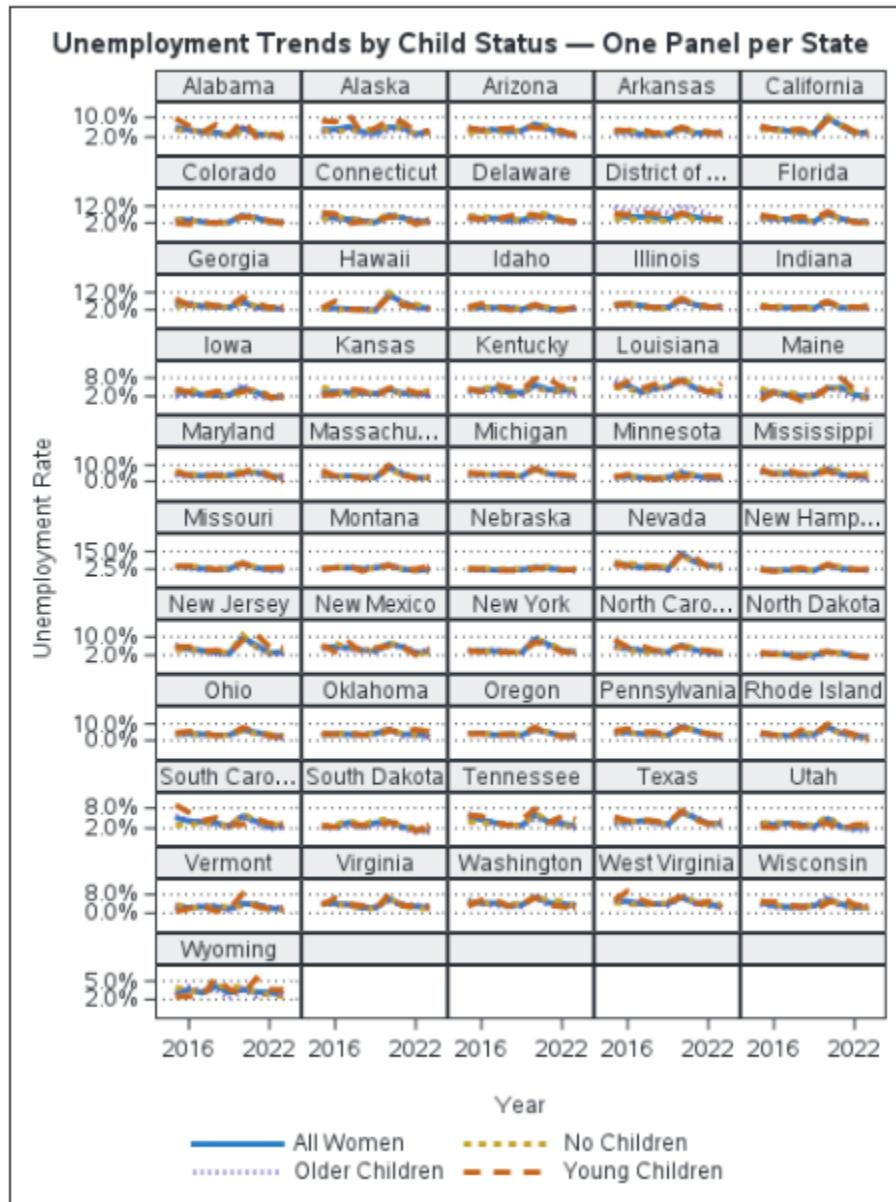
/* Small multiples: one panel per state, four lines per panel */
ods graphics on;

proc sgpanel data=HHS.COVID_LABOR_SUPPLY;
  panelby state_name / columns=5 novarnname uniscale=column onepanel;
  series x=year y=UE_Women      / legendlabel="All Women";
  series x=year y=UE_Women_NoKids / legendlabel="No Children"
  series x=year y=UE_Women_OlderKids / legendlabel="Older Children"
  series x=year y=UE_Women_YoungKids / legendlabel="Young Children"
  colaxis label="Year" integer;
  rowaxis label="Unemployment Rate" valuesformat=percent8.1 grid;
  keylegend / position=bottom across=2;
  title "Unemployment Trends by Child Status – One Panel per State";
run;

ods graphics off;

```

- Submit that code. And here's my output:



- That's still a lot... and we'll want to adjust more settings. But it's a solid start and better than the initial output. Moreover, there's also Option 2 – one state per page – if you want to go that route instead. The adventure is yours 😊

Final Trick: Documentation!

- The last thing I want to show you is that ChatGPT is great at helping you document your code. Because there is nothing worse than – say 6 months from now – looking at your code and thinking “what the heck is this?” We'll ask ChatGPT to help with the code

under the **Plot over time by State** section, but you could do it for the full program later, if you wanted.

ChatGPT Prompt

Alright, one more request. Can you add some details in SAS comments to this code, so I can better understand it 6 months from now?

[insert the code from your program]

ChatGPT Code

```
*****
*****  
Program: covid_labor_supply_summary.sas  
Purpose: Summarize and visualize unemployment rates for women by state  
and year from the HHS.COVID_LABOR_SUPPLY dataset  
Author: (You!)  
Updated: <add date here>  
Notes: This code prints summary tables, then creates multiple graphs:  
1. Table of unemployment measures by state & year  
2. Small-multiple line charts (one per state)  
3. Focused line chart for select states  
4. Heatmap of unemployment rates across states & years  
*****  
*****/  
  
/*-----*/  
/* STEP 1: Sort data by state and year */  
/*-----*/  
title "Unemployment Rates for Women by State and Year";  
  
proc sort data=HHS.COVID_LABOR_SUPPLY out=work.covid_sorted;  
by state_name year; /* Ensures table and plots display in order */  
run;  
  
/*-----*/  
/* STEP 2: Print simple summary table */  
/*-----*/  
proc print data=work.covid_sorted noobs label;  
var state_name year
```

```

UE_Women UE_Women_NoKids UE_Women_OlderKids
UE_Women_YoungKids;
  format UE_Women--UE_Women_YoungKids percent8.1; /* Show % with 1
decimal */
run;

/*-----*/
/* STEP 3: Line charts by state (small multiples)      */
/*      Each panel = one state; shows 4 lines per panel   */
/*-----*/
ods graphics on;

proc sgpanel data=HHS.COVID_LABOR_SUPPLY;
  /* Create one panel per state, 5 panels per row */
  panelby state_name / columns=5 novarname uniscale=column onepanel;

  /* Add 4 time series lines per panel (different line styles) */
  series x=year y=UE_Women      / legendlabel="All Women";
  series x=year y=UE_Women_NoKids / legendlabel="No Children"
lineattrs=(pattern=shortdash);
  series x=year y=UE_Women_OlderKids / legendlabel="Older Children"
lineattrs=(pattern=dot);
  series x=year y=UE_Women_YoungKids / legendlabel="Young Children"
lineattrs=(pattern=dashdot);

  /* Format axes */
  colaxis label="Year" integer;
  rowaxis label="Unemployment Rate" valuesformat=percent8.1 grid;

  /* Add legend below charts */
  keylegend / position=bottom across=2;

  title "Unemployment Trends by Child Status — One Panel per State";
run;

ods graphics off;

/*-----*/
/* STEP 4: Focused plot for selected states (less cluttered)  */
/*-----*/

```

```

proc sgplot data=HHS.COVID_LABOR_SUPPLY;
  where state_name in ("California", "Texas", "New York"); /* Filter states */
  series x=year y=UE_Women / group=state_name lineattrs=(thickness=2);
  yaxis label="Unemployment Rate" grid;
  title "Unemployment Trends — Selected States";
run;

/*
/* STEP 5: Heatmap of unemployment rate by state & year */
/* Darker color = higher unemployment */
title "Unemployment Rate (All Women) by State and Year";

proc sgplot data=HHS.COVID_LABOR_SUPPLY;
  heatmap x=year y=state_name /
    colorresponse=UE_Women
    colormodel=(white orange red); /* White=low, red=high unemployment */
    colorbar title="Unemployment Rate"; /* Adds color scale legend */
    xaxis discreteorder=data;
    yaxis discreteorder=data;
run;

title;

```

ChatGPT Notes

Quick Reference for Future You:

- `PROC SORT` → puts data in order for tables & plots
- `PROC PRINT` → gives a clean, tabular snapshot
- `PROC SGANEL` → creates *small multiples* (one chart per state)
- `PROC SGLOT (WHERE)` → zooms in on select states
- `HEATMAP` → shows state-year patterns in color

- Submit that code to ensure that everything still runs as expected. And then have a big exhale. You did it! And by “it” I mean that you’ve created an impressive program very quickly, all with the help of a few refined prompts and a great copilot like ChatGPT. Yay!

Recap

This SAS Guided Demo highlights how large language models like ChatGPT can serve as effective coding partners. Whether you're drafting new syntax, interpreting complex code, or documenting your work, LLMs can accelerate learning and improve clarity. By applying prompt-engineering best practices, programmers can collaborate with these tools to code smarter, learn faster, and communicate their work more effectively.

A few more key takeaways:

- ChatGPT will get you most of the way there!
- But, it doesn't always give you the most efficient code first...
 - ... So you'll (potentially) need to refine your prompts.
- And, it may give you code that doesn't work. So, you'll need to know how to debug... or ask better prompts.
- But... ChatGPT is great at getting you unstuck... and helps a lot with laborious tasks like documentation.

Main point – if you forget everything else: ChatGPT is a great teammate to get your program up and running. And then you can take it from there!