



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

SAS Guided Demo | SAS Academic Programs

### Purpose

This SAS Guided Demo seeks to get coders more comfortable using large language models (LLMs), such as ChatGPT, to help them generate code. We walk through two main cases where LLMs can be particularly useful for those learning new code: (1) in generating a first iteration of syntax, which gets you about 80% of where you need to be and (2) understanding complex code written by someone else. 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 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.....	3
General Notes (from a human) .....	3
Direct Advice from ChatGPT .....	3
Part 2: SAS Guided Demo.....	5
Learning Objectives (aka – what I’m trying to trick you into learning): .....	5
GitHub Links .....	5
Disclaimers .....	5
Start the adventure in SAS Studio – and setup the project .....	5
Shortcuts to assigning libraries .....	7
Read in a file from GitHub.....	12
Next Step: Better understand your data .....	17
Next Step: Collapse the data using SQL. Then ask ChatGPT what the heck happened. ....	26
One Final Item: Let’s plot state-level trends over time! .....	33

## Part 1: LLM best practices for better prompt engineering

### General 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.

### Direct Advice from ChatGPT

1. Be Clear and Specific
  - Detail Your Goals: Clearly state what you want to achieve with your code. For example, specify if you need a Python script to analyze a dataset or a SAS procedure to perform statistical tests.
  - Define the Scope: Limit the scope of your request to ensure clarity. For example, ask for a SAS procedure to calculate the mean and standard deviation, rather than a broad request for data analysis.
2. Provide Context
  - Explain the Background: Include any relevant background information that might influence how the code should be written, such as the data structure or specific libraries you intend to use.
  - State the Environment: Mention any specific IDEs, version numbers, or operating systems, if relevant.
3. Use Precise Terminology
  - Programming Language: Clearly state whether you are asking for Python, SAS, or another programming language.
  - Functions and Libraries: Specify any particular libraries or functions you want used, such as Pandas for Python or PROC SQL in SAS.

4. Include Examples
  - Data Samples: Provide small snippets of data or describe the data structure to help understand the context and requirements.
  - Expected Output: Show an example of the expected output, which can guide the development of the solution.
5. Break Down Complex Requests
  - Step-by-Step Instructions: If the task is complex, break it down into smaller, manageable tasks and request assistance step by step.
  - Multiple Prompts: Consider using multiple prompts to handle different parts of the problem.
6. Iterate and Refine
  - Feedback Loop: Use the initial responses to refine your questions. Be open to iterating on the solution.
  - Clarify Misunderstandings: If the response doesn't meet your expectations, clarify what was missing or misunderstood.
7. Maintain a Logical Flow
  - Sequential Order: When working on a project, ask questions in a logical sequence that matches the development flow of your code.
  - Follow-up Questions: If building on previous code, reference it and ask specific follow-up questions to expand or optimize the code.
8. Ethical and Privacy Considerations
  - Avoid Sensitive Data: Do not include any personal or sensitive data in your examples.
  - Generalize Problems: When possible, generalize the problem to avoid any specific business logic or proprietary information.
9. Review and Proofread
  - Check Your Prompt: Review your prompt to ensure it's clear and free of typos or ambiguous phrasing, which can lead to misunderstandings.
10. Leverage Examples and Documentation
  - Refer to Documentation: When possible, refer to or include links to documentation or resources that could help understand specific functions or methods.

## 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 isn’t on the verge of replacing experienced coders... anytime soon.

### GitHub Links

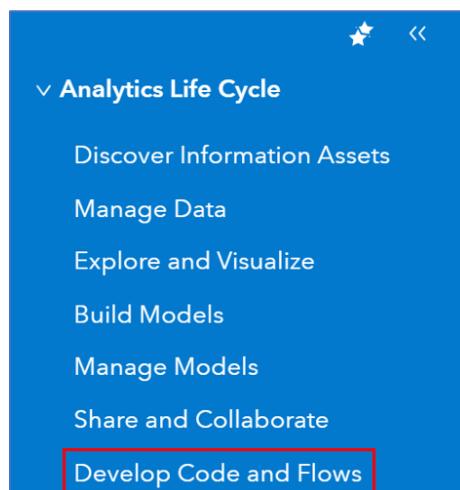
- SAS On-the-Job HHS Analyst Project: <https://github.com/lincolngroves/SAS-OTJ-HHS>
- Repo for this workshop: <https://github.com/SAS-Innovate-2025/Better-Prompt-Engineering-Leveraging-ChatGPT-to-get-started-with-SAS-coding>
  - This is where you can directly copy SAS Code used in the program we’ll create together.

### Disclaimers

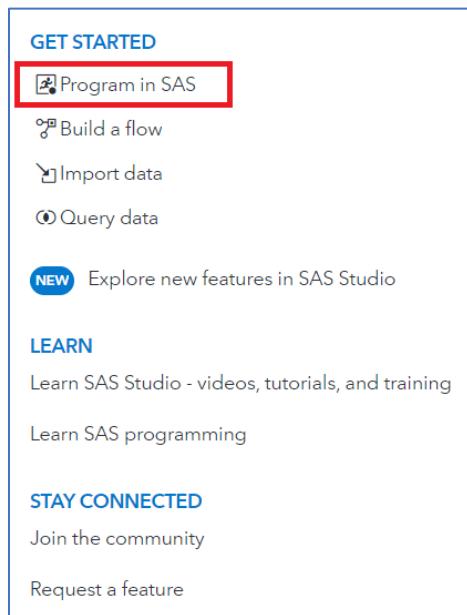
- This SAS Guided Demo was written in SAS Viya 4, version 2024.09 LTS.
- Moreover, we use ChatGPT-4o, as it’s now free to use for everyone without an account. Learn more <https://openai.com/chatgpt/>.

Start the adventure in SAS Studio – and setup the project

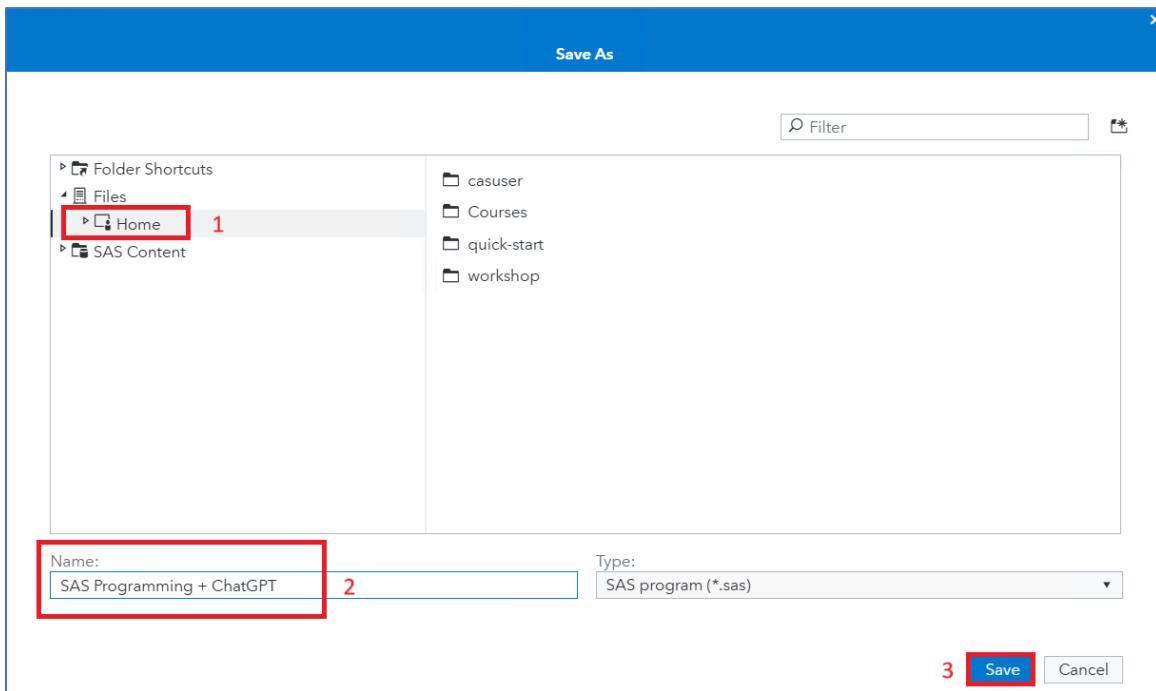
- From the **Applications** menu, select **Develop Code and Flows**



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



- Click to save that new program as **SAS Programming + ChatGPT** in your **Home** directory:

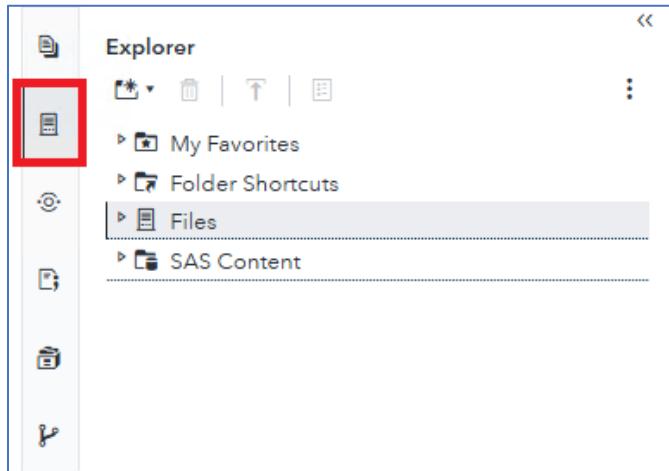


- Save – and save often!
- Any good program starts with a nice header and some titles to describe the project and objective. So, create something akin to the following – or simply copy from the example below (also in the GitHub folder):

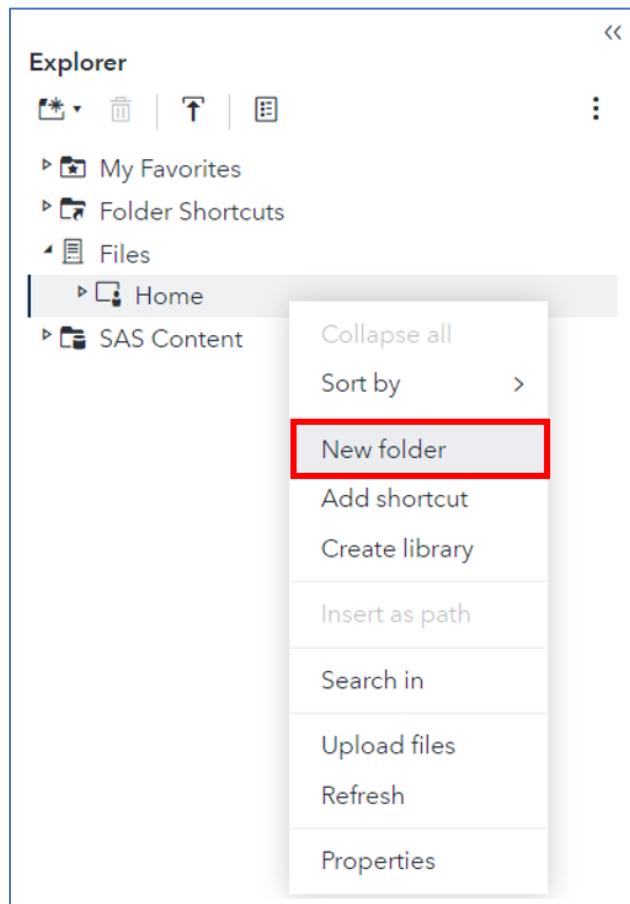
```
1 *----- SAS Innovate 2025
2 | Learning SAS Programming with Help from ChatGPT
3 |
4 *-----;
5 options orientation=landscape mlogic symbolgen pageno=1 error=3;
6 title1 "Global Academic Programs Workshop";
7 title2 "Learning SAS Programming with Help from ChatGPT";
8 footnote "File = SAS Programming + ChatGPT";
9
```

## Shortcuts to assigning libraries

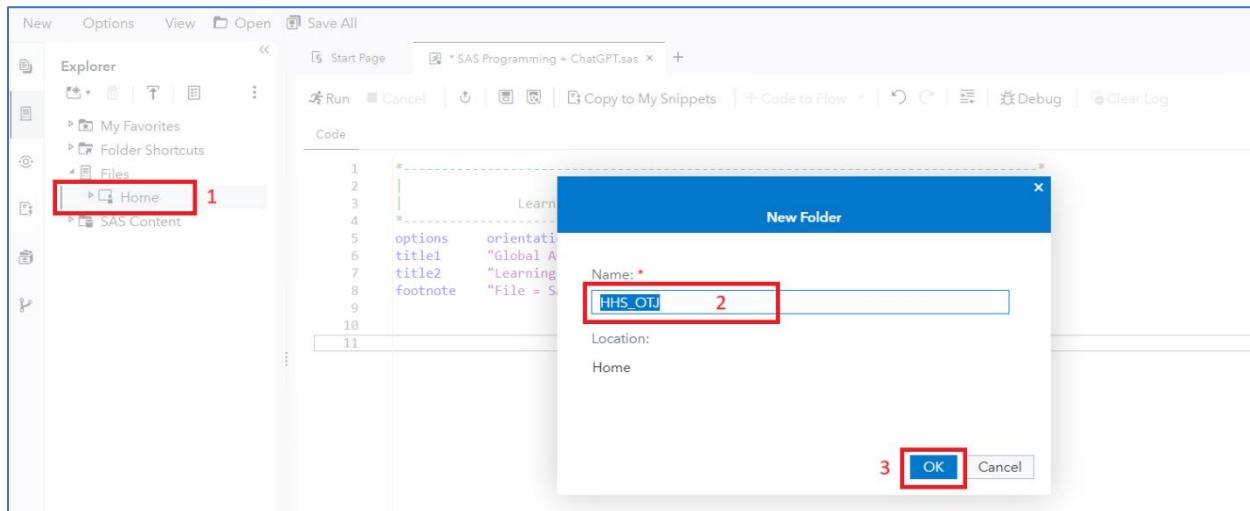
- The next step is to tell SAS where we'd like to save our data. Start by expanding **Explorer** on the left navigation pane:



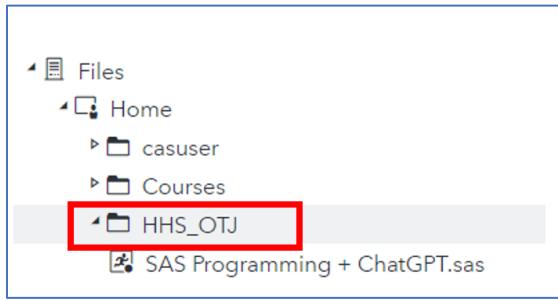
- For this project, let's create a new folder under **Files >> Home** called **HHS\_OTJ**. To do so, expand the **Files** folder, then find **Home**. By right-clicking on **Home**, you can select **New folder**, as follows:



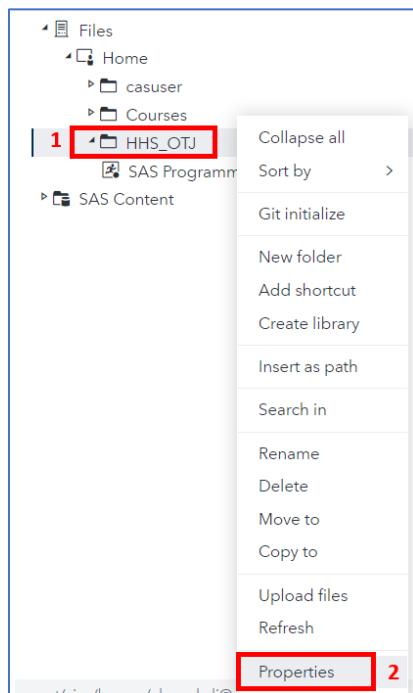
- Then create the **New Folder** titled **HHS\_OTJ**:



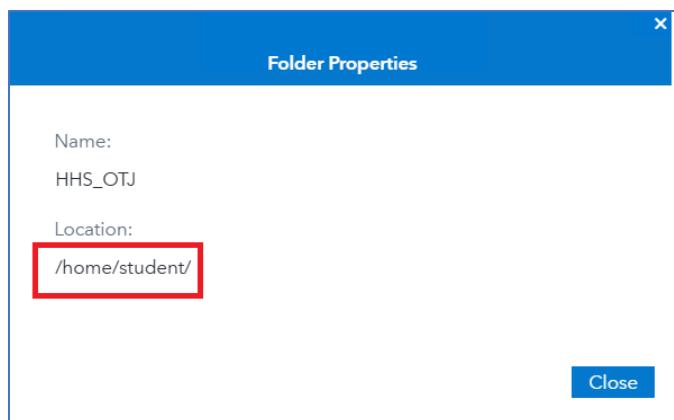
- Check under **Home** to see that a new folder was created. There it is!



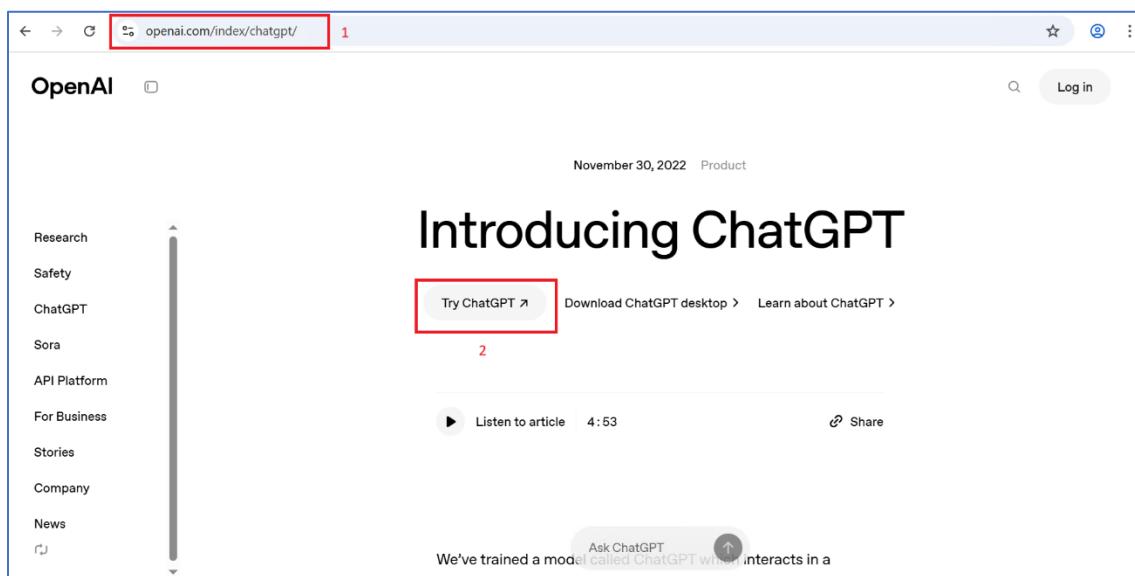
- Fun fact: we're all different users – and people. So, our paths on the VFL system will be different. Want proof? Well, right-click on **HHS\_OTJ**, then select **Properties**:



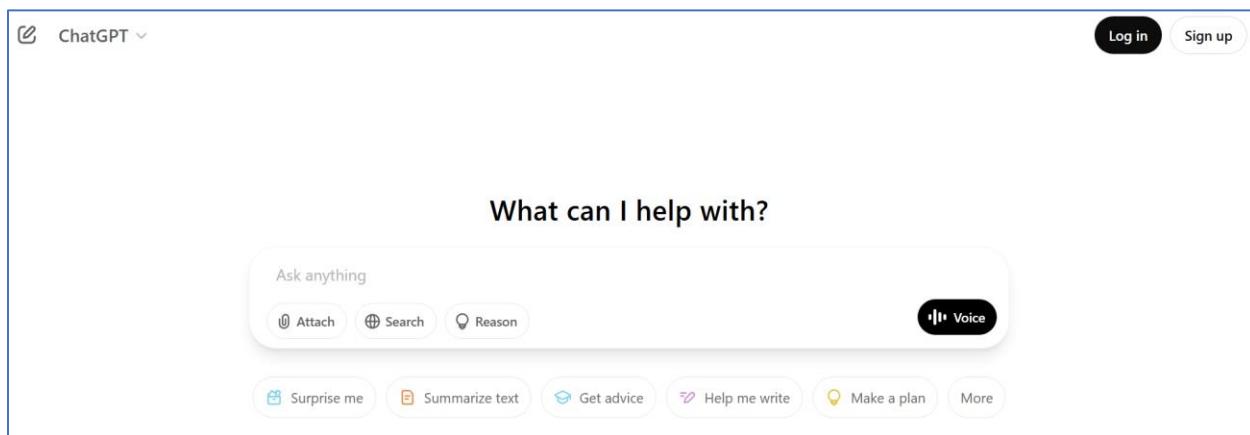
- Can you match this?



- Alright, alright. You can. But this is a special SAS Innovate 2025 fun fact, as the environments are replicated for consistency. When you go back to the office you'll have a different path from my environment. Because you're not me 😊 Anyway, a relative file reference could be useful here to make the program flexible across environments – 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.
- To start, open ChatGPT by going here: <https://openai.com/index/chatgpt/>. Then click **Try ChatGPT:**



- Aww. It's ready to help:



- Now let's get right to our first question:
  - Enter the following in the ChatGPT Prompt

I have a question about assigning SAS libraries. Instead of writing out the full path to a library, such as libname HHS "/home/student/HHS\_OTJ", how can I use a ~ to shorten the path? And what exactly does the ~ do?

- ChatGPT Code

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

- ChatGPT Explanation

In SAS, you can use the tilde (~) as shorthand to represent the home directory of the current user. This is a feature of many Unix-like systems, including Linux, and it can be used to make file paths more concise.

Instead of writing out the full path like:

```
sas
```

 Copy

```
libname HHS "/home/student/HHS_OTJ";
```

You can use:

```
sas
```

 Copy

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

- More info:

#### Explanation of ~:

- The tilde (~) refers to the home directory of the current user. So when you use ~, it tells the system to automatically substitute that with the full path of your home directory. For example, if the current user's home directory is /home/student, then ~/HHS\_OTJ will resolve to /home/student/HHS\_OTJ.

This can make your code more portable, as it avoids hard-coding specific paths to the home directory and can easily work across different environments.

- So, that's a sweet hack! Add that line of code to your program, 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 library where you can store data. Submit that program using the **Run** icon and make sure it all runs smoothly:

The screenshot shows the SAS Studio interface. In the top navigation bar, the 'Run' button is highlighted with a red box. The code editor displays the SAS program. The log window on the right shows the execution results, indicating that the libref HHS was successfully assigned to the path ~/HHS\_OTJ.

```

Log
Errors (0) Warnings (0) Notes (9)
There are no messages.

1 /* region: Generated preamble */
79 *
80 *
81 | SAS Innovate 2025
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: /home/student/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 bounced back quickly in the United States after the initial COVID shockwaves.
- 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/ACS\\_2015\\_2022\\_ltd.csv](https://github.com/lincolngroves/SAS-OTJ-HHS/blob/main/ACS_2015_2022_ltd.csv)
- As the .CSV file name suggests, the data spans from 2015 to 2022.
- Now let's see how ChatGPT can help us write some code to read in the data. The prompt:  
 Thanks! 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/ACS\\_2015\\_2022\\_ltd.csv](https://github.com/lincolngroves/SAS-OTJ-HHS/blob/main/ACS_2015_2022_ltd.csv). Can you please help me write this code?
- Response:

### Step 1: Get the raw file URL

First, get the direct raw URL of the .csv file. For the file you mentioned, the raw file URL is:

```
bash Copy
https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/ACS_2015_2022_ltd.csv
```

### Step 2: SAS code to download the CSV file

You can use the following SAS code to download the file and save it to your local machine.

- Code

```

filename mycsv url
"https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/ACS_2015_2022_ltd.csv";

proc http

url="https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/ACS_2015_2022_ltd.csv"
method="GET"
out=mycsv;
run;

* Now you can import the CSV file into SAS;
proc import datafile=mycsv
out=work.acs_data
dbms=csv
replace;
```

```
getnames=yes;  
run;  
  
* Print the data to verify the import;  
proc print data=work.acs_data;  
run;
```

- Additional ChatGPT details:

#### Explanation:

##### 1. FILENAME Statement:

The `filename` statement assigns a URL to a SAS filename reference (`mycsv`). The `url` option allows you to refer to the CSV file directly from the raw GitHub URL.

##### 2. PROC HTTP:

The `proc http` procedure downloads the file using the raw GitHub URL and saves it to the filename reference `mycsv`.

##### 3. PROC IMPORT:

This procedure imports the `.csv` file into a SAS dataset (`work.acs_data`). You can set the `getnames=yes` option to automatically use the first row of the CSV file as variable names.

##### 4. PROC PRINT:

Finally, `proc print` displays the imported data so you can verify the download.

Make sure you have an internet connection for this to work, as SAS will need to download the file from the GitHub repository.

Let me know if you need further assistance!

- Copy and paste that code into your SAS program. Run it so we can see if it works!

```

7 title2 Learning SAS Programming WITH Help FROM ChatGPT ;
8 footnote "File = SAS Programming + ChatGPT";
9
10 libname HHS "~/HHS_OTJ";
11
12
13
14
15 filename mycsv url "https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/ACS_2015_2022_ltd.csv";
16
17 proc http
18   url="https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/ACS_2015_2022_ltd.csv"
19   method="GET"
20   out=mycsv;
21 run;
22
23 * Now you can import the CSV file into SAS;
24 proc import datafile=myscv
25   out=work.acs_data
26   dbms=csv
27   replace;
28   getnames=yes;
29 run;
30
31 * Print the data to verify the import;
32 proc print data=work.acs_data;
33 run;

```

Log Results Output Data (1)

```

1 /* region: Generated preamble */
79
80
81 filename mycsv url "https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/ACS_2015_2022_ltd.csv";
82
83 proc http
84   url="https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/ACS_2015_2022_ltd.csv"
85   method="GET"
86   out=mycsv;
87 run;
ERROR: URL/HTTP Access Method supports INPUT mode only.
ERROR: Invalid open mode.
NOTE: 200 OK
ERROR: Generic HTTP Client Error
NOTE: PROCEDURE HTTP used (Total process time):
      real time          0.40 seconds
      cpu time           0.02 seconds
NOTE: The SAS System stopped processing this step because of errors.
88
89 * Now you can import the CSV file into SAS;
90 proc import datafile=myscv
91   out=work.acs_data
92   dbms=csv
93   replace;
94   getnames=yes;
95 run;
NOTE: Unable to open generated catalog. CACHED DABMS DABMC SELECT

```

- Well... does it work? Kind of. But I never like seeing an error to start the program. Additionally, we don't need to print all the data... because it'll just eat up resources unnecessarily. A refined request to ChatGPT is as follows:

That code looks like it worked – but the proc http step has an error. Can you write me some new code that doesn't use proc http? Additionally, can you just print the first 100 observations? Thanks!

- (Partial) Response:

Sure! If you want to download the file without using `PROC HTTP`, we can instead use SAS's `FILENAME` statement with the `URL` option, and the `PROC IMPORT` step can handle reading the CSV file directly from the URL without needing to download it explicitly.

Here's an updated version of the code that avoids `PROC HTTP`:

- Code

```

filename mycsv url "https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/ACS_2015_2022_ltd.csv";

* Import the CSV file directly from the GitHub raw URL;
proc import datafile=myscv
  out=work.acs_data
  dbms=csv
  replace;
  getnames=yes;

```

```

run;

* Print the data to verify the import;
proc print data=work.acs_data (obs=100);
run;

```

- Boom! Let's make a couple of further refinements because I have preferences.
  - Add a pretty section title, like "Import Data from GitHub"
  - For the proc import statement:
    - Change the output data file to **HHS.HHS\_OTJ\_Raw** – so that we can save the imported data to that newly created SAS library HHS.
    - Add the **guessingrows=100000** statement (this avoids truncation of the state names which – trust me – will happen without it).
- Original code – and those cumulative changes – should yield the following:

```

*-----*
|           Import Data from GitHub           |
*-----*;

filename mycsv url "https://raw.githubusercontent.com/lincolngroves/SAS-OTJ-HHS/main/ACS_2015_2022_ltd.csv";
* Import the CSV file directly from the GitHub raw URL;
proc import datafile=mycsv
  out=HHS.HHS_OTJ_Raw
  dbms=csv
  replace;
  getnames=yes;
  guessingrows=100000;
run;

* Print the data to verify the import;
proc print data=HHS.HHS_OTJ_Raw (obs=100);
run;

```

- Save your program! Then mash that running man icon to run the entire program from start to finish:



- 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:

The screenshot shows the SAS Studio interface. In the left sidebar (Explorer), there is a folder named 'HHS\_OTJ' containing a file 'hhs\_otj\_raw.sas7bdat'. In the center, a preview pane displays a table titled 'hhs\_otj\_raw' with 20 rows of data. The columns are: State\_FIP, State\_Name, YearQuarter, Race\_Ethnic, EDUC\_LTD, Child\_Status, and Un. The data shows various records for Alabama from 2015Q1, with details like Black, Non-Hispanic, < HS, and No Children.

	State_FIP	State_Name	YearQuarter	Race_Ethnic	EDUC_LTD	Child_Status	Un
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	

- A few key takeaways here:
  - ChatGPT will get you most of the way there.
  - But, it won't always give you the most efficient code first...
    - ... So you'll need to refine your prompts.
    - And, it may give you code that doesn't work.
  - Point: ChatGPT is not quite ready to replace me as a programmer... yet!

## Next Step: Better understand your data

- You likely still don't understand the underlying data. And that's ok because we're just getting started!
- 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 Response, Part 1

Certainly! To summarize both the numeric and non-numeric variables in your dataset (`HHS.HHS_OTJ_Raw`), you can use a combination of `PROC MEANS`, `PROC FREQ`, and `PROC UNIVARIATE` for statistics, and `PROC SGPLOT` or `PROC GCHART` for graphs.

Here's a breakdown of the steps we'll take:

**1. Summary of Numeric Variables:**

We'll use `PROC MEANS` for basic statistics like mean, standard deviation, min, max, and number of observations.

**2. Summary of Non-Numeric Variables:**

We'll use `PROC FREQ` to count the frequency of each level for categorical variables (non-numeric).

**3. Bonus Graphs:**

- For numeric variables, we can use `PROC SGPLOT` to create histograms.
- For non-numeric variables, we can use bar charts to visualize their distribution.

○ ChatGPT Code:

```
* Step 1: Summary of numeric variables;
proc means data=HHS.HHS_OTJ_Raw n mean std min max;
    var _numeric_; * This will automatically include all
    numeric variables;
run;

* Step 2: Summary of non-numeric variables;
proc freq data=HHS.HHS_OTJ_Raw;
    tables _character_; * This will automatically include
    all character variables;
run;

* Step 3: Bonus Graph - Histogram for numeric variables;
proc sgplot data=HHS.HHS_OTJ_Raw;
    histogram _numeric_ / scale=count;
    title "Histograms of Numeric Variables";
run;

* Step 4: Bonus Graph - Bar chart for non-numeric
(categorical) variables;
proc sgplot data=HHS.HHS_OTJ_Raw;
    vbar _character_ / groupdisplay=cluster;
    title "Bar Chart of Non-Numeric (Categorical)
Variables";
run;
```

● ChatGPT Response, Part 2

## Explanation:

### 1. Summary of Numeric Variables ( PROC MEANS ):

- `n` gives the number of non-missing values.
- `mean` gives the average of the numeric variable.
- `std` gives the standard deviation.
- `min` and `max` give the minimum and maximum values.

The `var _numeric_;` statement tells SAS to include all numeric variables in the dataset.

### 2. Summary of Non-Numeric Variables ( PROC FREQ ):

- This procedure generates a frequency table for all categorical (non-numeric) variables in the dataset.
- The `tables _character_;` statement tells SAS to include all character variables in the dataset.

### 3. Bonus Graphs:

- **Histogram for Numeric Variables:**

`PROC SGPLOT` creates histograms for each numeric variable in the dataset.

- **Bar Chart for Non-Numeric Variables:**

`PROC SGPLOT` is also used to generate bar charts for non-numeric (categorical) variables, showing how often each category occurs.

## Customization:

- If you want to focus on specific variables, you can replace `_numeric_` or `_character_` with a list of variable names. For example, `var var1 var2;` if you're only interested in `var1` and `var2`.
- You can modify the `proc sgplot` steps to adjust for multiple histograms or bar charts (for example, by using a `by` statement or looping over variables if needed).

Let me know how this works for you or if you'd like any further adjustments!

- Let's add that code to our ever-evolving SAS program! The title section can be something like "Explore Data Set" and it'll look like the following:

- Go ahead and run that code!
  - What do you see? I see a WHOLE lot of output:

Log    Results    Output Data (1)

Global Academic Programs Workshop  
Learning SAS Programming with Help from ChatGPT

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
State_FIP	141515	28.4641345	15.6774686	1.0000000	56.0000000
YearQuarter	141515	21476.39	842.2515547	20089.00	22919.00
Unemp	141515	0.1798891	0.3840964	0	1.0000000
in_LF	141515	0.6195739	0.4854933	0	1.0000000
WTFINL	141515	43356.65	102299.40	123.3032000	2366665.82

File = SAS Programming + ChatGPT

---

Global Academic Programs Workshop  
Learning SAS Programming with Help from ChatGPT

The FREQ Procedure

State_Name	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Alabama	2958	2.09	2958	2.09
Alaska	2589	1.83	5547	3.92
Arizona	3022	2.14	8569	6.06
Arkansas	2951	2.09	11520	8.14
California	4150	2.93	15670	11.07
Colorado	2550	1.80	18220	12.87
Connecticut	2541	1.80	20761	14.67
Delaware	2692	1.90	23453	16.57
District of Columbia	2614	1.85	26067	18.42
Florida	3810	2.69	29877	21.11

- But wait: I also see an error in the Log. Actually a couple of them:

Log    Results    Output Data (1)

(X) Errors (2) (A) Warnings (0) (N) Notes (29)

(X) ERROR: Variable \_NUMERIC\_ not found.  
(X) ERROR: Variable \_CHARACTER\_ not found.

NOTE: PROCEDURE FREQ used (total process time):  
real time 0.10 seconds  
cpu time 0.10 seconds

```

166
167 * Step 3: Bonus Graph - Histogram for numeric variables;
168 proc sgplot data=HHS.HHS_OTJ_Raw;
169   histogram _numeric_ / scale=count;
ERROR: Variable _NUMERIC_ not found.
170   title "Histograms of Numeric Variables";
171 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

172
173 * Step 4: Bonus Graph - Bar chart for non-numeric (categorical) variables;
174 proc sgplot data=HHS.HHS_OTJ_Raw;
175   vbar _character_ / groupdisplay=cluster;
ERROR: Variable _CHARACTER_ not found.
176   title "Bar Chart of Non-Numeric (Categorical) Variables";
177 run;
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE SGПPLOT used (Total process time):
      real time 0.00 seconds
      cpu time 0.00 seconds

SYMBOLGEN: Macro variable GRAPHTERM resolves to options noaccessible;
178

```

- Yikes. Looks like – once again - ChatGPT got us close but not the complete code. We'll need to refine that code
  - ChatGPT prompt:  
For step 3 and step 4, I'm getting the error message that `_numeric_` and `_character_` are not found. Can you write that code another way?
- ChatGPT response:

Thanks for letting me know! The issue arises because `_numeric_` and `_character_` are not valid variable names but are used in `PROC MEANS` and `PROC FREQ` to represent all numeric and character variables, respectively. However, for `PROC SGPLOT`, we need to explicitly list the variables.

Let's adjust the code to handle that properly by explicitly specifying the numeric and character variables for the histograms and bar charts.

### New SAS Code:

1. For numeric variables, we will first create a list of all numeric variables and then plot them.
2. For non-numeric variables, we will similarly create a list of character variables and plot them.

### Updated Code:

```
SAS
Copy

* Step 1: Identify the numeric variables and summarize them;
proc contents data=HHS.HHS_OTJ_Raw out=vars(keep=name type) noprint;
run;

* Step 2: Get the numeric variables;
proc sql noprint;
  select name into :num_vars separated by ' '
  from vars
  where type=1; /* Numeric variables are of type 1 */
quit;

* Step 3: Summary of numeric variables;
proc means data=HHS.HHS_OTJ_Raw n mean std min max;
  var &num_vars; /* Use the numeric variables list */
run;

* Step 4: Identify the non-numeric variables and summarize them;
proc sql noprint;
  select name into :char_vars separated by ' '
  from vars
  where type=2; /* Character variables are of type 2 */
quit;
```

```

* Step 5: Summary of non-numeric variables (categorical);
proc freq data=HHS.HHS_OTJ_Raw;
    tables &char_vars; /* Use the character variables list */
run;

* Step 6: Bonus Graph - Histogram for numeric variables;
%macro plot_histograms;
    %let num_vars_count = %sysfunc(countw(&num_vars));
    %do i = 1 %to &num_vars_count;
        %let var = %scan(&num_vars, &i);
        proc sgplot data=HHS.HHS_OTJ_Raw;
            histogram &var;
            title "Histogram of &var";
        run;
    %end;
%mend;

```

```

%plot_histograms; /* Call the macro to plot histograms for numeric variables */

* Step 7: Bonus Graph - Bar chart for non-numeric (categorical) variables;
%macro plot_barcharts;
    %let char_vars_count = %sysfunc(countw(&char_vars));
    %do j = 1 %to &char_vars_count;
        %let var = %scan(&char_vars, &j);
        proc sgplot data=HHS.HHS_OTJ_Raw;
            vbar &var / response=count stat=freq;
            title "Bar Chart of &var";
        run;
    %end;
%mend;

%plot_barcharts; /* Call the macro to plot bar charts for non-numeric variables */

```

- Yikes. That got real, quick. If you're an advanced programmer, then that macro approach could be useful. But a beginner would get lost if that code didn't work exactly as intended. So, pretend that you already know a bit about proc univariate and proc freq – and the fact that they can easily produce graphs. Then you could write the following prompt:

Hmmm. That code is very complex. Let's try again. Can you add plots to the proc univariate and proc freq statements to produce graphs for the numeric and non-numeric variables, respectively?

- ChatGPT response:

You're right! Let's simplify things by adding graphs directly within `PROC UNIVARIATE` (for numeric variables) and `PROC FREQ` (for non-numeric variables), which will automatically generate the plots we need.

Here's the revised and simplified code:

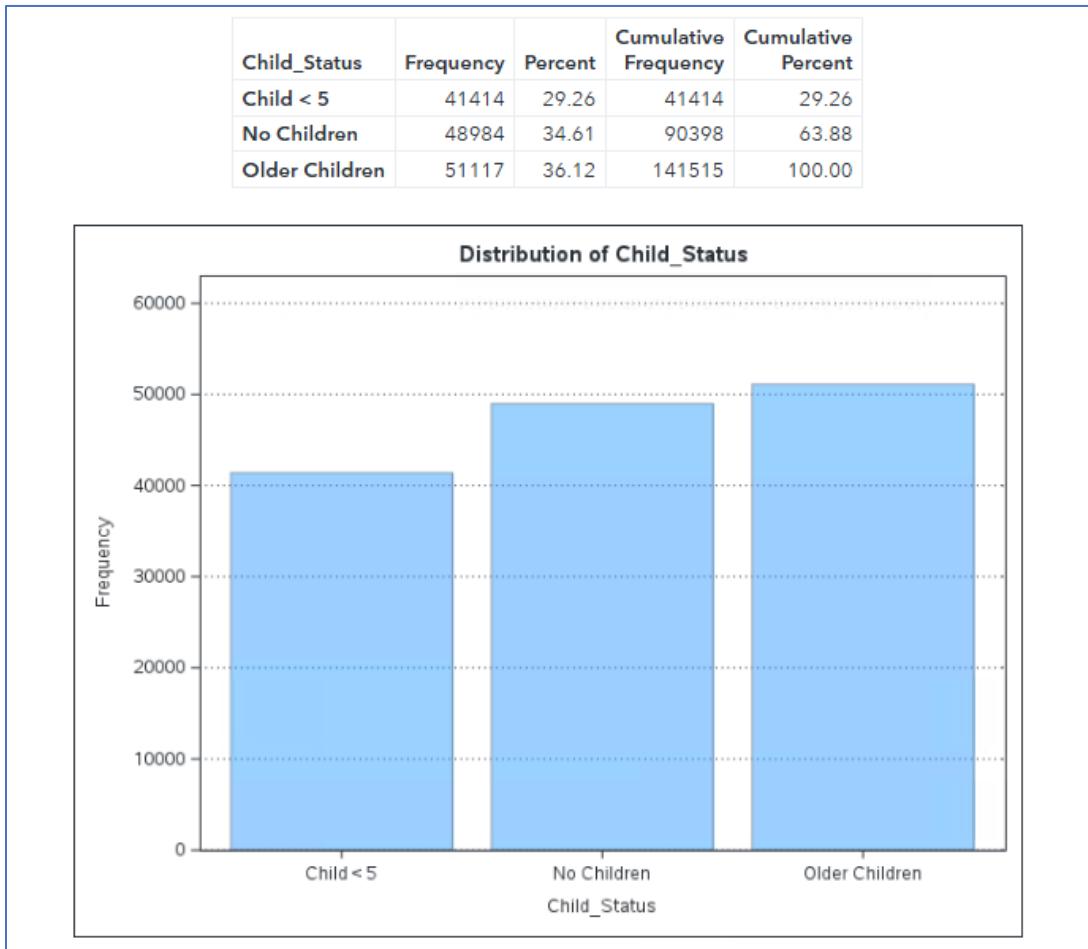
- ChatGPT code:

```
* Step 1: Summary of numeric variables with graphs;  
proc univariate data=HHS.HHS_OTJ_Raw;  
    var _numeric_; /* Automatically includes all numeric variables */  
    histogram / normal; /* Adds a histogram with a normal curve for each  
        numeric variable */  
    inset mean std min max / position=ne; /* Adds summary statistics to the plot  
        */  
run;  
  
* Step 2: Summary of non-numeric variables with graphs;  
proc freq data=HHS.HHS_OTJ_Raw;  
    tables _character_ / plots=freqplot; /* Automatically includes all character  
        variables and generates a frequency plot */  
run;
```

- Delete the old Step 3 and Step 4 and add the new code from this iteration of our evolving conversation with ChatGPT. Code would appear as:

```
33 *-----|  
34 | Explore Data Set |  
35 *-----*;  
36 * Step 1: Summary of numeric variables;  
37 ⊕ proc means data=HHS.HHS_OTJ_Raw n mean std min max;  
38     var _numeric_; * This will automatically include all numeric variables;  
39 run;  
40  
41 * Step 2: Summary of non-numeric variables;  
42 ⊕ proc freq data=HHS.HHS_OTJ_Raw;  
43     tables _character_; * This will automatically include all character variables;  
44 run;  
45  
46 * Step 3: Bonus Graph - Histogram for numeric variables;  
47 ⊕ proc univariate data=HHS.HHS_OTJ_Raw;  
48     var _numeric_; /* Automatically includes all numeric variables */  
49     histogram / normal; /* Adds a histogram with a normal curve for each numeric variable */  
50     inset mean std min max / position=ne; /* Adds summary statistics to the plot */  
51 run;  
52  
53  
54 * Step 4: Bonus Graph - Bar chart for non-numeric (categorical) variables;  
55 ⊕ proc freq data=HHS.HHS_OTJ_Raw;  
56     tables _character_ / plots=freqplot; /* Automatically includes all character variables and generates a freq  
57 run;  
58
```

- And after submitting you should have a working program and some output to digest. A sample:



**Point:** ChatGPT can get complex quickly. And knowing a little bit about the programming language can help you get to more efficient code faster.

- Other considerations (... in this section)
  - Do you now understand your data a bit better? Anything you'd change +/- 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, add the MAXDEC=2 option to clean-up the first table
    - This shows the dynamic program editor (aka autocomplete) in SAS Studio

The screenshot shows a portion of a SAS program in the SAS Studio interface. The code includes several PROC statements:

```

35  *-----*;
36  * Step 1: Summary of numeric variables;
37  proc means data=HHS.HHS_OTJ Raw n mean std min max maxdec;
38    var _numeric_; /* This will automatically include
39    run;
40
41  * Step 2: Summary of non-numeric variables;
42  proc freq data=HHS.HHS_OTJ_Raw;
43    tables _character_; /* This will automatically
44    run;
45
46  * Step 3: Bonus Graph - Histogram for numeric variables;
47  proc univariate data=HHS.HHS_OTJ_Raw;
48    var _numeric_; /* Automatically includes all numeric
49    histogram / normal; /* Adds a histogram with a
50    inset mean std min max / position=ne; /* Adds
51    run;
52
53
54  * Step 4: Bonus Graph - Bar chart for non-numeric (categorical) variables;
55  proc freq data=HHS.HHS_OTJ_Raw;
56    tables _character_ / plots=freqplot; /* Automatically includes all character variables a
57

```

A tooltip is displayed over the line `MAXDEC=`, which is highlighted with a red box. The tooltip contains the following information:

- Keyword:** MAXDEC\_=
- Context:** [PROC MEANS] MAXDEC= o
- [Syntax:** MAXDEC=number]
- Specifies the maximum number of decimal digits displayed output.**
- Search:** [Product Documentation](#) [Search](#)

- And that's a great place to stop for now.

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, double-pretend that 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 grab the code here:

The screenshot shows a GitHub repository page for the project "Better-Prompt-Engineering-Leveraging-ChatGPT-to-get-started-with-SAS-coding". The repository contains several files, including "HHS + OTJ - State Level Estimates using SQL.sas", which is the file being viewed. The code in this file is a large SQL script designed to produce state-level estimates. A red box highlights the beginning of the script, specifically the first few lines and the start of the "proc sql" block. The code is as follows:

```

1  *-----*;
2  | Produce State-Level Estimates
3  |-----*
4  **** By State ;
5
6 **** By Labor Force Status | All *
7  proc sql;
8    create table hhs.covid_labor_supply as
9      select distinct state_fip, state_name,
10        year(yearquarter) as Year format 9.,
11
12  **** By Labor Force Status | All *
13  sum( ( unemp-1 ) * WTFINL )
14  sum( ( in_LF-1 ) * WTFINL )           / sum( ( in_LF-1 ) * WTFINL )
15
16
17  **** By Education *
18
19  **** Unemployment *
20  sum( ( educ_ltd="High School Diploma" ) * ( unemp-1 ) * WTFINL )           / sum( ( educ_ltd="High School Diploma" ) * ( in_LF-1 ) * WTFINL ) as UE_High_School_Diploma
21  sum( ( educ_ltd="Some College" ) * ( unemp-1 ) * WTFINL )                   / sum( ( educ_ltd="Some College" ) * ( in_LF-1 ) * WTFINL ) as UE_Some_College
22  sum( ( educ_ltd="College +" ) * ( unemp-1 ) * WTFINL )                      / sum( ( educ_ltd="College +" ) * ( in_LF-1 ) * WTFINL ) as UE_College_Plus
23
24  **** LFP *
25  sum( ( educ_ltd="High School Diploma" ) * ( in_LF-1 ) * WTFINL )           / sum( ( educ_ltd="High School Diploma" ) * WTFINL )
26  sum( ( educ_ltd="Some College" ) * ( in_LF-1 ) * WTFINL )                   / sum( ( educ_ltd="Some College" ) * WTFINL ) as LFP_Some_College

```

- Another nice feature of ChatGPT is that it can help explain 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 smaller 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, in parts:**

- Part 1: general overview

Let's break down this `PROC SQL` code in SAS and explain it step by step in a way that's easier to understand, like we're explaining it to a 10-year-old. The main goal of this code is to take a large dataset and create a smaller summary dataset with new variables that show the unemployment rate and labor force participation (LFP) rate for different groups, broken down by various factors.

#### What the code is doing:

1. **Creating a New Dataset:** The `CREATE TABLE hhs.covid_labor_supply AS` part is creating a new table called `covid_labor_supply` in the `hhs` library. This table will hold the summarized data after the calculations.
2. **Selecting Variables:** The `SELECT DISTINCT` part ensures that each combination of `state_fip`, `state_name`, and `Year` appears only once in the new table. These three variables will be used to group the data.
3. **Year Calculation:** The code is taking the `yearquarter` variable (which likely represents the year and quarter, like "2020Q1") and extracting just the year using the `year()` function. This creates a new variable called `Year`.

- Part 2: Labor Force Status

## Labor Force Status Summaries

The code then summarizes various labor force statistics. There are many calculations, but they follow a pattern. Here's a quick breakdown of the types of calculations:

### 1. Unemployment Rate (UE) for Women:

- `sum((unemp=1) * WTFINL)` : This part adds up the values of `WTFINL` (likely a weight or population count) for women who are unemployed (`unemp=1`).
- `sum((in_LF=1) * WTFINL)` : This adds up the values of `WTFINL` for women who are in the labor force (`in_LF=1`).
- `as UE_Women` : This creates a new variable called `UE_Women`, which is the unemployment rate for women. It is the sum of unemployed women divided by the sum of women in the labor force, which gives the unemployment rate.

### 2. Labor Force Participation (LFP) Rate for Women:

- `sum((in_LF=1) * WTFINL)` : This adds up the values of `WTFINL` for women who are in the labor force.
- `sum(WTFINL)` : This adds up the values of `WTFINL` for all women (whether they are in the labor force or not).
- `as LFP_Women` : This creates a new variable called `LFP_Women`, which is the labor force participation rate for women. It is the sum of women in the labor force divided by the total number of women.

### ○ Part 3: Other Calculations

#### Breaking Down the Other Calculations:

The code then repeats similar calculations for different subgroups based on **education level** and **child status**. For each of these groups, it calculates the **unemployment rate** and **labor force participation rate** separately.

#### By Education:

- `educ_ltd="High School Diploma"` : This part checks for women with less than or equal to a high school education.
- `educ_ltd="Some College"` : This part checks for women with some college education.
- `educ_ltd="College +"` : This part checks for women with a college degree or higher.

The code does the same calculations for these subgroups as it did for the overall women, but now it's dividing the population by education level.

#### By Child Status:

- `child_status="No Children"` : Women who do not have children.
- `child_status="Older Children"` : Women who have older children.
- `child_status="Child < 5"` : Women who have children under the age of 5.

Again, it calculates the unemployment and labor force participation rates separately for these groups.

#### **GROUP BY and ORDER BY:**

- `GROUP BY 1,2,3` : This means the data is being grouped by the first three variables in the `SELECT` statement: `state_fip`, `state_name`, and `Year`. This ensures that the statistics are calculated for each combination of these three variables.
- `ORDER BY 1,2,3` : This sorts the resulting table by `state_fip`, `state_name`, and `Year` in ascending order.

- Why we're doing all this...

#### **Final Output:**

The output of this SQL code will be a table (`covid_labor_supply`) where each row represents a unique combination of `state_fip`, `state_name`, and `Year`, and includes the following variables:

- **Unemployment rates and Labor Force Participation rates for women**, broken down by:
  - Overall (all women).
  - By education level (high school, some college, college+).
  - By child status (no children, older children, young children).

- Summary

#### **The Simplified Version:**

To sum it up in a simpler way:

- We start with a huge dataset.
- We calculate unemployment and labor force participation rates for women, but we do it for different groups:
  - All women.
  - Women with different education levels (high school, some college, college+).
  - Women with different child statuses (no children, older children, young children).
- We group and sort this information by state and year.

#### **Why this is useful:**

This code is transforming a large dataset into a smaller, more manageable one by calculating key statistics (like unemployment rates) for women in different states, years, and subgroups (based on education and child status). This summary data can then be used to see trends and patterns over time, by state, and for different groups of women.

Let me know if this clarifies the code or if you'd like further explanation on any part!

- Gotta admit, that was pretty cool + useful.
  - Let's add that **Produce State-Level Estimates** code to your SAS program.
  - And submit it.
  - You'll hopefully have a new + much smaller data set to examine:

```

57 RUN;
58
59
60
61
62
63
64
65
66 ****
67 proc sql;
68   create table hhs.covid_labor_supply as
69     select distinct state_fip, state_name,
70       year(yearquarter) as Year format 9.,
71
72     sum( (unemp=1) * WTFINL ) / sum( (in_LF=1) * WFTINL ) as
73     Labor Force Status | All /*/
74     sum( (in_LF=8) * WFTINL ) / sum( (in_LF=1) * WFTINL ) as
75
76     Labor Force Status | Education /*/
77
78     /*****
79     sum( ( educ_ltd="High School Diploma" ) * ( unemp=1 ) * WFTINL ) / sum( ( educ_ltd="High School Diploma" ) * ( in_LF=1 ) * WFTINL )
80     sum( ( educ_ltd="Some College" ) * ( unemp=1 ) * WFTINL ) / sum( ( educ_ltd="Some College" ) * ( in_LF=1 ) * WFTINL )
81     sum( ( educ_ltd="College+" ) * ( unemp=1 ) * WFTINL ) / sum( ( educ_ltd="College+" ) * ( in_LF=1 ) * WFTINL )
82
83     *****/
84     sum( ( educ_ltd="High School Diploma" ) * ( in_LF=1 ) * WFTINL ) / sum( ( educ_ltd="High School Diploma" ) * ( in_LF=1 ) * WFTINL )
85     sum( ( educ_ltd="Some College" ) * ( in_LF=1 ) * WFTINL ) / sum( ( educ_ltd="Some College" ) * ( in_LF=1 ) * WFTINL )
86     sum( ( educ_ltd="College+" ) * ( in_LF=1 ) * WFTINL ) / sum( ( educ_ltd="College+" ) * ( in_LF=1 ) * WFTINL )
87
88     *****/
89     sum( ( child_status="No Children" ) * ( unemp=1 ) * WFTINL ) / sum( ( child_status="No Children" ) * ( in_LF=1 ) * WFTINL )
90     sum( ( child_status="Older Children" ) * ( unemp=1 ) * WFTINL ) / sum( ( child_status="Older Children" ) * ( in_LF=1 ) * WFTINL )
91     sum( ( child_status="Child < 5" ) * ( unemp=1 ) * WFTINL ) / sum( ( child_status="Child < 5" ) * ( in_LF=1 ) * WFTINL )
92
93     *****/
94     sum( ( child_status="No Children" ) * ( in_LF=1 ) * WFTINL ) / sum( ( child_status="No Children" ) * ( WFTINL ) )
95
96     *****/
97     sum( ( educ_ltd="High School Diploma" ) * ( unemp=1 ) * WFTINL ) as LFP_Women_HSLabel=>EDUC <> HS form
98     sum( ( educ_ltd="Some College" ) * ( unemp=1 ) * WFTINL ) as LFP_Women_SCollLabel=>Some Collag
99
100    sum( ( educ_ltd="College+" ) * ( unemp=1 ) * WFTINL ) as LFP_Women_CollLabel=>College
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
827
828
829
829
830
831
832
833
834
835
836
837
837
838
839
839
840
841
842
843
844
845
846
846
847
848
848
849
849
850
851
852
853
854
855
856
856
857
858
858
859
859
860
861
862
863
864
865
866
866
867
868
868
869
869
870
871
872
873
874
875
876
876
877
878
878
879
879
880
881
882
883
884
885
885
886
886
887
887
888
888
889
889
890
891
892
893
893
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
158
```

```

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;
    title "Summary Statistics for New Variables in the covid_labor_supply Dataset";
run;

/* Step 2: Use PROC FREQ to summarize categorical variables (if any, like state and year)
*/
proc freq data=hhs.covid_labor_supply;
    tables state_fip state_name Year;
    title "Frequency Distribution for State and Year Variables";
run;

```

- ChatGPT Additional Details

### Explanation:

1. PROC MEANS :

- This will give us a summary of the **numeric variables** that represent the labor force participation (LFP) rates and unemployment (UE) rates.
- **Statistics:** `n` (number of non-missing observations), `mean`, `std` (standard deviation), `min`, `max`, and `median`.
- The variables we're summarizing are those calculated in your SQL code (`UE_Women`, `LFP_Women`, `UE_Women_HS`, etc.).

2. PROC FREQ :

- This will summarize the **categorical variables** like `state_fip`, `state_name`, and `Year`.
- We use `tables` to display the frequency distribution for each of these variables.

### What the Output Will Look Like:

- **For the numeric variables**, you will get a table with the mean, standard deviation, min, max, and median for each of the rates (e.g., `UE_Women`, `LFP_Women`, `UE_Women_HS`, etc.).
- **For the categorical variables**, you will get a frequency distribution for the `state_fip`, `state_name`, and `Year` variables, showing how many records exist for each state and year.

### Sample Output Table from PROC MEANS :

Variable	N	Mean	Std Dev	Min	Max	Median
UE_Women	1000	0.084	0.022	0.01	0.20	0.08
LFP_Women	1000	0.74	0.15	0.45	0.95	0.75
UE_Women_HS	1000	0.09	0.03	0.02	0.18	0.09
LFP_Women_HS	1000	0.70	0.12	0.50	0.85	0.72
...	...	...	...	...	...	...

### Sample Output Table from PROC FREQ :

state_fip	Frequency	Percent
01	50	5.0%
02	60	6.0%
...	...	...

This code will provide you with a concise summary of the key variables from your dataset. Let me know if you need any further customizations or have additional questions!

- For our adventure, grab the code and add it to your program. For the PROC MEANS statement, add that MAXDEC=2 option. And check out your program:

```

108 *-----*
109 |          Summarize State-Level Estimates          |
110 *-----*;
111 /* Summarizing the new variables in the hhs.covid_labor_supply dataset */
112
113 /* Step 1: Use PROC MEANS to summarize the numeric variables (rates) */
114 ⊕ proc means data=hhs.covid_labor_supply n mean std min max median maxdec=2;
115   var UE_Women LFP_Women
116     UE_Women_HS UE_Women_SCollege UE_Women_CollegeP
117     LFP_Women_HS LFP_Women_SCollege LFP_Women_CollegeP
118     UE_Women_NoKids UE_Women_OlderKids UE_Women_YoungKids
119     LFP_Women_NoKids LFP_Women_OlderKids LFP_Women_YoungKids;
120   title "Summary Statistics for New Variables in the covid_labor_supply Dataset";
121 run;
122
123 /* Step 2: Use PROC FREQ to summarize categorical variables (if any, like state and year) */
124 ⊕ proc freq data=hhs.covid_labor_supply;
125   tables state_fip state_name Year;
126   title "Frequency Distribution for State and Year Variables";
127 run;
128

```

- Next, submit your program and examine those new summary statistics. These will be weighted population averages at the state-level, and should make a bit more sense:

Summary of Labor Statistics						
The MEANS Procedure						
Variable	Label	N	Mean	Std Dev	Minimum	Maximum
UE_Women	Unemployment Rate	408	0.04	0.02	0.01	0.13
LFP_Women	LFP Rate	408	0.77	0.04	0.67	0.86
UE_Women_HS	EDUC <= HS	408	0.06	0.03	0.01	0.22
UE_Women_SCollege	Some College	408	0.05	0.02	0.01	0.18
UE_Women_CollegeP	College +	408	0.02	0.01	0.00	0.08
LFP_Women_HS	EDUC <= HS	408	0.69	0.05	0.56	0.82
LFP_Women_SCollege	Some College	408	0.77	0.04	0.67	0.88
LFP_Women_CollegeP	College +	408	0.85	0.03	0.72	0.92
UE_Women_NoKids	No Children	408	0.04	0.02	0.01	0.12
UE_Women_OlderKids	Older Children	408	0.04	0.02	0.01	0.15
UE_Women_YoungKids	Young Children	408	0.04	0.02	0.01	0.13
LFP_Women_NoKids	No Children	408	0.79	0.05	0.64	0.89
LFP_Women_OlderKids	Older Children	408	0.78	0.05	0.68	0.91
LFP_Women_YoungKids	Young Children	408	0.68	0.06	0.50	0.83

### One Final Item: Let's plot state-level trends over time!

- For our 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 Response

Sure! To examine the unemployment rates (UE) for women in different categories (`UE_Women`, `UE_Women_NoKids`, `UE_Women_OlderKids`, `UE_Women_YoungKids`) by state over time, we can create a line plot to visualize how these unemployment rates change across states over the years.

We'll use `PROC SGPlot` in SAS to create a nice line plot with `year` on the x-axis, `UE_Women`, `UE_Women_NoKids`, `UE_Women_OlderKids`, and `UE_Women_YoungKids` as the y-axis values, and `state_name` as a grouping variable.

Here's the SAS code to achieve this:

- ChatGPT Code

```
/* Step 1: Create a plot of UE_Women by state over time */
proc sgplot data=hhs.covid_labor_supply;
    series x=Year y=UE_Women / group=state_name lineattrs=(pattern=solid) markers;
    title "Unemployment Rate for Women (UE_Women) by State Over Time";
    xaxis label="Year" grid;
    yaxis label="Unemployment Rate" grid;
run;

/* Step 2: Create a plot of UE_Women_NoKids by state over time */
proc sgplot data=hhs.covid_labor_supply;
    series x=Year y=UE_Women_NoKids / group=state_name lineattrs=(pattern=solid)
    markers;
    title "Unemployment Rate for Women (No Kids) by State Over Time";
    xaxis label="Year" grid;
    yaxis label="Unemployment Rate (No Kids)" grid;
run;

/* Step 3: Create a plot of UE_Women_OlderKids by state over time */
proc sgplot data=hhs.covid_labor_supply;
    series x=Year y=UE_Women_OlderKids / group=state_name lineattrs=(pattern=solid)
    markers;
    title "Unemployment Rate for Women (Older Kids) by State Over Time";
    xaxis label="Year" grid;
    yaxis label="Unemployment Rate (Older Kids)" grid;
run;

/* Step 4: Create a plot of UE_Women_YoungKids by state over time */
proc sgplot data=hhs.covid_labor_supply;
    series x=Year y=UE_Women_YoungKids / group=state_name lineattrs=(pattern=solid)
    markers;
    title "Unemployment Rate for Women (Young Kids) by State Over Time";
    xaxis label="Year" grid;
```

```
      yaxis label="Unemployment Rate (Young Kids)" grid;  
run;
```

- **ChatGPT Additional Details**

**Explanation:**

1. `PROC SGPlot :`

- This is used for generating plots in SAS.
- `series` : Creates a line plot where `x=Year` (year) and `y=UE_Women` , `UE_Women_NoKids` , `UE_Women_OlderKids` , or `UE_Women_YoungKids` depending on the plot.
- `group=state_name` : This groups the data by `state_name` , so each state will have its own line in the plot.
- `lineattrs=(pattern=solid)` : This option ensures that the lines are solid, but you can customize this further if needed.
- `markers` : Adds markers to each data point on the line.

2. `xaxis label="Year" grid; :`

- This adds a label to the x-axis ("Year") and adds a grid to make the plot easier to read.

3. `yaxis label="Unemployment Rate" :`

- This labels the y-axis (which represents the unemployment rate) and adds a grid for easier reading.

**What the Output Will Look Like:**

- The code will generate **four separate line plots**, each showing how the unemployment rate for women (broken down into the four categories: overall women, women with no kids, women with older kids, and women with young kids) changes over time for each state.
- Each state will be represented by a line, and you'll be able to compare trends across different states over the years.

If you'd like these plots combined into one graph, we can adjust the code to show multiple variables on a single plot with different colored lines. Let me know if you'd like that or need any further adjustments!

- Copy and paste that code into your program. Give it a section title like *Plots over time by State*:

```

130  *-----*  

131  |          Plots over time by State          |  

132  *-----*;  

133 /* Step 1: Create a plot of UE_Women by state over time */  

134 ⊕ proc sgplot data=hhs.covid_labor_supply;  

135   series x=Year y=UE_Women / group=state_name lineattrs=(pattern=solid) markers;  

136   title "Unemployment Rate for Women (UE_Women) by State Over Time";  

137   xaxis label="Year" grid;  

138   yaxis label="Unemployment Rate" grid;  

139 run;  

140  

141 /* Step 2: Create a plot of UE_Women_NoKids by state over time */  

142 ⊕ proc sgplot data=hhs.covid_labor_supply;  

143   series x=Year y=UE_Women_NoKids / group=state_name lineattrs=(pattern=solid) markers;  

144   title "Unemployment Rate for Women (No Kids) by State Over Time";  

145   xaxis label="Year" grid;  

146   yaxis label="Unemployment Rate (No Kids)" grid;  

147 run;  

148  

149 /* Step 3: Create a plot of UE_Women_OlderKids by state over time */  

150 ⊕ proc sgplot data=hhs.covid_labor_supply;  

151   series x=Year y=UE_Women_OlderKids / group=state_name lineattrs=(pattern=solid) markers;  

152   title "Unemployment Rate for Women (Older Kids) by State Over Time";  

153   xaxis label="Year" grid;  

154   yaxis label="Unemployment Rate (Older Kids)" grid;  

155 run;  

156  

157 /* Step 4: Create a plot of UE_Women_YoungKids by state over time */  

158 ⊕ proc sgplot data=hhs.covid_labor_supply;  

159   series x=Year y=UE_Women_YoungKids / group=state_name lineattrs=(pattern=solid) markers;  

160   title "Unemployment Rate for Women (Young Kids) by State Over Time";  

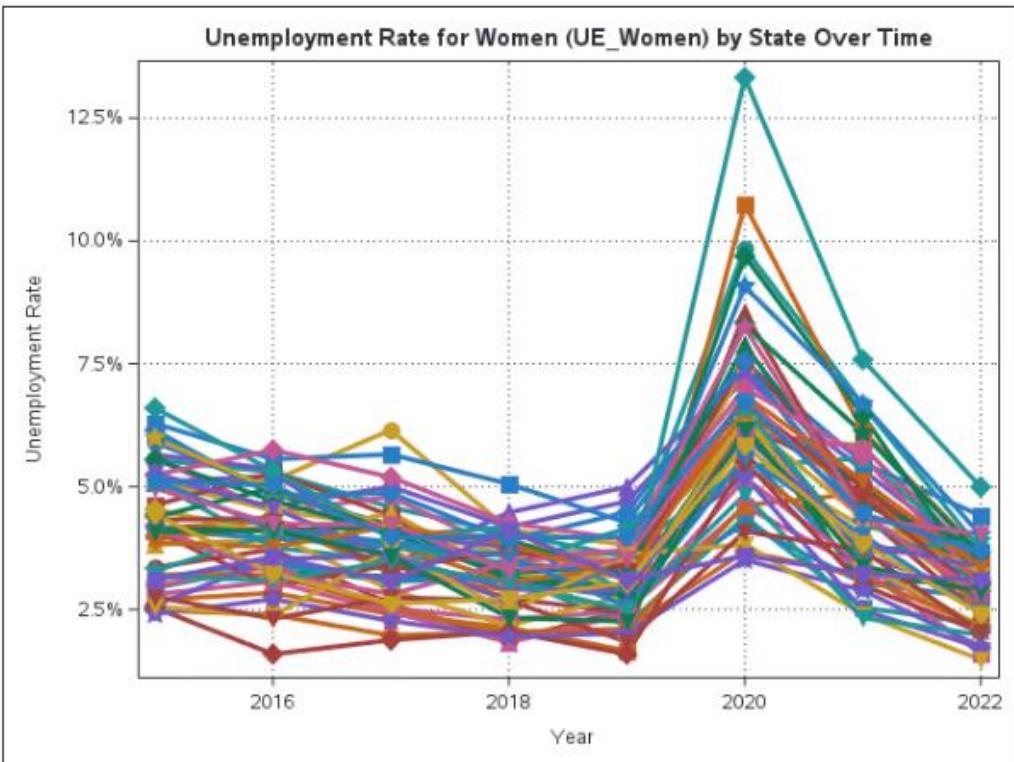
161   xaxis label="Year" grid;  

162   yaxis label="Unemployment Rate (Young Kids)" grid;  

163 run;

```

- Submit the code and examine the output.
  - Should we make any changes?



- Umm, yeah. What the heck just happened? Let's refine the **ChatGPT prompt**:
  - I don't actually see that the code you provided grouped the data by state\_name. Can you please try again and create new code which produces separate graphs for each state? Thank you!
- ChatGPT Response, Part 1:

Ah, I see what you're aiming for! If you'd like to produce **separate graphs for each state**, we can use the **BY** statement in SAS. This allows us to create a separate plot for each `state_name` in the dataset.

Here's how we can modify the code to generate a **separate graph for each state**, while examining the unemployment rates for women (`UE_Women`, `UE_Women_NoKids`, `UE_Women_OlderKids`, `UE_Women_YoungKids`) over time.

- Chat GPT Code:

```
/* Step 1: Sort the data by state_name to use the BY statement */
proc sort data=hhs.covid_labor_supply;
  by state_name;
run;

/* Step 2: Create a plot for UE_Women by state over time */
proc sgplot data=hhs.covid_labor_supply;
  by state_name; /* This will create separate plots for each state */
  series x=Year y=UE_Women / lineattrs=(pattern=solid) markers;
  title "Unemployment Rate for Women (UE_Women) by State Over Time";
  xaxis label="Year" grid;
  yaxis label="Unemployment Rate" grid;
run;

/* Step 3: Create a plot for UE_Women_NoKids by state over time */
proc sgplot data=hhs.covid_labor_supply;
  by state_name; /* This will create separate plots for each state */
  series x=Year y=UE_Women_NoKids / lineattrs=(pattern=solid) markers;
  title "Unemployment Rate for Women (No Kids) by State Over Time";
  xaxis label="Year" grid;
  yaxis label="Unemployment Rate (No Kids)" grid;
run;

/* Step 4: Create a plot for UE_Women_OlderKids by state over time */
proc sgplot data=hhs.covid_labor_supply;
  by state_name; /* This will create separate plots for each state */

```

```

series x=Year y=UE_Women_OlderKids / lineattrs=(pattern=solid) markers;
title "Unemployment Rate for Women (Older Kids) by State Over Time";
xaxis label="Year" grid;
yaxis label="Unemployment Rate (Older Kids)" grid;
run;

/* Step 5: Create a plot for UE_Women_YoungKids by state over time */
proc sgplot data=hhs.covid_labor_supply;
by state_name; /* This will create separate plots for each state */
series x=Year y=UE_Women_YoungKids / lineattrs=(pattern=solid) markers;
title "Unemployment Rate for Women (Young Kids) by State Over Time";
xaxis label="Year" grid;
yaxis label="Unemployment Rate (Young Kids)" grid;
run;

```

- ChatGPT added detail:

#### Explanation of Changes:

1. `proc sort :`
  - Before using the `BY` statement, we first need to sort the dataset by `state_name`. This ensures that the `SGPlot` procedure will generate a separate graph for each state.
2. `by state_name; :`
  - The `BY` statement inside each `PROC SGPLOT` procedure will produce a separate graph for each value of `state_name` in the dataset.
  - For example, if you have 5 states, this will generate 5 separate plots—one for each state—showing how the unemployment rates for women change over time.
3. `series x=Year y=... :`
  - The `series` plot creates a line plot with `Year` on the x-axis and the respective unemployment rate (`UE_Women`, `UE_Women_NoKids`, `UE_Women_OlderKids`, or `UE_Women_YoungKids`) on the y-axis.
  - `lineattrs=(pattern=solid)` : Solid lines are used for the plot.
  - `markers` : Adds markers at each data point for better visibility.
4. `xaxis and yaxis :`
  - These statements define labels for the axes and include a grid for easy reading of the graph.

## Expected Output:

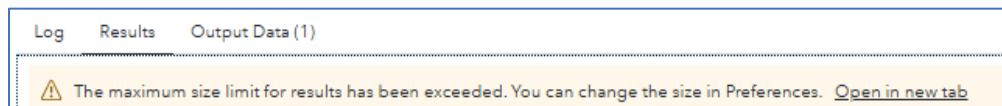
- The `BY` statement will create **separate plots for each state**.
- Each plot will display the **unemployment rate for women** over time, with lines for:
  - `UE_Women` (overall unemployment rate for women)
  - `UE_Women_NoKids` (unemployment rate for women with no kids)
  - `UE_Women_OlderKids` (unemployment rate for women with older kids)
  - `UE_Women_YoungKids` (unemployment rate for women with young kids)

## Key Takeaway:

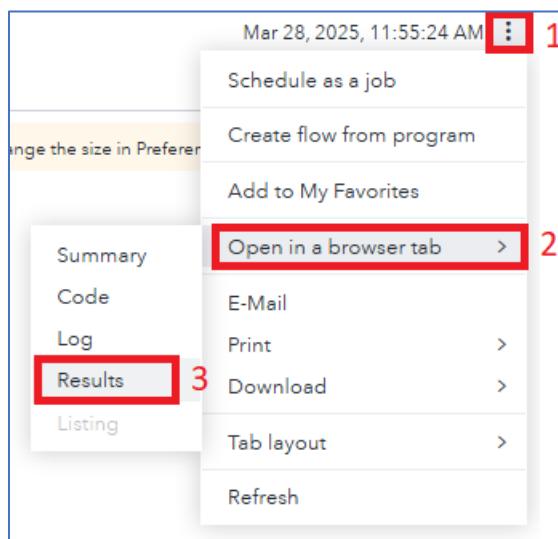
- After running this code, SAS will create **one plot per state**, so you'll be able to compare how the unemployment rate for women (and the different subgroups) changes over time for each state.

If there are too many states and you don't want to generate a separate plot for each, let me know and I can show you how to limit the number of plots or adjust the layout. Let me know if you need anything else!

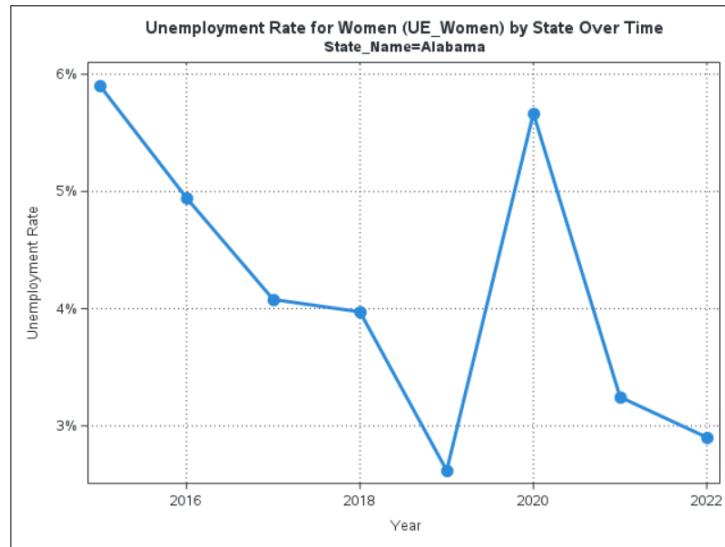
- Now, update your code to incorporate that *BY statement*. Submit the new code.
  - You may get the following message:



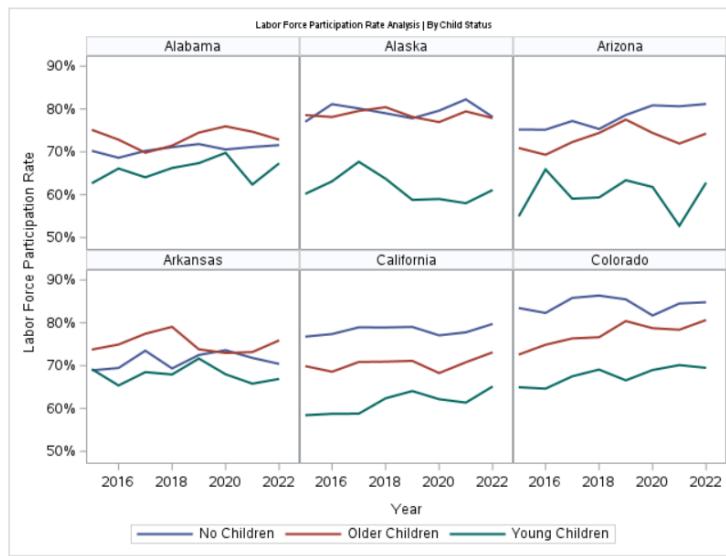
- If you do, no worries. Navigate to **More options** and select **Open in a browser tab** and then **Results**, with the following clicks:



- And there will be lots of output to explore. Like a lot:



- And while I don't love the fact that there is a single graph for each demographic group, but it's a starting point.
  - At a minimum, we can examine the general trends in the series.
  - As time goes on – and with a little research – we may see that PROC SGANEL is likely the better option here, which would produce the following:



- But that's simply a teaser for another day! We can rest on our laurels for now. Nice work!