The Simple Guide to SAS: Code Snippet Organizer

Code Snippet Organizer

Kirby Thomas

1 Contents

Tables

Figures

2 Introduction

2.1 Note of Encouragement

There is a common misconception that you must know everything there is to know about coding before you begin and that there is no room for errors. This could not be further from the truth. The best way to learn how to code is simply to start coding and to make a ton of mistakes along the way. Coding is messy, frustrating, and nonlinear, but there is nothing quite like the feeling of accomplishment after struggling with a program and ending up with your desired results.

2.2 Purpose of this Code Snippet Organizer

When I started programming in SAS, I struggled to find the best resource to help me learn. User Manuals were dense and hard to read. Videos were helpful but did not provide syntax and explanations I could refer back to. I relied heavily on Google, but it always took time to find the right article or blog post, and I would have to find that web page again the next time I needed to use that code or function. I cataloged my journey through various scattered web bookmarks and an unformatted, ever-growing Word document saved to my desktop with random code snippets and explanations. As I continued my coding journey, I relied heavily on this haphazard reference guide I had created, often searching for keywords to find the piece of code I needed.

I programmed like this for 8 years before a friend reached out to me for some tips on how to start coding with SAS. I opened my makeshift guide, and, as I stared bewildered at the unorganized wall of text, I realized that it was completely unhelpful to anyone but me. I decided that the best path forward was to do a complete overhaul of my document, organizing the information by topic, explaining common business problems and how to solve them using SAS, detailing common pitfalls to avoid, and providing example code. The document became the story of my SAS journey, which is still close to the beginning, and all the struggles and pitfalls I have encountered along the way. The result was *The Simple Guide to SAS – From Null to Novice*, which is intended to help new coders learn the basics and get started with using SAS.

I realized that if I had organized all of my code snippets and examples like this when I first started coding, it would have saved me a lot of time and frustration. So, I made this Code Snippet Organizer based on the book for you. It is meant to be a living, breathing document that helps you stay organized and grows with you along your SAS journey. Please add your thoughts, experiences, and code snippets that are relevant to your work to the appropriate sections. Then, use the Table of Contents to quickly find the code snippet you need. Simply paste the snippet into your code editor and update the code to reflect the data set and variables that you are working with.

3 Data Setup

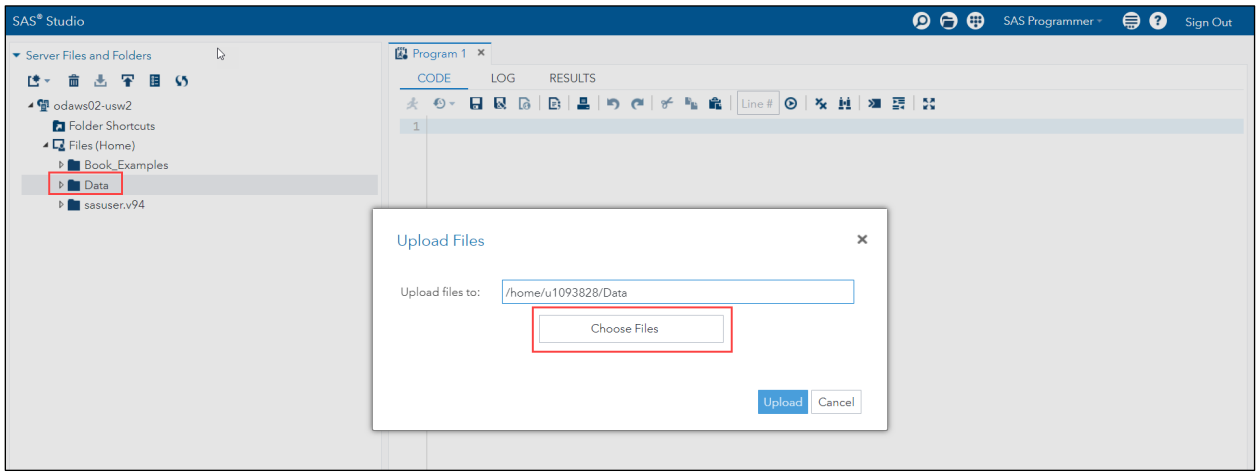
The data used in the companion book, *The Simple Guide to SAS – From Null to Novice*, is available for download from the SAS Author Page found at https://support.sas.com/en/books/authors/kirby-thomas.html. You can download this data to provide context for the code snippets in this document. Once the data has been downloaded, save it to your computer. I created a folder called Book_Data on my desktop and saved the files in a subfolder called Input_Data. View the properties of the folder that you save the book data to and make note of the location. When using SAS® Enterprise Guide® or the SAS Windowing environment, create a LIBREF called INPUT that references the location of your saved data files. In my case, I would begin my program with the following statement:

```
LIBNAME input 'C:\Users\Kirby\Desktop\Book_Data\Input_Data';
```

Update the location in single quotation marks to reflect where your data is saved.

If you are using SAS® Studio, you will need to upload the data files to the SAS server before you can work with them. First, create a folder to hold your book data. Right-click **Files** in the navigation pane and select **New** > **Folder**. I named my folder Data. Now, right-click the folder you just created, and click **Upload Files** (Figure 1).

Figure 1: Upload Files



When you click **Choose Files**, the file explorer opens and enables you to navigate to where you saved your data files for this book. You can select one data file or hold down the **Ctrl** button and select several files to upload them all at once. Next, click **Open**. A list of selected files appears in the Upload Files window, and you can click the **Upload** button to upload the files to the SAS server into your Data folder.

Right-click the **Data** folder that you uploaded the files to, and then click **Properties** to copy this location. In my case, the location is: /home/u1093828/Data. Create a LIBREF called INPUT at the beginning of your SAS® Studio programs to reference any files found in this folder location:

```
LIBNAME input '/home/u1093828/Data';
```

The INPUT LIBREF will be used throughout the code snippets to reference the location where the book data files are saved.

4 SAS Libraries

Add your code snippets for SAS Libraries. What libraries do you use most often?

4.1 Create Library

```
LIBNAME Libref 'path';
LIBNAME input 'C:\Users\Kirby\Desktop\Book_Data\Input_Data';
LIBNAME output 'C:\Users\Kirby\Desktop\Book_Data\Output_Data';
```

4.2 Clear Library

```
LIBNAME Libref CLEAR;
```

4.3 Set Macro Library

```
%let input = C:\Users\Kirby\Desktop\Book_Data\Input_Data;
%let output = C:\Users\Kirby\Desktop\Book_Data\Output_Data;
```

4.4 Create Ready Only Library

```
libname SF 'C:\Users\Kirby\Desktop\Book_Data\Input_Data'
ACCESS=READONLY;
```

5 Import/Export

Add your code snippets for importing/exporting data here. You may import or export data files in a data type (e.g., JSON, rdata) that isn't listed here. Add your new heading name and click the lower right-hand corner of the Styles box on the Home ribbon. Highlight your new heading name and click on the **Heading 3** format in the Styles box to apply that style to the text. You now have a new heading under which you can paste your code snippet for importing or exporting data of that file type.

5.1 Importing Data

5.1.1 Dynamic CSV Import with GUESSINGROWS

```
proc import datafile = "&input.\Bills.csv"
      out = Eww
      dbms = CSV replace;
      guessingrows=5000;
run;
```

5.1.2 Static CSV Import with Hardcoded Values

```
data WORK.EWW    ;
%let _EFIERR_ = 0;
      infile "&input.\Bills.csv" delimiter = ',' MISSOVER
      DSD lrecl=13106 firstobs=2 ;
            informat Monthly_Bills $9. ;
            informat Amount best32. ;
            format Monthly_Bills $9. ;
            format Amount best12. ;
      input
            Monthly_Bills  $
            Amount
      ;
      if _ERROR_ then call symputx('_EFIERR_',1);
run;
```

5.1.3 VALIDVARNAME option

```
options VALIDVARNAME=V7;

proc import OUT=coffee
      DATAFILE= "&input.\Read In Data.xlsx"
      DBMS=xlsx REPLACE;
      SHEET="Office Coffee Orders";
      GETNAMES=YES;
run;

proc freq data=coffee;
      table afternoon_favorites_;
run;
```

5.1.4 VALIDVARNAME option with a RENAME statement

```
options VALIDVARNAME=V7;
```

```
proc import OUT=coffee (rename=(Afternoon_Favorites_ =
Afternoon_Favs))
      DATAFILE= "&input.\Read In Data.xlsx"
      DBMS=xlsx REPLACE;
      SHEET="Office Coffee Orders";
      GETNAMES=YES;
run;

proc freq data=coffee;
      table Afternoon_Favs;
run;
```

5.1.5 Use invalid variable names in code

```
      options VALIDVARNAME=ANY;

proc import OUT=Coffee
      DATAFILE= "&input.\Read In Data.xlsx"
      DBMS=xlsx REPLACE;
      SHEET="Office Coffee Orders";
      GETNAMES=YES;
run;

proc freq data=coffee;
      table 'afternoon favorites!'n;
run;
```

5.1.6 Fixed-Width:

```
data Plants;
      INFILE "&input.\Plants_Ive_Killed.txt"
      firstobs=1
      obs=MAX
      LRECL=5000
      ENCODING="WLATIN1"
      NOPAD
      TRUNCOVER;
      INPUT
            @1    Plant_Name   $20.
            @21    Store    $27.
            @48    Date_Bought   yymmdd8.
            @56    Date_Died   yymmdd8.
;
format Date_Bought Date_Died   mmddyy10.;
run;
```

5.1.7 Tab Delimited:

```
proc import datafile="&input.\GroceryList.txt"
      out=food
      dbms=dlm
      replace;
      delimiter='09'x;
run;
```

5.1.8 SPSS:

```
proc import out=gss_imp
      datafile="&input.\GSS7218_R3.sav"
      dbms=SAV replace;
run;
```

5.2 Exporting Data

```
%let output=C:\Users\Kirby\Desktop\Book_Data\Output_Data;
libname output "&output.";
```

5.2.1 Excel:

```
proc export DATA=input.claim
      OUTFILE= "&output.\Claim_Data.xlsx"
      DBMS=XLSX REPLACE;
      SHEET="Medical";
run;
```

5.2.2 CSV:

```
proc export data=input.claim
      outfile="&output.\medical_claims.csv"
      dbms=csv replace;
run;
```

5.2.3 Fixed Width:

```
data _null_;
set input.Claim;
file "&output.\Claim_Fixed_Width.txt"
      LRECL=500
      ENCODING="WLATIN1"
      NOPAD;
put
      @1    PatientID  $9.
      @10   ClaimID $11.
      @21   Ctype 3.
      @24   Date  mmddyy10.
      @34 Claim_Desc $35.
      ;
run;
```

5.2.4 Tab Delimited:

```
proc export DATA=input.Claim
      OUTFILE= "&output.\Claim_Tab_Delimited.txt"
      DBMS=tab REPLACE;
run;
```

6 Viewing and Summarizing Data

Add your code snippets for viewing and summarizing data here.

6.1 Viewing Data

6.1.1 PROC Contents

```
proc contents DATA=table-name;
run;
```

```
proc contents data=input.coffee;
run;
```

6.1.2 Proc Print

```
proc print Data=input.GSS (OBS=10);
      VAR YEAR ID AGE MARITAL WRKSTAT SEX PARTYID;
      run;
```

```
proc print data=input.GSS;
      WHERE year=1988 and marital="MARRIED" and Age=25;
      VAR YEAR ID AGE MARITAL SEX CLASS;
run;
```

6.1.3 Proc Univariate

```
proc univariate Data=input.GSS;
      VAR Age;
run;
```

6.2 Summarizing Data

6.2.1 Proc Means

```
proc means data=input.GSS;
      VAR Age;
      CLASS Sex;
run;
```

6.2.2 Proc Freq

```
proc freq data=input.GSS;
      TABLES PartyID;
run;
```

6.2.3 Proc Tabulate

```
proc tabulate data=input.GSS;
      CLASS PartyID Sex;
      TABLE PartyID, Sex;
run;
proc tabulate data=input.GSS;
      CLASS PartyID Sex Race;
```

```
      TABLE Race, PartyID, Sex;
run;
proc tabulate data=input.GSS;
      CLASS PartyID Sex Race;
      TABLE PartyID, Sex*Race all;
run;
```

7 Data Transformations

Add your code snippets for data transformations here. Feel free to add additional sections based on the work that you do. Use the Heading Styles in the Styles pane to create new headings. To add these headings to your Table of Contents, right-click your Table of Contents, click **Update Field,** and then **Update Entire Table**.

7.1 Sorting and De-duplicating Data

```
proc sort data=input.Claim out=Claims_Sorted;
      by PatientID Date ClaimID;
run;

proc sort data=claims_sorted out=claims_nd nodupkey;
      by PatientID;
run;
```

7.2 Calculating New Variables

```
data Create_Vars (keep=First_name Last_Name FullName);
      Set input.Donations_JUL;
      Length FullName $50.;
      FullName=CATX(' ',First_Name, Last_Name);
run;
```

7.3 Filtering

```
Data And_Example;
      set input.GSS;
      where year=2006 and age=32 and degree in ("BACHELOR"
      "GRADUATE")and class^=" ";
      drop marital partyid;
run;
```

7.4 Conditional Logic

```
data Eligible_Sample;
      Set input.health_chart;
      IF Age >= 18 THEN Eligible=1;
      ELSE Eligible=0;
run;

data Travel_Needs;
      set input.Travel_Destinations;
      length Language $25. Currency $15.;

      if upcase(Country) = "USA" then do;
      Language="English";
      Currency="Dollar";
      end;

      else if upcase(Country) in("GERMANY" "AUSTRIA") then do;
      Language="German";
```

```
      Currency="Euro";
      end;

      else if upcase(Country) = "ITALY" then do;
      Language="Italian";
      Currency="Euro";
      end;

      else do;
      Language="Other";
      Currency="Other";
      end;
run;

proc sql;
      create table Travel as
      select *,
      case when Miles <= 1 then 'Walk'
      when Miles >= 600 then 'Fly'
      else 'Drive' end as Transportation
      from input.Travel_Destinations;
quit;
```

7.5 Manipulating Values

```
data Transformations;
      set input.Donations_JUL;
      Substr_Ex=substr(Address, 1, 4);
      Scan_Ex=scan(Address, 2, ' ');
      Upcase_Ex=upcase(First_Name);
      Catx_Ex=catx(' ',First_Name, Last_Name);
      Input_Ex=input(compress(Address,' ', 'a'), 8.);
      Sum_Ex=sum(Donation, Merchandise);
      Month_Ex=month(DOB);
      Mdy_Ex=mdy(7,31,2022); /*date of analysis*/
      Yrdif_Ex=round(yrdif(DOB, Mdy_Ex, 'AGE'), .01);
      Datdif_Ex=datdif(Donation_Date, Mdy_Ex, 'ACTUAL');
      Intnx_Ex=intnx('month', Donation_Date, 1, 'SAMEDAY');
      format Mdy_Ex Intnx_Ex mmddyy10.;
run;
```

7.6 Formatting

```
data Format;
      set input.Donations_JUL;
      format DOB mmddyy8. Donation_Date YYMMDD10.
      Donation dollar10.2 First_Name $1.
      Merchandise best8. Last_Name $upcase15.;
run;
```

7.6.1 User-Defined Formats

```
proc format;
```

```
      VALUE format-name
      Data-value-1 = 'Label 1'
      Data-value-2 = 'Label 2';
run;
```

7.6.1.1 Add text format to the numeric values of a variable

```
proc format;
      VALUE HEALTH_TXT
            1    = "Poor"
            2    = "Fair"
            3    = "Good"
            4    = "Very Good"
            5    = "Excellent";
run;
```

7.6.1.2 Add text format to a range of numeric values of a variable

```
proc format;
      VALUE INCOME_TXT
            LOW - < 25000 = "Low"
            25000 - < 75000 = "Middle"
            75000 - HIGH    = "High";
run;
```

7.6.1.3 Add text format to character values of a variable

```
proc format;
      VALUE $GENDER_TXT
            "M"    = "Male"
            "F"    = "Female"
            OTHER = "Other";
run;

data Survey_Formatted;
      SET input.Survey;
      FORMAT Health HEALTH_TXT. Income INCOME_TXT. Gender $GENDER_TXT.;
run;
proc print data=Survey_Formatted; run;
```

8 Combining and Aggregating Data

Add your code snippets for Combining and Aggregating data here.

8.1 Combing Data Using the DATA Step

8.1.1 Appending Data

```
data Donations_All;
      set input.Donations_JUL input.Donations_AUG;
run;
```

8.1.2 One-to-One Merge

The basic syntax for a DATA step merge is below.

```
data output-table-name;
   merge input-table-name1 input-table-name2;
   by column(s);
run;
```

```
proc sort data=input.Donations_JUL out=Sort_JUL; by First_Name
Last_Name DOB; run;
```

```
proc sort data=input.Donations_AUG out=Sort_AUG; by First_Name
Last_Name DOB; run;
```

```
data Donations_Merged (drop=Donation_Date Merchandise);
      merge Sort_JUL (rename=(Donation=Donation_JUL))
            Sort_AUG (rename=(Donation=Donation_AUG)) ;
      by First_Name Last_Name DOB;
      label Donation_JUL='July Donations' Donation_AUG='August
      Donations';
run;
```

8.1.3 One-to-Many Merge

```
proc sort data=input.Claim out=sort_claim; by Ctype; run;
```

```
proc sort data=input.Claim_Type out=sort_claim_type; by Ctype; run;
```

```
data Merged_Claims;
      merge sort_claim sort_claim_type;
      by Ctype;
proc sort; by PatientID Date;
run;
```

8.2 PROC SQL

8.2.1 PROC SQL Syntax

```
proc sql;
      CREATE TABLE output-table-name AS
      SELECT column(s)
      FROM input-table-name AS nickname
```

```
      <join type input-table-name2 AS nickname2>
      ON column(s)
      WHERE expression
      GROUP BY column(s)
      HAVING expression
      ORDER BY column(s);
quit;
```

8.2.2 PROC SQL Data Merge Example

8.2.2.1 Inner Join

```
proc sql;
     create table Demo_Exam as
      select a.StudentID, a.Race, a.Gender, b.Exam_Date, b.Subject,
      b.Score
      from input.Student_Demographic as a
      inner join input.Assessment as b
      on a.StudentID=b.StudentID;
quit;
```

8.2.2.2 Full Join

```
proc sql;
      create table Demo_Exam as
      select coalesce(a.StudentID, b.StudentID) as student_ID, a.Race,
      a.Gender, b.Exam_Date, b.Subject, b.Score
      from input.Student_Demographic as a
      full join input.Assessment as b
      on a.StudentID=b.StudentID;
quit;
```

```
proc sql;
      create table Demo_Exam as
      select *
      from input.Student_Demographic as a
      full join input.Assessment(rename=(StudentID=ID)) as b
      on a.StudentID=b.ID;
quit;
```

8.2.3 PROC SQL GROUP BY Example

```
proc sql;
      create table Avg_Assessment as
      select StudentID, avg(Score) as Avg_Score
      from input.Assessment
      group by StudentID
      order by Avg_Score desc;
      quit;
```

```
proc sql;
      create table Avg_Assessment as
      select StudentID,
      max(Exam_Date) as Last_Exam format=mmddyy10.,
```

```
      avg(Score) as Avg_Score
      from input.Assessment
      group by StudentID
      order by Avg_Score desc;
quit;
```

9 Comparing Data

Add your code snippets for comparing data here.

9.1 DATA Step Compare Merge

```
data Base_Only Compare_Only Both;
     merge Base (in=a) Compare (in=b);
     by ID;
     if a=1 and b=0 then output Base_Only;
     if b=1 and a=0 then output Compare_Only;
     if a=1 and b=1 then output Both;
run;
```

9.2 PROC COMPARE

```
proc compare base=dataset1 compare=dataset2 out=nonmatches
     outnoequal outall listvar;
run;
```

10 Reporting

Add your code snippets for data reporting here.

10.1 PROC PRINT

```
Proc print data=input.donations_JUL noobs label;
      var First_Name Last_Name Address Donation_Date;
      var Donation / style(data)={backgroundcolor=gold};
      label First_Name = 'Donor First Name'
             Last_Name = 'Donor Last Name'
             Donation_Date = 'Donation Date';
      format Donation_Date date9.;
      sum Donation;
run;

      proc format;

      value donorlevel
      low-<75='rose'
      75-<250='yellow'
      250-high='BILG';
run;


title 'Large Donors for Follow Up Marketing Campaign';
proc print data=input.donations_JUL noobs label;
      var First_Name Last_Name Address Donation_Date;
      var Donation / style(data)={backgroundcolor=donorlevel.};
      label First_Name = 'Donor First Name'
             Last_Name = 'Donor Last Name'
             Donation_Date = 'Donation Date'
             Donation = 'Donation Level';
      format Donation_Date date9.;
run;
title;
```

10.2 PROC REPORT

```
proc report DATA=input-table-name <option(s)>;
      COLUMN variable-1 < … variable-n>;
      DEFINE variable-1 / <options>;
      DEFINE variable-n / <options>;
      COMPUTE; ENDCOMP;
      BREAK;
      RBREAK;
run;

proc report DATA=input.demo_exam;
      COLUMN StudentID exam_date Race Gender subject score;
      DEFINE StudentID / DISPLAY 'Student ID';
      DEFINE exam_date / DISPLAY 'Exam Date';
      DEFINE subject / DISPLAY 'Subject';
```

```
      DEFINE score / DISPLAY 'Exam Score';
run;
```

10.2.1 PROC REPORT Spanning Headers

```
proc report DATA=input.demo_exam spanrows;
      COLUMN ('Student Exam Scores 2019' ('Student Info' StudentID Race
      Gender) ('Exam Info' exam_date subject score));
      DEFINE StudentID / ORDER descending 'Student ID';
      DEFINE Race / ORDER;
      DEFINE Gender / ORDER;
      DEFINE exam_date / DISPLAY 'Exam Date';
      DEFINE score / ANALYSIS mean 'Exam Score';
      BREAK after StudentID / summarize;
      RBREAK after / summarize;
run;
```

10.2.2 PROC REPORT COMPUTE BLOCK

```
proc report DATA=input.Demo_Exam (WHERE=(Score~=.)) ;
      COLUMN ('Student Exam Scores 2019' ('Student Info' StudentID Race
      Gender) ('Exam Info' N Score Score=Score2 Extra_Credit
      Final_Score));
      DEFINE StudentID / GROUP 'Student ID';
      DEFINE Race / GROUP;
      DEFINE Gender / GROUP;
      DEFINE N / 'Exam Count';
      DEFINE Score / ANALYSIS mean 'Exam Score Average';
      DEFINE Score2 / ANALYSIS max 'Exam Score Max';
      DEFINE Extra_Credit / computed 'Extra Credit';
      DEFINE Final_Score / computed 'Final Score';
      COMPUTE Extra_Credit;
            Extra_Credit=2;
      ENDCOMP;
      COMPUTE Final_Score;
            Final_Score= Score.mean+Extra_Credit;
      ENDCOMP;
run;
```

10.2.3 PROC REPORT ACROSS Option

```
proc report DATA=input.demo_exam;
      COLUMN ('Student Exam Scores 2019' StudentID Race Gender subject,
      (exam_date score));
      DEFINE StudentID / GROUP 'Student ID';
      DEFINE Race / GROUP;
      DEFINE Gender / GROUP;
      DEFINE subject / across;
      DEFINE exam_date / ANALYSIS 'Exam Date';
      DEFINE score / ANALYSIS mean 'Exam Score';
run;
```

10.3 Output Delivery System

10.3.1 ODS Destinations

```
ODS destination <destination option(s)>;
<SAS code to generate output for destination>;
ODS destination close;
```

10.3.2 ODS Graphics

```
proc sgplot DATA=input-table <options>;
plot statement(s) / <options>;
<appearance statements>;
run;
```

10.3.2.1 Scatter Plot

```
proc sgplot data=input.health_chart;
      scatter y=weight x=height;
run;

proc sgplot data=input.Health_Chart;
      scatter y=Weight x=Height / datalabel=Patient_ID
      datalabelpos=top
      markerattrs=(symbol=DiamondFilled size=14px);
run;
```

10.3.2.2 Histogram

```
ods pdf file = "&output.\Histogram_Default.pdf";
proc sgplot data=input.GSS;
      histogram Age;
run;
ods pdf close;

ods pdf file = "&output.\Histogram_Fancy.pdf";
proc sgplot data=input.GSS;
      histogram Age/ showbins binstart=5 binwidth=10;
      yaxis values=(0 to 24 by 2);
run;
ods pdf close;
```

10.3.2.3 Bar Chart

```
ods rtf file = "&output.\BarChart_Default.rtf";
proc sgplot data=input.GSS;
      vbar Marital;
run;
ods rtf close;

ods rtf file = "&output.\BarChart_Fancy.rtf";
proc sgplot data=input.GSS;
      vbar Marital/ response=Age stat=mean
      fillattrs=(color=blue)
```

```
      datalabelattrs=(color=black size=15 weight=bold)
      barwidth=.5 dataskin=pressed;
      label Age='MEAN AGE' Marital='MARITAL STATUS';
run;
ods rtf close;
```

11 Add your own Section Heading (Style = Heading 1)

11.1 Add Section Subheading (Style = Heading 2)

11.1.1 Add Section Subheading 2 (Style = Heading 3)

Appendix A: Resources

Table 1: Resource Links

Resource	Link
SAS® 9.4 and SAS Viya® 3.5 Programming Documentation	https://go.documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/pgmsaswlcm/home.htm
SAS Colors	https://support.sas.com/content/dam/SAS/support/en/books/pro-template-made-easy-a-guide-for-sas-users/62007_Appendix.pdf
SAS Functions	https://go.documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/lefunctionsref/p1q8bq2v0o11n6n1gpij335fqpph.htm
SAS Library	https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/basess/n0a43pssblhvu0n1b51enwlu24n5.htm
SAS® Studio – Free Version	https://welcome.oda.sas.com/
SAS Style Attributes	https://go.documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/odsug/p1pt77toue3iyun0z4l9gth5as9f.htm
SAS Video Portal	https://video.sas.com/