



SAS[®] Programming 1: Essentials

Course Notes

SAS® Programming 1: Essentials Course Notes was developed by Stacey Syphus and Beth Hardin. Additional contributions were made by Bruce Dawless, Brian Gayle, Anita Hillhouse, Marty Hultgren, Mark Jordan, Eva-Maria Kegelmann, Gina Repole, Gemma Robson, Samantha Rowland, Allison Saito, Prem Shah, Charu Shankar, Kristin Snyder, Peter Styliadis, Su Chee Tay, and Kitty Tjaris. Instructional design, editing, and production support was provided by the Learning Design and Development team.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.

SAS® Programming 1: Essentials Course Notes

Copyright © 2020 SAS Institute Inc. Cary, NC, USA. All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

Book code E71640, course code LWPG1V2/PG1V2, prepared date 26Mar2020. LWPG1V2_001

ISBN 978-1-64295-937-6

Table of Contents

Lesson 1 Essentials	1-1
1.1 The SAS Programming Process	1-3
Demonstration: SAS Programming Process.....	1-6
1.2 Using SAS Programming Tools	1-11
Demonstration: Submitting a SAS Program in SAS Enterprise Guide	1-13
Demonstration: Submitting a SAS Program in SAS Studio	1-15
Practice	1-18
1.3 Understanding SAS Syntax	1-29
Demonstration: Understanding SAS Program Syntax	1-36
Demonstration: Finding and Resolving Syntax Errors	1-39
1.4 Solutions	1-43
Solutions to Activities and Questions	1-43
Lesson 2 Accessing Data	2-1
2.1 Understanding SAS Data	2-3
2.2 Accessing Data through Libraries.....	2-13
Demonstration: Exploring Automatic SAS Libraries.....	2-21
Demonstration: Using a Library to Read Excel Files	2-25
2.3 Importing Data into SAS.....	2-28
Demonstration: Importing a Comma-Delimited (CSV) File	2-31
Practice	2-35
2.4 Solutions	2-38
Solutions to Practices.....	2-38
Solutions to Activities and Questions	2-39

Lesson 3	Exploring and Validating Data	3-1
3.1	Exploring Data	3-3
	Demonstration: Exploring Data with SAS Procedures	3-10
	Practice	3-14
3.2	Filtering Rows	3-18
	Demonstration: Filtering Rows with Basic Operators.....	3-22
	Demonstration: Filtering Rows Using Macro Variables	3-30
	Practice	3-33
3.3	Formatting Columns	3-37
	Demonstration: Formatting Data Values in Results	3-41
3.4	Sorting Data and Removing Duplicates	3-43
	Demonstration: Identifying and Removing Duplicate Values	3-49
	Practice	3-52
3.5	Solutions	3-54
	Solutions to Practices.....	3-54
	Solutions to Activities and Questions	3-57
Lesson 4	Preparing Data	4-1
4.1	Reading and Filtering Data	4-3
	Practice	4-12
4.2	Computing New Columns	4-14
	Demonstration: Using Expressions to Create New Columns	4-16
	Demonstration: Using Character Functions	4-22
	Demonstration: Using Date Functions	4-26
	Practice	4-28
4.3	Conditional Processing	4-30
	Demonstration: Conditional Processing with IF-THEN	4-31
	Demonstration: Processing Multiple Statements with IF-THEN/DO	4-42
	Practice	4-46

4.4 Solutions	4-48
Solutions to Practices.....	4-48
Solutions to Activities and Questions	4-51
Lesson 5 Analyzing and Reporting on Data	5-1
5.1 Enhancing Reports with Titles, Footnotes, and Labels	5-3
Demonstration: Enhancing Reports	5-9
5.2 Creating Frequency Reports.....	5-12
Demonstration: Creating Frequency Reports and Graphs	5-14
Demonstration: Creating Two-Way Frequency Reports	5-17
Practice	5-19
5.3 Creating Summary Statistics Reports	5-23
Demonstration: Creating Summary Statistics Reports	5-24
Practice	5-30
5.4 Solutions	5-33
Solutions to Practices.....	5-33
Solutions to Activities and Questions	5-36
Lesson 6 Exporting Results.....	6-1
6.1 Exporting Data	6-3
Demonstration: Exporting Data to an Excel Workbook	6-8
6.2 Exporting Reports.....	6-11
Demonstration: Exporting Results to Excel	6-15
Demonstration: Exporting Results to PDF.....	6-20
Practice	6-23
6.3 Solutions	6-27
Solutions to Practices.....	6-27
Solutions to Activities and Questions	6-29

Lesson 7 Using SQL in SAS®	7-1
7.1 Using Structured Query Language (SQL) in SAS	7-3
Demonstration: Reading and Filtering Data with SQL	7-9
7.2 Joining Tables Using SQL in SAS.....	7-13
Demonstration: Joining Tables with PROC SQL	7-16
7.3 Solutions	7-23
Solutions to Activities and Questions	7-23

To learn more...



For information about other courses in the curriculum, contact the SAS Education Division at 1-800-333-7660, or send e-mail to training@sas.com. You can also find this information on the web at <http://support.sas.com/training/> as well as in the Training Course Catalog.

For a list of SAS books (including e-books) that relate to the topics covered in this course notes, visit <https://www.sas.com/sas/books.html> or call 1-800-727-0025. US customers receive free shipping to US addresses.

Lesson 1 Essentials

1.1	The SAS Programming Process.....	1-3
	Demonstration: SAS Programming Process.....	1-6
1.2	Using SAS Programming Tools.....	1-11
	Demonstration: Submitting a SAS Program in SAS Enterprise Guide.....	1-13
	Demonstration: Submitting a SAS Program in SAS Studio	1-15
	Practice.....	1-18
1.3	Understanding SAS Syntax.....	1-29
	Demonstration: Understanding SAS Program Syntax.....	1-36
	Demonstration: Finding and Resolving Syntax Errors.....	1-39
1.4	Solutions	1-43
	Solutions to Activities and Questions.....	1-43

1.1 The SAS Programming Process

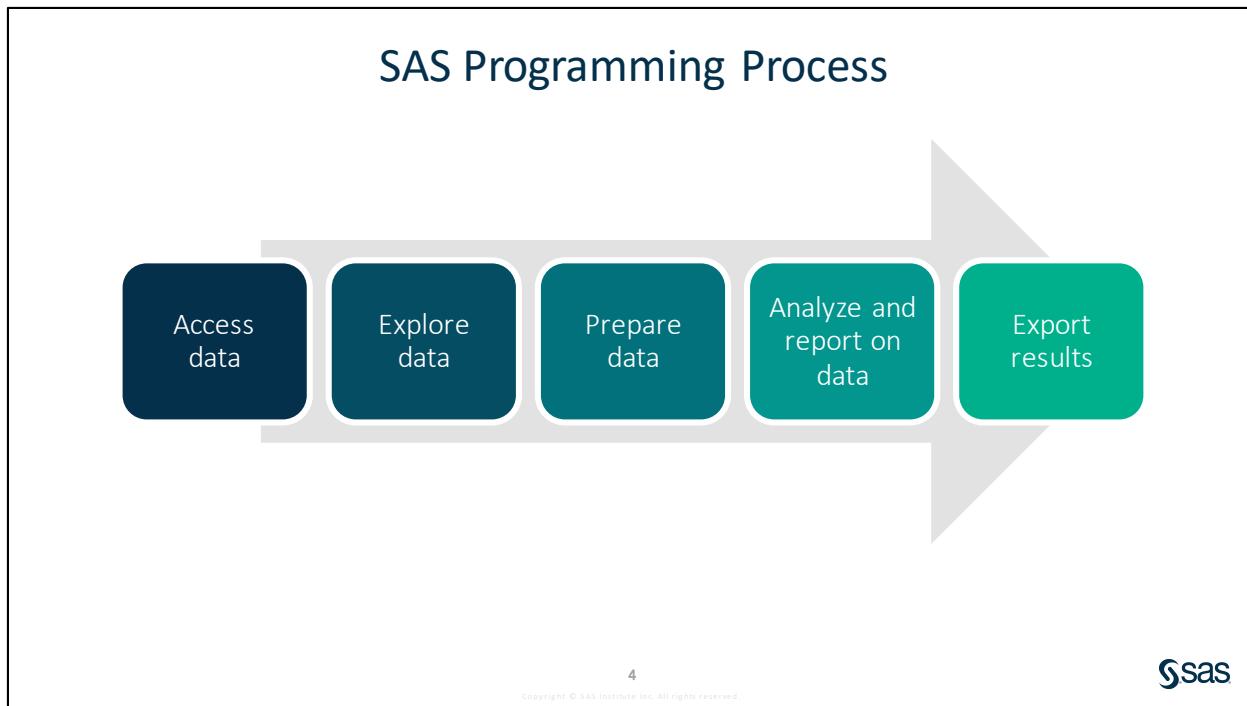
SAS Programming Language

The diagram illustrates the SAS Programming Language process. It features a central figure of a person's head with a lightbulb above it, symbolizing ideas or analysis. To the left, there are icons representing data input: a calendar, a grid, and a calculator. To the right, there are icons representing output: a bar chart and a line graph. Above the person's head is a lightbulb, symbolizing an idea or insight. To the right of the person is a computer monitor displaying the SAS logo and the tagline "THE POWER TO KNOW." To the far right is a mathematical formula $f(\cdot)$. The entire diagram is contained within a box with a thin black border.

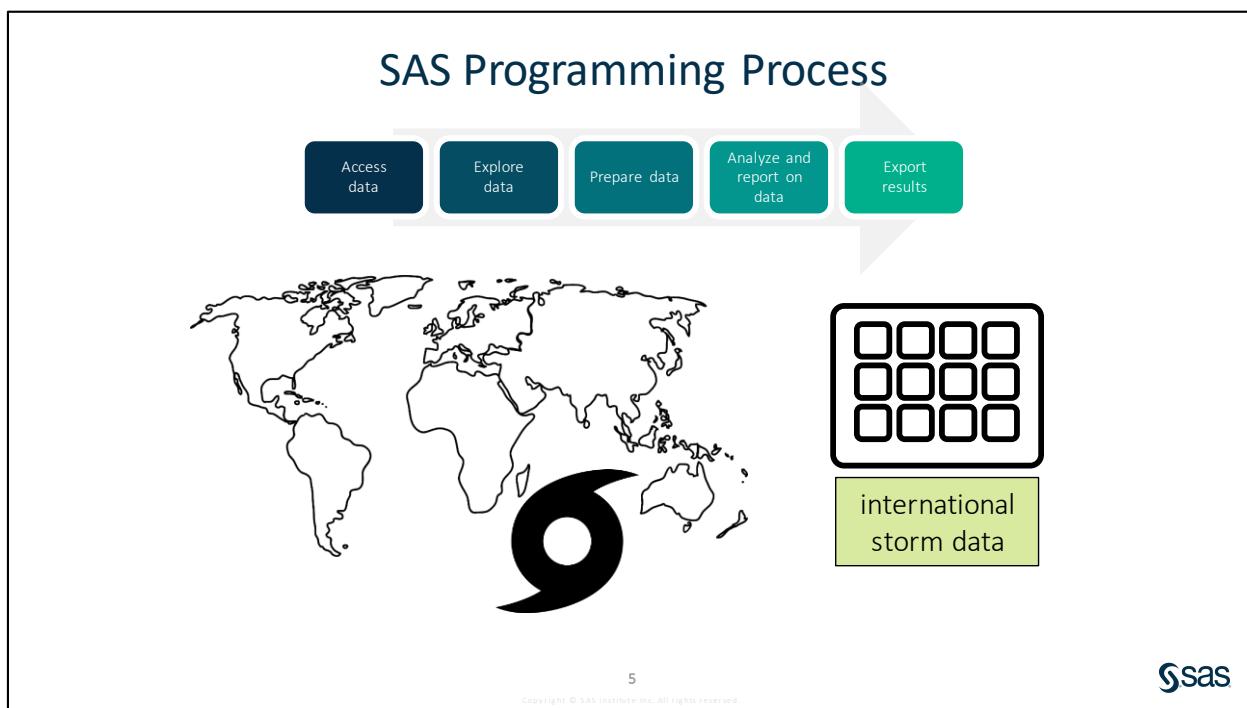
3
Copyright © SAS Institute Inc. All rights reserved.

 sas

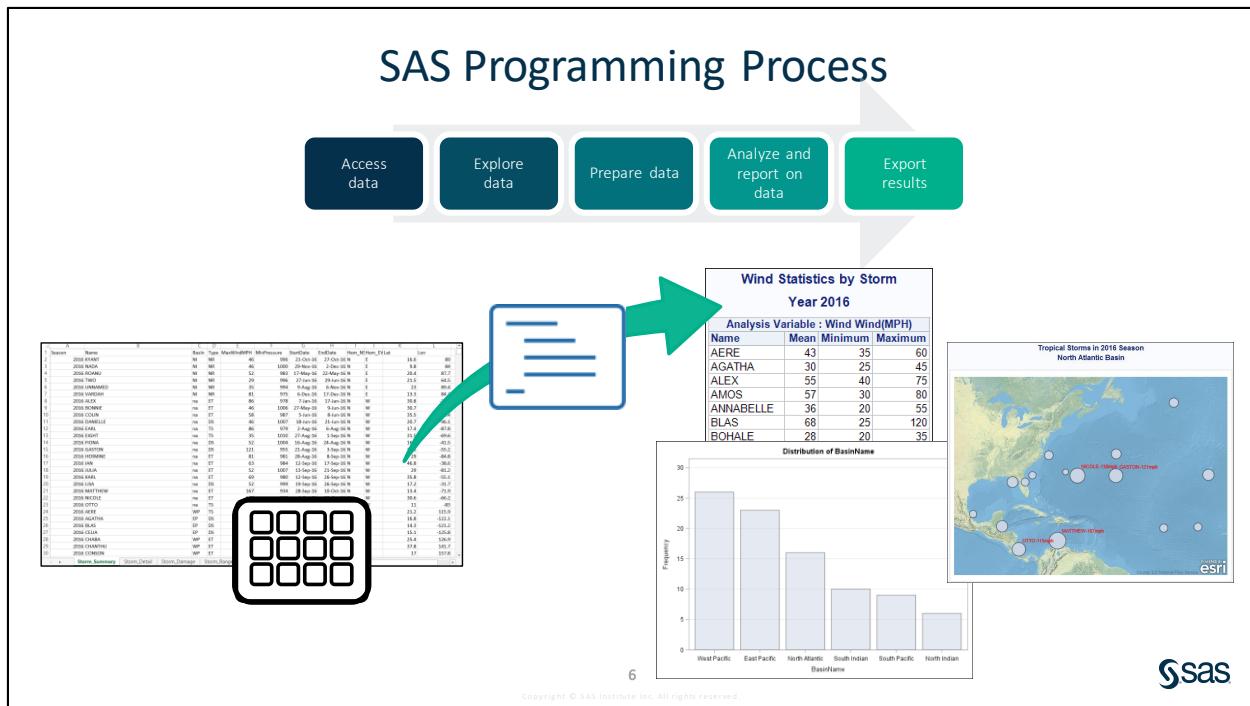
It is impossible to understand data without using tools that help you derive meaning from numbers and text. SAS offers a huge collection of tools and solutions to handle all your data needs. At the core of all that SAS offers is the SAS programming language. Regardless of the SAS suite of tools that you licensed, the Base SAS programming language is included. This course teaches you how to write SAS code to handle the most common data processing tasks.



As you go through the process of making data meaningful and actionable, you will likely follow these basic steps: access, explore, prepare, analyze and report, and export. SAS has programming tools for each of these steps in the process. You follow this process as you learn the fundamentals of the SAS programming language.



In this class, we analyze mainly international storm data, which is real data about storms such as hurricanes, typhoons, and cyclones that has been collected since 1980. This data is stored in a variety of formats, and the first thing you learn to do is write a SAS program to access the data.



As you continue through the programming process, you learn to write SAS programs that turn this data into informative reports, tables, and graphics.



SAS Programming Process

Scenario

Examine the international storm data that is used in course demonstrations. Open and run a SAS program that follows the SAS programming process. The code included in the program is covered throughout this course.

Files

- **p101d01.sas**
- **Storm.xlsx** – a Microsoft Excel workbook containing detail and summary data about international storms

Demo (for Instructors Only)

Note: The intent of this demo is not to study the specific syntax in the program. The purpose is to see a complete program that addresses the steps of the SAS programming process and view the results created by SAS. The details of the syntax in the program are discussed in subsequent lessons.

1. Start SAS Studio or SAS Enterprise Guide. Open the **cre8data.sas** program from the course files folder. If necessary, change **s:/workshop** to the path of your course files folder. Run the program and verify that a report listing 22 SAS tables is created.
2. Start Excel and open **Storm.xlsx** from the **data** folder in the course files. Examine each worksheet:
 - a. **Storm_Summary** contains one row per storm between 1980 and 2016. Wind speeds are measured in miles per hour (MPH).
Note: Refer to the **Basin_Codes**, **SubBasin_Codes**, and **Type_Codes** worksheets for the full names of the codes in the respective fields.
 - b. **Storm_Detail** contains one measurement for every six hours of a storm. Wind speeds are measured in knots.
 - c. **Storm_Damage** includes a description and damage estimates (adjusted for inflation) for storms in the US with damages greater than one billion dollars.
 - d. **Storm_Range** contains one row per storm with a minimum of four wind measurements. The top four wind measurements are in columns **Wind1** through **Wind4**.
 - e. **Storm_2017** contains one row per storm for 2017.
 - f. **Basin_Codes**, **SubBasin_Codes**, and **Type_Codes** are lookup tables with codes and descriptive values.
3. Close the **Storm.xlsx** file.
4. In either SAS Studio or SAS Enterprise Guide, open **p101d01.sas** from the **demos** folder.
5. Find the section labeled **Section 1: Access Data**. Highlight the code in Section 1 and run the selected portion of the program. Click **Run** or press the F3 key. Examine the Output Data tab to view the imported table.

The SAS code in this section does the following:

- a. sets system options

- b. establishes the location of the course files and generated results
 - c. connects to the Excel **Storm.xlsx** data so that it can be used in the program
 - d. creates a temporary copy of the **Storm_Damage** Excel spreadsheet as a SAS table
6. Click the **Code** tab. Highlight the code in Section 2 and press F3 to run the selected code. Examine the Results tab to view the reports.

The SAS code in this section does the following:

- a. creates a frequency report to examine the unique values of the **Basin** and **Type** columns
- b. calculates summary statistics for the **MaxWindMPH** and **MinPressure** columns
- c. prints the first five rows from the **storm_damage** table that was imported from Excel

7. Click the **Code** tab. Highlight the code in Section 3 and press F3 to run the selected code. Examine the Output Data tab to view the generated tables.

The SAS code in this section does the following:

- a. adds additional storm data from 2017
- b. cleans data by correcting case differences and assigning descriptive values to coded values
- c. creates additional new columns with numeric calculations and character manipulations
- d. joins tables to combine columns from two tables

8. Click the **Code** tab. Highlight the code in Section 4 and press F3 to run the selected code. Examine the Results tab to view the reports.

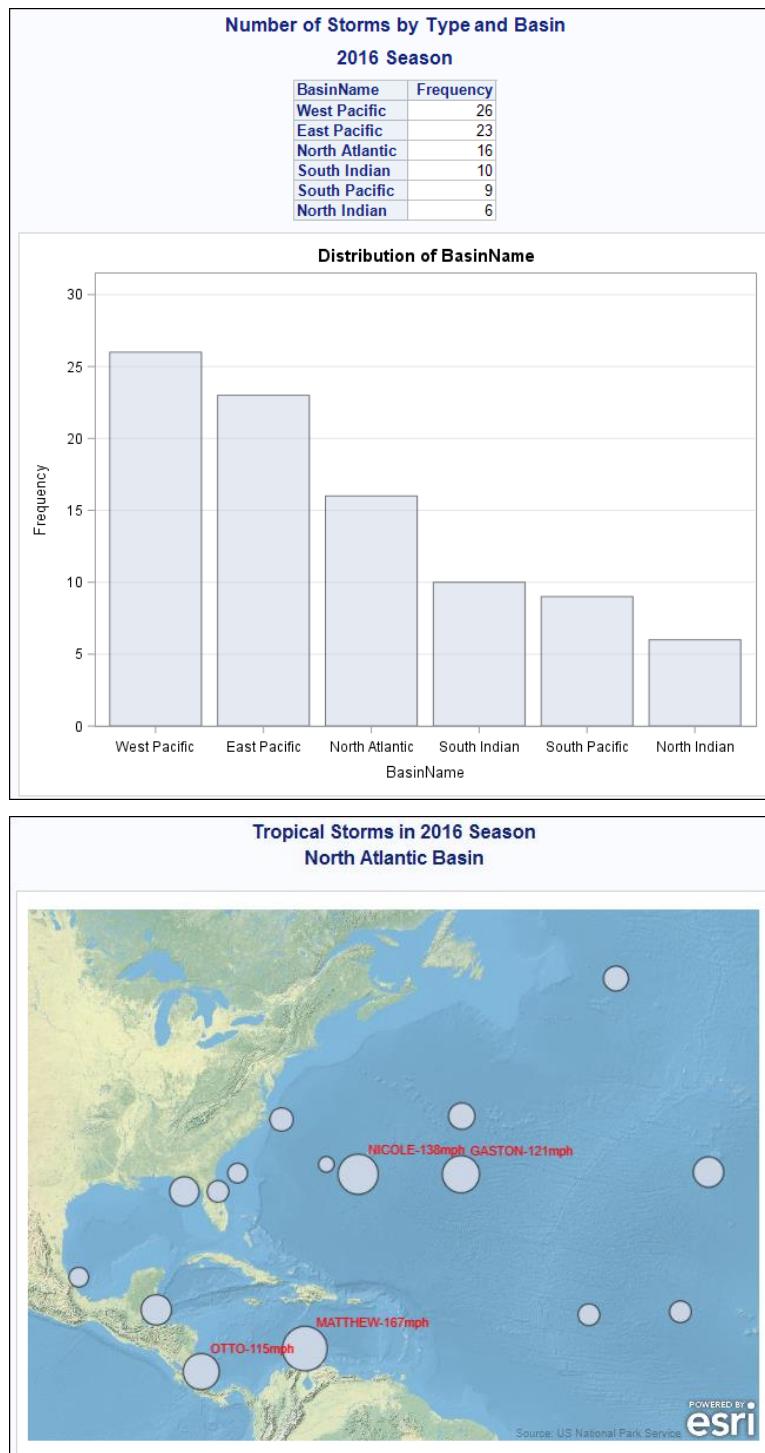
The SAS code in this section does the following:

- a. designates values to subset the data for the report as **2016** and **NA** (North Atlantic basin)
- b. generates default reports as well as an Excel workbook with the reports
- c. creates a frequency report for **BasinName** for the selected year
- d. creates a summary statistics report and a table that include statistics for storms in the selected year
- e. creates a map of storms for the selected year and basin

9. Open the **Storm_Report2016.xlsx** file in the **output** folder in the course files and view the results that were created in Excel.

- a. SAS Studio: Select **Files and Folders**, navigate to the **output** folder, and select the **Storm_Report2016.xlsx** file. Click **Download**  and open the file when you are prompted in your browser.

- b. Enterprise Guide: Click the **Results** tab and double-click the Excel file to open the new file. You can also right-click on the file and select **Open**.



End of Demonstration

Data Used in This Course

The diagram illustrates the data used in the course. It features four main data sources represented by icons of data grids:

- international storm and weather data**: Represented by a grid icon with a green label below it.
- US National Park data**: Represented by a grid icon with a green label below it.
- Europe tourism and trade data**: Represented by a grid icon with a green label below it.
- SAS sample tables**: Represented by three separate grid icons labeled **cars**, **class**, and **shoes**, each with a green label below it. These three are connected to the first three data sources by lines.

9
Copyright © SAS Institute Inc. All rights reserved.



- The detailed international storm data can be found at <https://www.ncdc.noaa.gov/ibtracs/index.php?name=wmo-data> as part of the International Best Track Archive for Climate Stewardship (IBTrACS). The data has been summarized and cleansed to use in this course.
- The US National Park data can be found at <https://irma.nps.gov/Stats/Reports/National>. The data has been summarized and cleansed to use in this course.
- The Europe tourism data can be found at <http://ec.europa.eu/eurostat/data/database>. The data has been summarized and cleansed to use in this course.
- SAS sample tables are provided in the **Sashelp** library. See <https://support.sas.com/documentation/tools/sashelpug.pdf> for documentation about the available tables.

Practicing in This Course

Demonstration	Performed by your instructor as an example for you to observe
Activity	Short practice opportunities for you to perform in SAS, either independently or with the guidance of your instructor
Practice	Extended practice opportunities for you to work on independently
Case Study	A comprehensive practice opportunity at the end of the class

Choosing a Practice Level

Level 1	Solve basic problems with step-by-step guidance
Level 2	Solve intermediate problems with defined goals
Challenge	Solve complex problems independently with SAS Help and documentation resources

Choose one practice to do in class based on your interest and skill level.



1.2 Using SAS Programming Tools

SAS Programming Interfaces

SAS Studio

SAS Enterprise Guide

SAS windowing environment

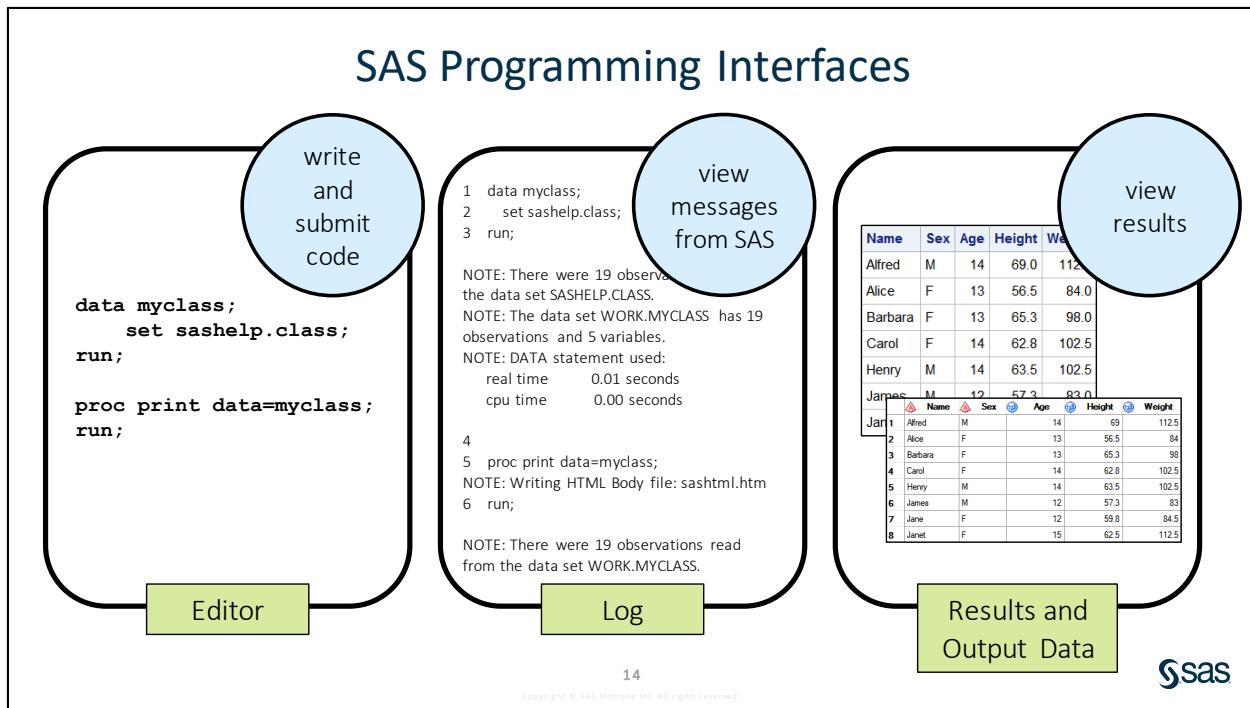
All these interfaces have the basic tools that you need for programming.

Sas

SAS provides several programming interfaces that can be used to interactively write and submit code.

- SAS Studio – a web-based interface to SAS that you can use on any computer. SAS Studio is the interface that is used in SAS OnDemand for Academics. SAS OnDemand for Academics is cloud-based software. For more information, visit https://www.sas.com/en_us/learn/academic-programs/software.html.
- SAS Enterprise Guide – a Windows client application that runs on your PC and accesses SAS on a local or remote server.
- SAS windowing environment – a legacy interface that is part of SAS.

Note: This course uses SAS Enterprise Guide and SAS Studio because these are the SAS interfaces that have the most modern programming tools.



To program, you need some basics: an editor to write and submit code, a way to read messages related to the code that you submit (this is called the *log* in SAS), and a way to view the reports and data that your programs create. Although they look different and are organized differently, all SAS interfaces have these interactive programming tools. In addition, SAS Studio and Enterprise Guide have an editor that is smart about SAS code, with features such as code completion and syntax coloring.

Programs can also be submitted to the operating environment behind the scenes. This is referred to as *batch processing* or *background submit*. The log and results are saved by default as separate files in the same location as the SAS program. Background submission is often used for programs that run regular jobs on a routine basis. These programs have typically been tested and can run unattended.

SAS Studio also enables you to submit programs by right-clicking a .sas file in the Navigation pane and selecting **Background Submit**. You can view the status of background programs and access the associated log and results files by clicking the **More application options** icon and selecting **Background Job Status**.



Submitting a SAS Program in SAS Enterprise Guide

Scenario

Write and submit a simple SAS program in SAS Enterprise Guide and examine the log and results.

Files

- **sashelp.class** – a sample table provided by SAS that includes information about 19 students

Notes

- Programs can be submitted by clicking **Run** or pressing the F3 key.
- A program generates a log. Depending on the code, a program might also generate results and output data.
- To run a subset of a program, highlight the desired code. Then click **Run** or press F3.

Demo

1. View **Sashelp** sample tables.

Note: **Sashelp** is a collection of sample data files provided by SAS that are useful for testing and practicing. This course references various data files in **Sashelp** to illustrate programming syntax.

- a. Open Enterprise Guide. On the Start Page, click **Create a new program**.

Note: In Enterprise Guide, your work can be organized in projects. To do so, select **Create a new project**. As you open tables and programs or create new programs, you will notice shortcuts added to your project in the Project pane. The project can be saved by selecting **File** ⇒ **Save Project**.

- b. In the Servers pane in the lower left corner, expand **Servers** ⇒ **Local** ⇒ **Libraries** ⇒ **SASHELP**.
 - c. Double-click the **CLASS** table to open and view the data. You do not need to close the table.
2. Write and submit a program in SAS Enterprise Guide.

- a. Type or copy and paste the program below on the Program tab and click **Run**.

Note: If the Program tab is not open, select **File** ⇒ **New** ⇒ **Program**, or click **Create a new item**  on the toolbar and select **Program**.

Note: If you copy and paste the program, click the **Format code** button  to improve the program spacing.

```
data myclass;
  set sashelp.class;
run;

proc print data=myclass;
run;
```

-
- b. Click the **Log** tab and toggle on the **Notes** button (if necessary). The log includes the program and messages that are returned from SAS. The Log Summary is displayed by default at the top of the window. You can click any of the messages in the Log Summary to find the message in the log.

Note: If the Log Summary is closed, click the drop-down arrow to the right of the Errors, Warnings, and Notes tab to expand the Log Summary.

- c. Click the **Output Data** and **Results** tabs to examine the output.
- d. Return to the Code tab. Highlight the PROC PRINT and RUN statements and click **Run** or press F3.

Note: In this course, you often need to run only a portion of a SAS program.

- e. Frequently, it is helpful to view multiple tabs at the same time. For example, you might want to view a program and the results, or possibly compare two tables. By default, SAS Enterprise Guide separates a program's tabs. To view more tabs at the same time, right-click the tab that you would like to view and select either **Float**, **New vertical tab group**, or **New horizontal tab group**. You can also drag and drop tabs outside of Enterprise Guide in any location that you prefer.

Note: To return a tab to the original location, right-click the top bar of the tab and select **Docked as tabbed document** for a Float window, or select **Move to previous tab group** for a vertical or horizontal tabbed group. You can also set defaults for program sub-tabs by going to **View** \Rightarrow **Program tab presets** and selecting the preset of your choice.

End of Demonstration



Submitting a SAS Program in SAS Studio

Scenario

Write and submit a simple SAS program in SAS Studio and examine the log and results.

Files

- **sashelp.class** – a sample table provided by SAS that includes information about 19 students

Notes

- Programs can be submitted by clicking **Run** or pressing the F3 key.
- A program generates a log. Depending on the code, a program might also generate results and output data.
- To run a subset of a program, highlight the desired code and click **Run** or press F3.
- When you rerun a program, the existing log, results, and output data are replaced.

Demo

1. View **Sashelp** sample tables.

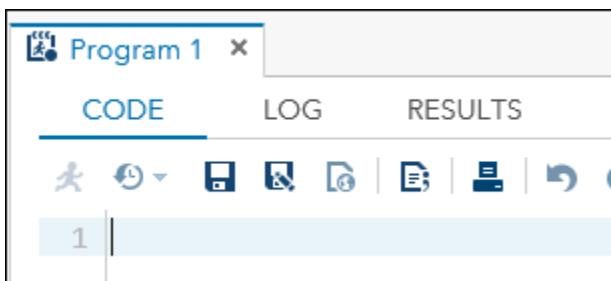
Note: **Sashelp** is a collection of sample data files provided by SAS that are useful for testing and practicing. This course references various data files in **Sashelp** to illustrate programming syntax.

- a. Open SAS Studio. In the navigation pane on the left side of the window, select **Libraries**. Expand **My Libraries** ⇒ **SASHELP**.
- b. Double-click the **CLASS** table to open and view the data. A panel to the left of the data lists the columns in the table. The Column panel can be collapsed by clicking the left-pointing arrow .
- c. Close the **SASHELP.CLASS** tab.

2. Write and submit a program in SAS Studio.

- a. A new program window labeled **Program 1** is open. Notice that there are tabs labeled **CODE**, **LOG**, and **RESULTS**.

Note: If you do not have a new program window, press F4 or click **New**  in the **Files and Folders** pane and select **SAS Program**.



- b. Type or copy and paste the program below on the CODE tab and click  Run.

Note: If you copy and paste the program, click the **Format code** button  to improve the program spacing.

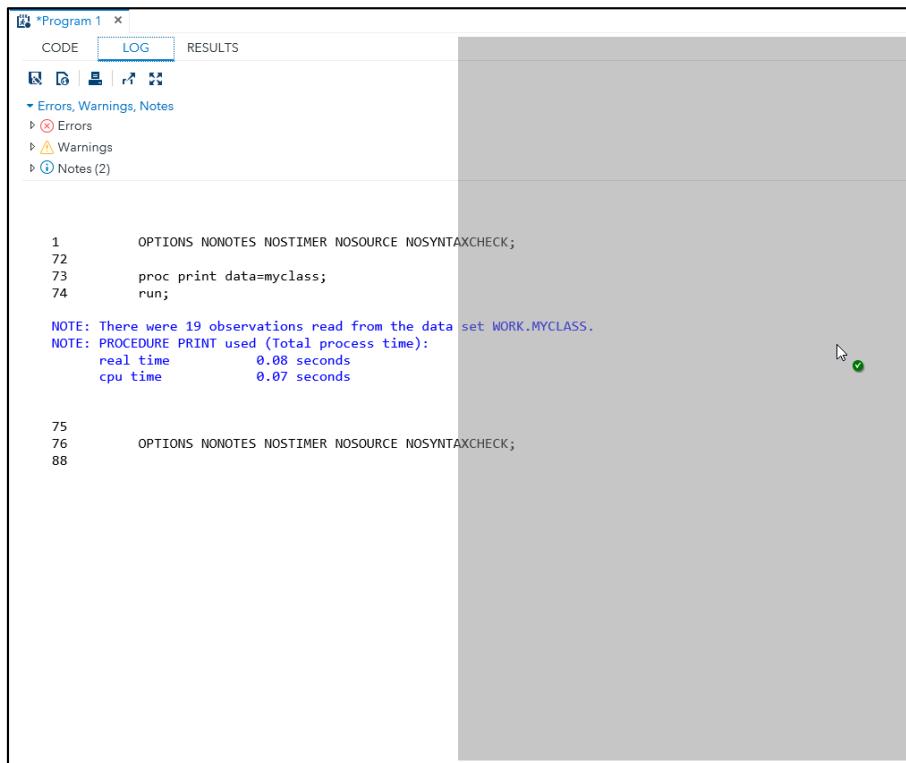
```
data myclass;
  set sashelp.class;
run;

proc print data=myclass;
run;
```

- c. Click the **LOG** tab. The log includes the program and messages returned from SAS. You can expand the **Errors**, **Warnings**, or **Notes** sections to see all messages in a summary list. You can click any of the messages to find the corresponding message in the log.
- d. Click the **RESULTS** and **OUTPUT DATA** tabs to examine the output.

Note: To optimize the display of the table on the OUTPUT DATA tab, right-click any of the column headings and select **Size grid columns to content**. To set this option for all tables, click **More application options**  and then click **Preferences**. Click the **Size grid columns to content** check box and click **Save**.

- e. Return to the CODE tab. Highlight the PROC PRINT and RUN statements and click **Run** or press F3. Confirm that the log and results were replaced.
- f. To view multiple tabs at the same time, click one of the tabs (CODE, LOG, OUTPUT DATA, or RESULTS) and drag it to the side or bottom of the work area until a highlighted region appears. To return to a single window, drag the separated tab back to the main tab area.



The screenshot shows the SAS IDE interface with the title bar "Program 1". The tabs at the top are "CODE", "LOG" (which is selected and highlighted in blue), and "RESULTS". On the left, there's a sidebar with icons for file operations and a section titled "Errors, Warnings, Notes" which is currently collapsed. Below the sidebar, the log content is displayed:

```

1      OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
2
3      proc print data=myclass;
4      run;

NOTE: There were 19 observations read from the data set WORK.MYCLASS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time      0.08 seconds
      cpu time      0.07 seconds

5
6      OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
7
8

```

End of Demonstration

1.01 Multiple Answer Question

Which SAS interface will you use in class?

- a. SAS Enterprise Guide
- b. SAS Studio



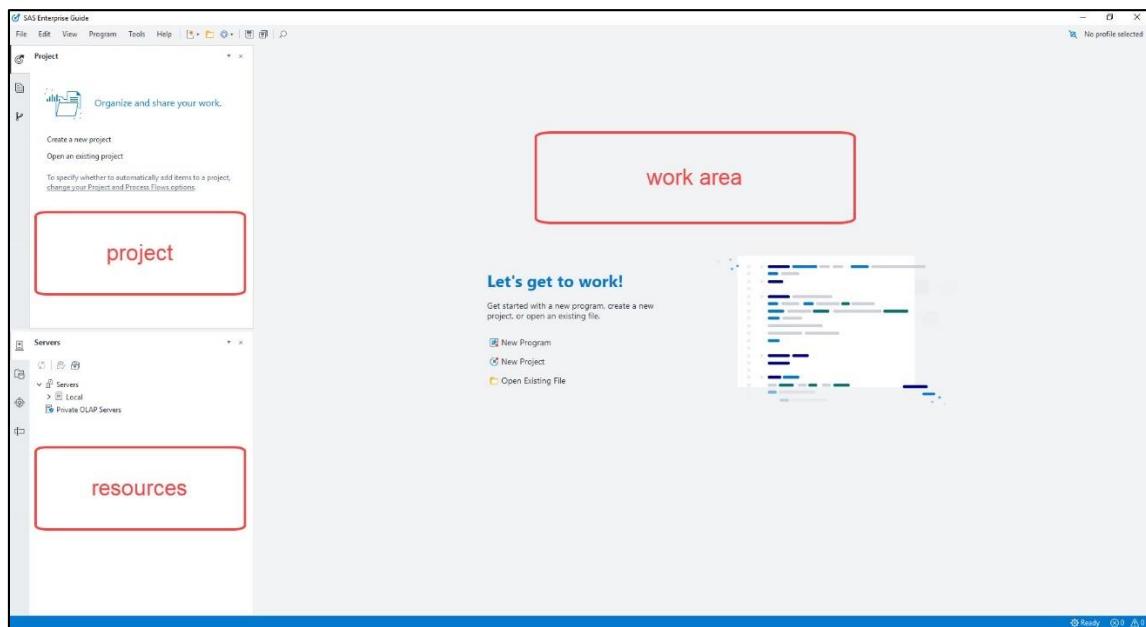
Practice

Note: Please choose either the SAS Enterprise Guide or SAS Studio practice to further explore your interface of choice.

Level 1 – SAS Enterprise Guide

1. Exploring the SAS Enterprise Guide Programming Windows

- Start SAS Enterprise Guide and close the Start Page. Enterprise Guide consists of a navigation area on the left and a work area on the right.



- Select **File** \Rightarrow **New** \Rightarrow **Program** (or click the **Create a new item** tool and select **Program**) to start writing a SAS program. On the Code tab, type or copy and paste the following code. This is a simple SAS program called a *DATA step*.

Note: If you copy and paste the program, click the **Format code** button to improve the program spacing.

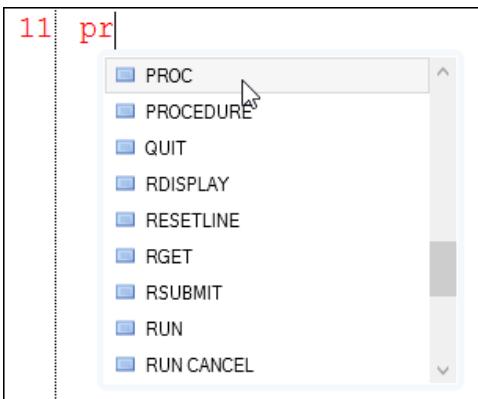
```

data work.shoes;
  set sashelp.shoes;
  NetSales=Sales>Returns;
run;

```

- Click **Run** or press F3 to submit the code. Examine the Log and Output Data tabs.
- Click the **Log** tab. Notice that there are additional statements included before and after the DATA step. This is called *wrapper code*, and it includes statements added by Enterprise Guide to set up the environment and results. To make the log easier to read, the wrapper code statements can be hidden. Select **Tools** \Rightarrow **Options** \Rightarrow **Results** \Rightarrow **General** and clear the **Show generated wrapper code in SAS log** check box. Click **OK**.
- Return to the Code tab and rerun the program. Examine the log.

- f. On the Code tab, add code to compute summary statistics. At the end of the program, begin by typing **pr**. Notice that a prompt appears with valid keywords. Press the Enter key or the spacebar to add the word **proc** to the program. Press the spacebar and type **me**. Press Enter again to add **means** to the program.



- g. Press the spacebar, use the prompts to select **data=work.shoes**, and press the spacebar again. Notice that the prompt lists all valid options. Type or select options in the window to complete the following statement:

```
proc means data=work.shoes mean sum maxdec=2;
```

Note: Autocomplete prompts can be modified or disabled by selecting **Program** ⇒ **Editor options** and then clicking the **Autocomplete** tab. On the tab, you can adjust the prompts.

- h. Complete the program by adding the highlighted statements below. Notice that after VAR and CLASS, the autocomplete prompt includes a list the columns from the **work.shoes** table.

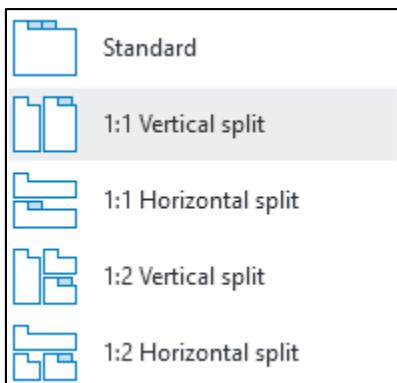
```
proc means data=work.shoes mean sum maxdec=2;
  var NetSales;
  class region;
run;
```

- i. Highlight the code from PROC MEANS through RUN, and select **Run** or press F3.

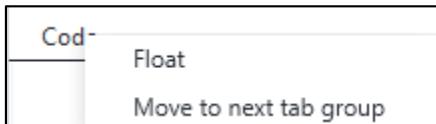
Note: The default output format in SAS Enterprise Guide is HTML.

The MEANS Procedure			
Analysis Variable : NetSales			
Region	N Obs	Mean	Sum
Africa	56	40508.95	2268501.00
Asia	14	32095.43	449336.00
Canada	37	111522.11	4126318.00
Central America/Caribbean	32	110339.22	3530855.00
Eastern Europe	31	74459.32	2308239.00
Middle East	24	226037.46	5424899.00
Pacific	45	49325.89	2219665.00
South America	54	43183.93	2331932.00
United States	40	132912.10	5316484.00
Western Europe	62	75858.79	4703245.00

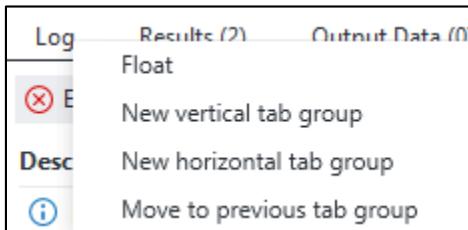
- j. By default, the tabs are a vertical split. To change the default layout view of the program tabs, go to **View** \Rightarrow **Program tab presets**. You can also right-click a tab and select **Float**, **New vertical tab group**, **New horizontal tab group**, or (if it is available) **Move to previous tab group**.



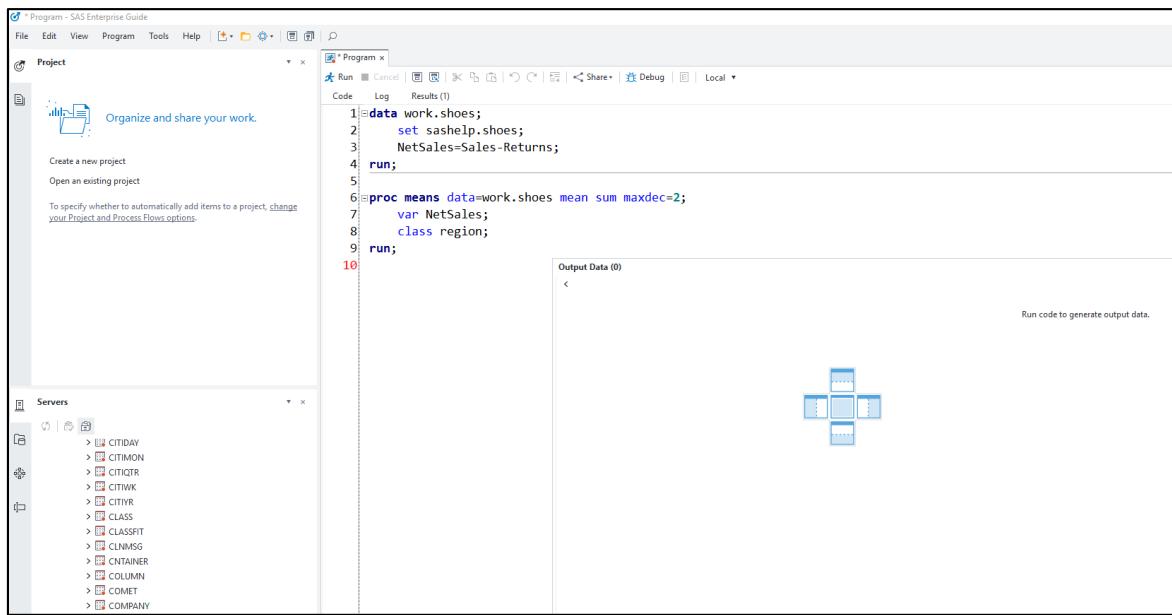
Note: Options for a single tab.



Note: Options for grouped tabs.

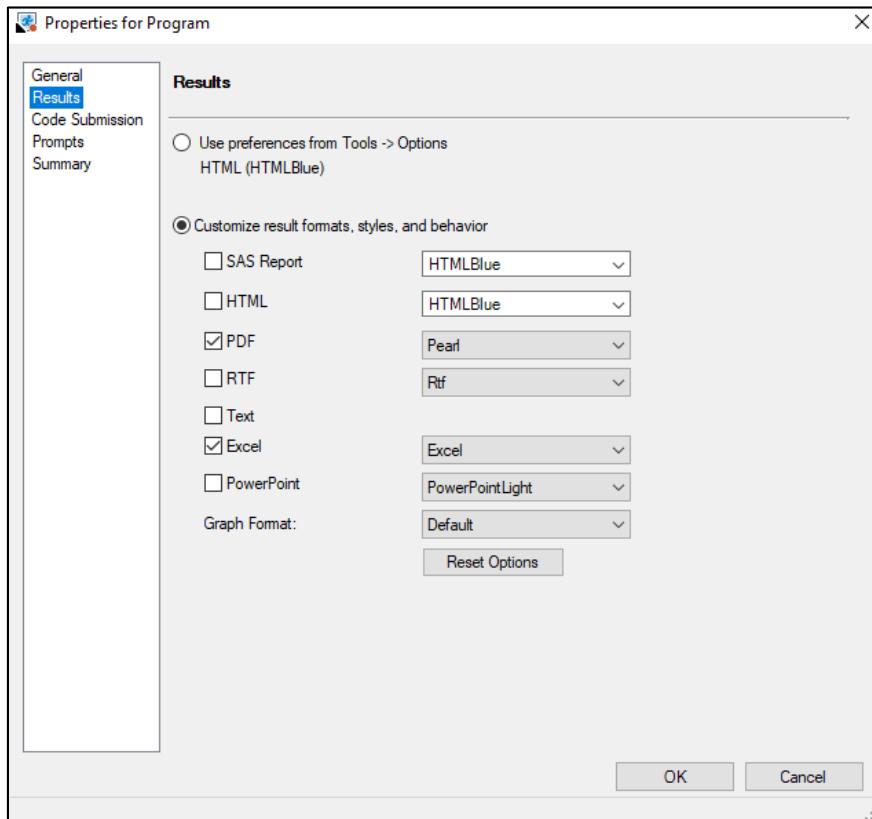


Note: You can also select a tab and drag it to a location of your choice.



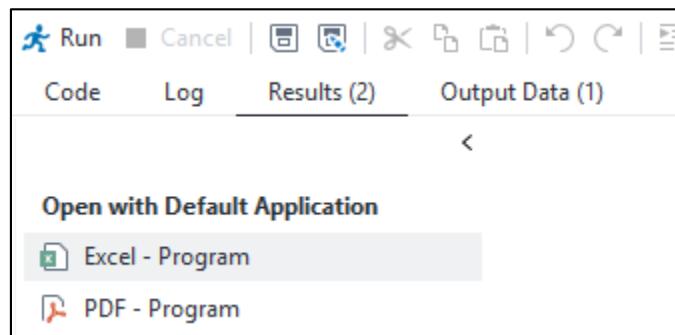
- To return to a single window for all program tabs, select **View** \Rightarrow **Program tab presets** \Rightarrow **Standard**.

- I. In addition to creating **HTML** output, you can create other output types. Click the **Code** tab and click the properties  icon. Select **Results** \Rightarrow **Customize result formats, styles, and behavior**. Clear any selected check boxes and then select the **PDF** and **Excel** check boxes. Click **OK**.



- m. Run the program again. An Excel file and a PDF file are created on the Results tab.

Note: PowerPoint, Excel, PDF, and RTF results must be viewed outside of Enterprise Guide. Double-click the Excel or PDF file to open it. You can also right-click the file and select **Open**.

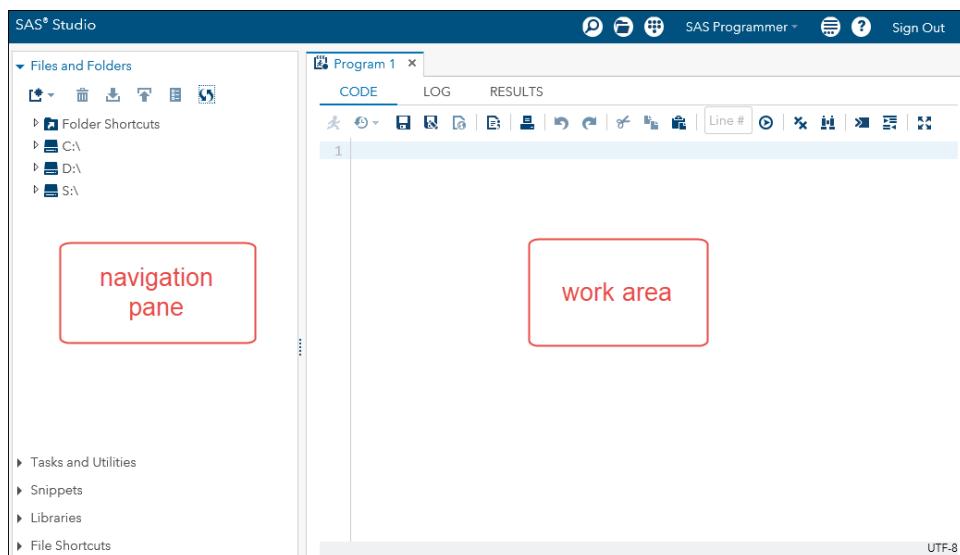


- n. To save the program, return to the Code tab and click the **Save "Program" As** icon . Navigate to the **output** folder in the course files. Enter **shoesprogram** in the **File name** field and click **Save**. The .sas file extension is automatically added to the file name.

Level 1 – SAS Studio

2. Exploring the SAS Studio Editor

- a. Start SAS Studio. The main window of SAS Studio consists of a navigation pane on the left and a work area on the right.



- b. Options are available in the banner area to customize your SAS Studio environment.

 Search	Search files and folders.
 Open	Open files from your files and shortcuts.
 New Options	New program, new import data, new query, close all tabs, and maximize view.
 SAS Programmer	SAS Studio includes two different perspectives: the SAS Programmer perspective and the Visual Programmer perspective. A <i>perspective</i> is a predetermined set of features that is customized to meet the needs of a specific type of user. This course is about programming in SAS, so you need to make sure that the SAS Programmer perspective is selected on the toolbar at the top of the application. You can find more information about both perspectives in <i>SAS Studio User's Guide</i> .
 More application options	More application options, including edit autoexec file, a view menu, preferences, tool options, background submission status, and reset SAS session.
 Help	A Help menu, including SAS Studio Help, SAS Product Documentation, and About SAS Studio.

- c. On the Program 1 tab, type or copy and paste the following code. This is a simple SAS program called a *DATA step*.

Note: If you copy and paste the program, click **Format code**  to improve the program spacing.

```
data work.shoes;
  set sashelp.shoes;
  NetSales=Sales-Returns;
run;
```

- d. Click **Run**  or press F3 to submit the code. Examine the LOG and OUTPUT DATA tabs. The RESULTS tab is empty because the program did not create a report.
- e. On the CODE tab, add code to compute summary statistics. At the end of the program, begin by typing **pr**. Notice that a prompt appears with valid keywords and syntax help. Press Enter to add the word **proc** to the program. Press the spacebar and type **me**, and press Enter again to add **means** to the program.

Note: The Autocomplete prompts also include a window with syntax Help and links to documentation and examples.



- f. Press the spacebar, use the prompt to select **data=**, and then type **work.shoes**. Press the spacebar and notice that the prompt lists all valid options. Type or select options in the window to complete the following statement:

```
proc means data=work.shoes mean sum;
```

- g. Autocomplete prompts can be disabled by clicking **More application options**  and then clicking **Preferences** \Rightarrow **Code and Log**. Clear the **Enable autocomplete** check box and click **Save**.
- h. Return to the CODE tab and press the spacebar after the SUM option and before the semicolon. Notice that a prompt does not appear. Type **MAXDEC=2** to round statistics to two decimal places.

Note: If autocomplete is turned off, you can temporarily toggle it on at any point by holding down the Ctrl key and pressing the spacebar to view the autocomplete prompt.

- i. Complete the program by adding the following statements:

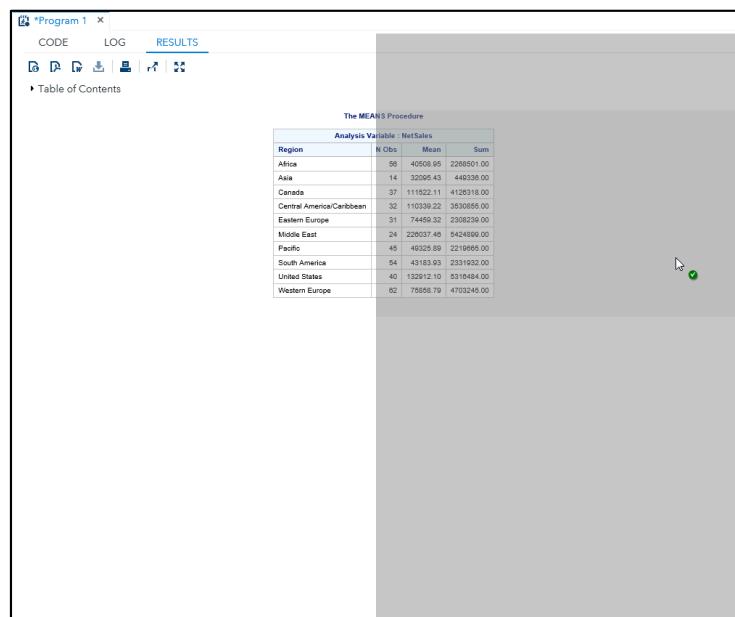
```
proc means data=work.shoes mean sum maxdec=2;
  var NetSales;
  class region;
run;
```

- j. Highlight the code from PROC MEANS through RUN and click **Run** or press F3 to run only the selected portion. Confirm the results.

Note: The default output format in SAS Studio is HTML.

The MEANS Procedure			
Analysis Variable : NetSales			
Region	N Obs	Mean	Sum
Africa	56	40508.95	2268501.00
Asia	14	32095.43	449336.00
Canada	37	111522.11	4126318.00
Central America/Caribbean	32	110339.22	3530855.00
Eastern Europe	31	74459.32	2308239.00
Middle East	24	226037.46	5424899.00
Pacific	45	49325.89	2219665.00
South America	54	43183.93	2331932.00
United States	40	132912.10	5316484.00
Western Europe	62	75858.79	4703245.00

- k. To view multiple tabs at the same time, click the **RESULTS** tab and drag it to the right side of the work area until a highlighted region appears. To return to a single window, drag the **RESULTS** tab back to the main tab area.

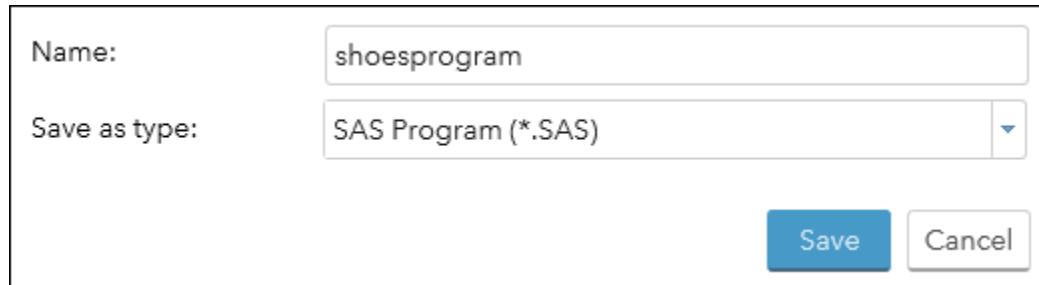


- l. On the RESULTS tab, click the HTML, PDF, or Word icon to open results in the corresponding file format. You are prompted to open or save the file in the browser.



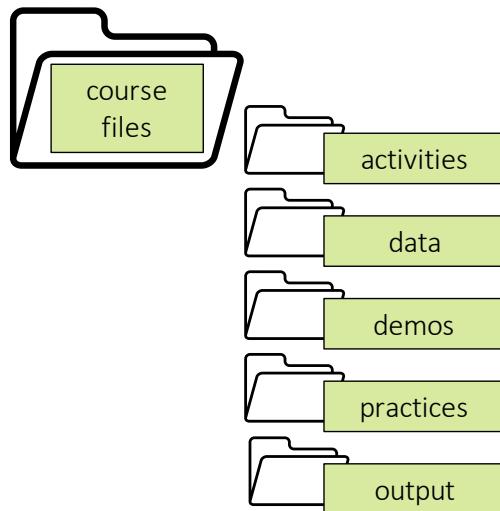
Note: Additional options for the output formats are available by clicking **More application options**  and selecting **Preferences**  **Results**.

- m. To save the program, return to the CODE tab and click the **Save As**  toolbar button. Navigate to the **output** folder in the course files. Enter **shoesprogram** in the **Name** field and click **Save**. The .sas file extension is automatically added to the file name.



End of Practices

Accessing the Course Files



Make a note of the location of your course files folder.

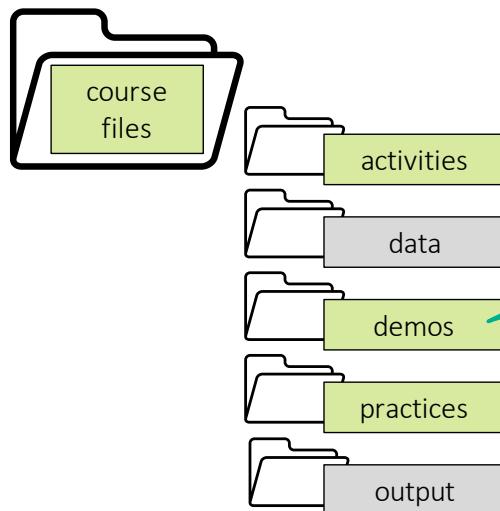


sas

18

Copyright © SAS Institute Inc. All rights reserved.

Accessing the Course Files



Programs in the activities, demos, and practices folders follow this naming convention.



p104d01.sas

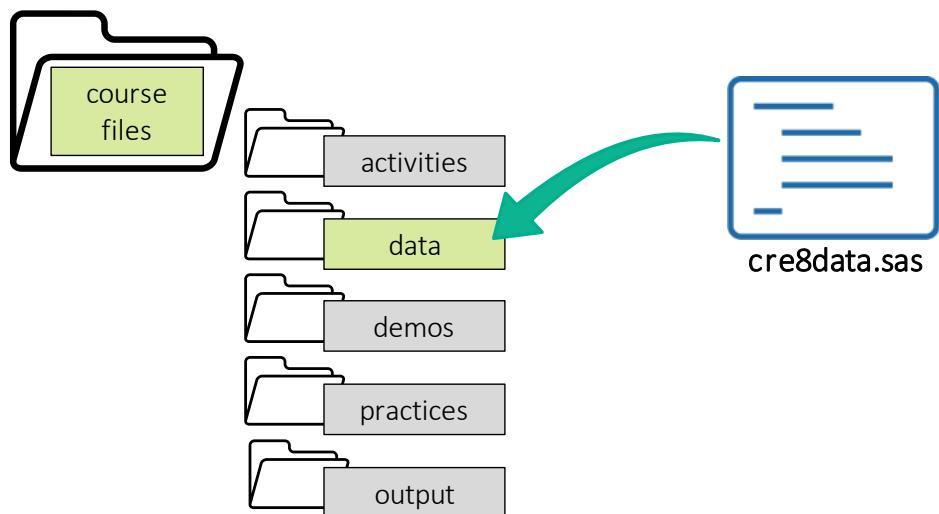
Programming 1, Lesson 4, demo 1

19

Copyright © SAS Institute Inc. All rights reserved.

sas

Creating the Course Data



20



Copyright © SAS Institute Inc. All rights reserved.

1.02 Activity (Required)

1. SAS Studio: In the Navigation pane, expand **Files and Folders** and then navigate to the course files folder.
SAS Enterprise Guide: In the Servers list, expand **Servers** ⇒ **Local** ⇒ **Files**, and then navigate to the course files folder.
2. Double-click the **cre8data.sas** file to open the program.
3. Find the %LET statement. As directed by your instructor, provide the path to your course files.
4. Run the program and verify that a report that lists 22 tables is created.

21



Copyright © SAS Institute Inc. All rights reserved.

1.3 Understanding SAS Syntax

SAS Program Structure

The diagram illustrates the structure of a SAS program. On the left, a vertical stack of three light blue rectangular boxes is labeled "step" at the top of each. Below the first two boxes is a colon followed by another colon, indicating continuation. This stack is enclosed in a rounded black border. At the bottom of this stack is a green rectangular box labeled "SAS program". To the right of this is a blue rectangular box containing SAS code:

```

data myclass;
  set sashelp.class;
  heightcm=height*2.54;
run;

proc print data=myclass;
run;

proc means data=myclass;
  var age heightcm;
run;

```

A thought bubble originates from a cartoon head icon at the bottom right and points to the text "A SAS program consists of a sequence of steps." located to the right of the code box. The SAS logo is in the bottom right corner.

24
Copyright © SAS Institute Inc. All rights reserved.

SAS Program Structure

The diagram shows two types of steps: "DATA step" represented by a teal icon with three overlapping rectangles, and "PROC step" represented by an orange icon with two interlocking gears. To the right is a blue rectangular box containing SAS code:

```

data myclass;
  set sashelp.class;
  heightcm=height*2.54;
run;

proc print data=myclass;
run;

proc means data=myclass;
  var age heightcm;
run;

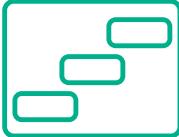
```

A thought bubble originates from a cartoon head icon at the bottom right and points to the text "A program can be any combination of DATA and PROC (procedure) steps" located to the right of the code box. The SAS logo is in the bottom right corner.

25
Copyright © SAS Institute Inc. All rights reserved.

SAS Program Structure

DATA step



```
data myclass;
  set sashelp.class;
  heightcm=height*2.54;
run;

proc print data=myclass;
run;

proc means data=myclass;
  var age heightcm;
run;
```

DATA steps typically read, process, or create data.



A DATA step can contain a variety of data manipulations, including filtering rows, computing new columns, and joining tables. In this program, the DATA step is creating a copy of an existing SAS table and adding a new column to convert height from inches to centimeters.

SAS Program Structure

PROC step



```
data myclass;
  set sashelp.class;
  heightcm=height*2.54;
run;

proc print data=myclass;
run;

proc means data=myclass;
  var age heightcm;
run;
```

PROC steps typically report, manage, or analyze data.



A PROC, or procedure, step typically processes a SAS data set. SAS has dozens of procedures that generate reports and graphs, manage data, or perform complex statistical analysis. This program has two PROC steps: PROC PRINT generates a list of all the rows and columns in the data, and PROC MEANS calculates basic summary statistics for **age** and **heightcm**.

SAS Program Structure

Steps begin with either DATA or PROC.

```

data myclass;
  set sashelp.class;
  heightcm=height*2.54;
run;

proc print data=myclass;
run;

proc means data=myclass;
  var age heightcm;
run;

```

Steps end with RUN. Some PROCs end with QUIT.

This program has three steps.

28 Copyright © SAS Institute Inc. All rights reserved.

sas

If a RUN or QUIT statement is not used at the end of a step, the beginning of a new step implies the end of the previous step. If a RUN or QUIT statement is not used at the end of the last step, SAS Studio and Enterprise Guide automatically submit a RUN and QUIT statement after the submitted code.

SAS Program Structure

step
statement;
statement;
statement;

step

step

SAS program

```

data myclass;
  set sashelp.class;
  heightcm=height*2.54;
run;

proc print data=myclass;
run;

proc means data=myclass;
  var age heightcm;
run;

```

A step is a sequence of SAS statements.

29 Copyright © SAS Institute Inc. All rights reserved.

sas

SAS Statement Syntax

```
data myclass;
  set sashelp.class;
  heightcm=height*2.54;
run;

proc print data=myclass;
run;

proc means data=myclass;
  var age heightcm;
run;
```

Most statements begin with a keyword, and all statements end with a semicolon.



30

Copyright © SAS Institute Inc. All rights reserved.

Most statements begin with an identifying keyword. In addition to DATA, PROC, and RUN statements, this program also includes SET and VAR statements. The one statement that does not begin with a keyword is the one that is creating the new column **heightcm**. The most important thing to remember here is that **all** statements end with a semicolon.

Global Statements

TITLE . . . ;

OPTIONS . . . ;

LIBNAME . . . ;

Global statements are typically outside of steps and do not need a RUN statement.



Copyright © SAS Institute Inc. All rights reserved.

In addition to DATA and PROC steps, a SAS program can also contain global statements. These statements can be outside DATA and PROC steps, and they typically define some option or setting for the SAS session. Global statements do not need a RUN statement after them.

1.03 Activity

Open **p101a03.sas** from the **activities** folder and perform the following tasks:

1. View the code. How many steps are in the program?
2. How many statements are in the PROC PRINT step?
3. How many global statements are in the program?
4. Run the program and view the log.
5. How many observations were read by the PROC PRINT step?

32

Copyright © SAS Institute Inc. All rights reserved.



SAS Program Syntax: Format

These are
the same
to SAS.

```
data myclass;set sashelp.class;run;
proc print data=myclass;run;
```

```
data myclass;
  set sashelp.class;
run;

proc print data=myclass;
run;
```

Formatting makes
your code easier
to read and
understand.



34

Copyright © SAS Institute Inc. All rights reserved.



These two programs have exactly the same code. Spacing does not matter to SAS, but it does matter to people reading your code. You can use spaces and extra lines to make your program easy to read and understand. There are also tools in your editor that format code for you. Click **Format code** on the toolbar or right-click in the program and select **Format code** to format a SAS program.

SAS Program Syntax: Case

```
data under13;
  set sashelp.class;
  where AGE<13;
  drop heIght Weight;
run;
```

Unquoted values can
be in any case.

SAS Program Syntax: Comments

```
/* students under 13 yo */
data under13;
  set sashelp.class;
  where Age<13;
  *drop Height Weight;
run;
```

comments out a single statement ending in a semicolon

comments out everything between /* and */

Comments are ignored when a program executes.



36

Another thing that you can do to make your code more understandable is to add comments. Any commented text is ignored when the program executes. Comments are also useful when you are testing code because you can suppress a portion of the code from execution.

To comment out a block of code using the /* */ technique in the SAS interfaces, you can highlight the code and then press Ctrl+/(forward slash).

- To uncomment a block of code in SAS Studio, highlight the block and then press Ctrl+/* again.
- To uncomment a block of code in SAS Enterprise Guide, highlight the block and then press Ctrl+Shift+/*.



Understanding SAS Program Syntax

Scenario

Examine program statements, improve program spacing, and add comments.

Files

- **p101d02.sas**
- **sashelp.cars** – a sample table provided by SAS that includes basic information about 428 cars

Syntax

```
/*comment*/
*comment;
```

Notes

- All statements end with a semicolon.
- Spacing does not matter in a SAS program.
- Values not enclosed in quotation marks can be lowercase, uppercase, or mixed case.
- Consistent program spacing is a good practice to make programs legible.
- Use the automatic spacing feature **Format code** to improve the spacing in a program.
- Comments can be added to prevent text in the program from executing.

Demo

1. Open the **p101d02.sas** program from the **demos** folder. Run the program. Does it run successfully?
2. Use the Format code feature to improve the program spacing. Use one of the following methods:
 - Click **Format code** .
 - Right-click in the program and select **Format code**.
3. Add the following text as a comment before the DATA statement: **Program created by <your-name>**

Note: Select the comment text and press Ctrl+/ to surround it with /* and */.

4. Comment out the first TITLE statement and the WHERE statement in PROC PRINT.
Run the code and verify that 428 rows are included in the results.

```

/*Program created by <name>*/
data mycars;
    set sashelp.cars;
    AvgMPG=mean(mpg_city, mpg_highway);
run;

*ttitle "Cars with Average MPG Over 35";

proc print data=mycars;
    var make model type avgmpg;
    *where AvgMPG > 35;
run;

title "Average MPG by Car Type";

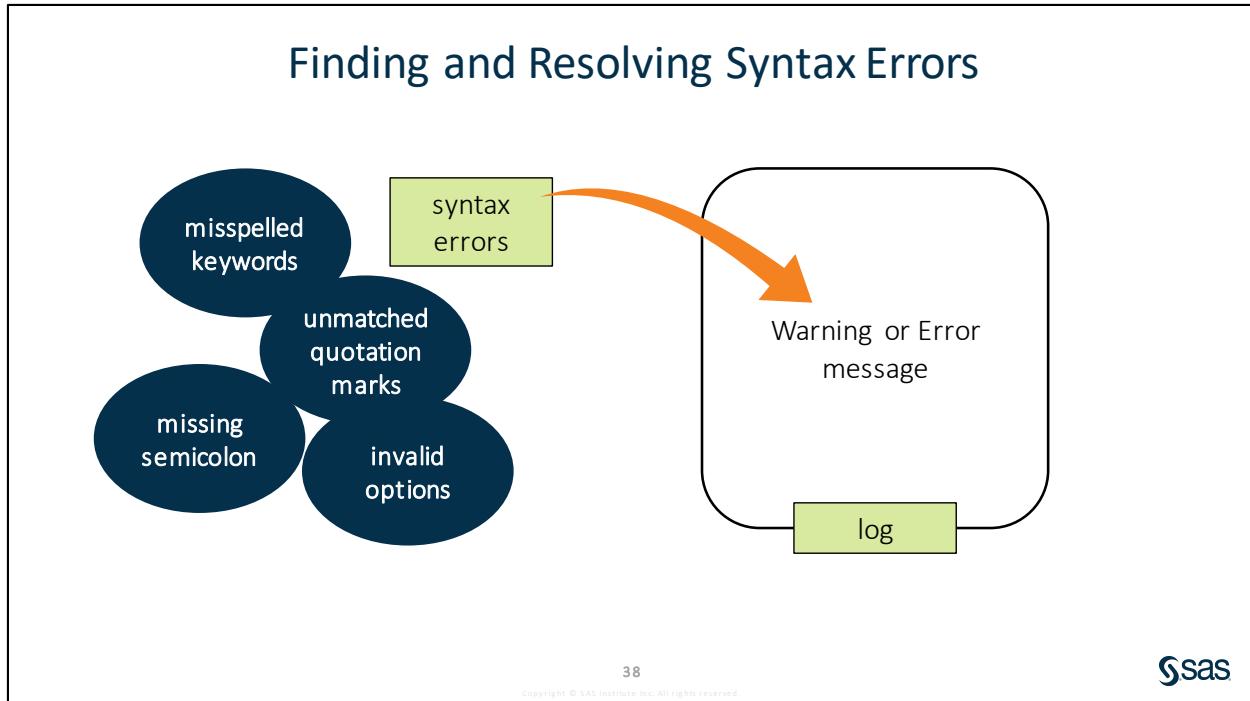
proc means data=mycars
    mean min max maxdec=1;
    var avgmpg;
    class type;
run;

title;

```

Obs	Make	Model	Type	AvgMPG
1	Acura	MDX	SUV	20.0
2	Acura	RSX Type S 2dr	Sedan	27.5
3	Acura	TSX 4dr	Sedan	25.5
4	Acura	TL 4dr	Sedan	24.0
5	Acura	3.5 RL 4dr	Sedan	21.0
6	Acura	3.5 RL w/Navigation 4dr	Sedan	21.0
7	Acura	NSX coupe 2dr manual S	Sports	20.5
8	Audi	A4 1.8T 4dr	Sedan	26.5

End of Demonstration



Syntax errors are a fact of programming life. As a programmer, it is incredibly valuable to be able to identify, diagnose, and fix syntax errors in your code. A syntax error is an error in the spelling or grammar of a SAS statement. Examples of syntax errors include misspelled keywords, unmatched quotation marks, missing semicolons, and invalid options. You can catch some syntax errors, such as an unmatched quotation mark, by paying attention to the color-coded syntax. When SAS finds a syntax error in your submitted program, a warning or error message is written in the log.



Finding and Resolving Syntax Errors

Scenario

Find and resolve some common syntax errors.

Files

- **p101d03.sas**
- **sashelp.cars** – a sample table provided by SAS that includes basic information about 428 cars

Notes

- Some common syntax errors are unmatched quotation marks, missing semicolons, misspelled keywords, and invalid options.
- Syntax errors might result in a warning or error in the log.
- Refer to the log to help diagnose and resolve syntax errors.

Demo

1. Open the **p101d03.sas** program from the **demos** folder. Identify the three syntax errors but do not fix them. Run the program.
2. Carefully review the messages in the log.

Note: The Log Summary is available to view the notes, warnings, and errors.

3. Fix the code and rerun the program.

```

data mycars;
  set sashelp.cars;
  AvgMPG=mean(mpg_city, mpg_highway);
run;

title "Cars with Average MPG Over 35";
proc print data=mymcars;
  var make model type avgmpg;
  where AvgMPG > 35;
run;

title "Average MPG by Car Type";
proc means data=mymcars mean min max maxdec=1;
  var avgmpg;
  class type;
run;

title;

```

End of Demonstration

1.04 Activity

Open **p101a04.sas** from the **activities** folder and perform the following tasks:

1. Format the program to improve the spacing. What syntax error is detected? Fix the error and run the program.
2. Read the log and identify any additional syntax errors or warnings. Correct the program and format the code again.
3. Add a comment to describe the changes that you made to the program.
4. Run the program and examine the log and results. How many rows are in the **canadashoes** data?

```
data canadashoes; set sashelp.shoes;
  where region="Canada";
  Profit=Sales-Returns;run;

proc print data=canadashoes;run;
```

40

Copyright © SAS Institute Inc. All rights reserved.



Extending Your Learning

Extended Learning - SAS® Programming 1: Essentials

[Dashboard](#) / [Courses](#) / Extended Learning - SAS® Programming 1: Essentials

General

Thank you for taking the SAS® Programming 1: Essentials course. You are invited to extend your learning experience by using the resources listed below.

Course Materials

- | | | |
|--|---|--------------------------|
| | SAS® Programming 1: Essentials Course Notes - English | <input type="checkbox"/> |
| | SAS® Programming 1: Essentials Course Notes - German | <input type="checkbox"/> |
| | SAS® Programming 1: Essentials Course Notes - French (SAS 9 - 2017) | <input type="checkbox"/> |

FREE SOFTWARE FOR LEARNING

Don't have access to SAS?
Download [SAS® University Edition](#), a learning version of SAS.

ADDITIONAL RESOURCES

Want SAS Training News?
Subscribe to the SAS Learning Report

42

Copyright © SAS Institute Inc. All rights reserved.



The Extended Learning page is designed to supplement your learning for SAS Programming 1. The Extended Learning page includes the following resources:

- PDF version of the course notes in English and other languages
- course files
- case studies for additional practice and application
- links to papers, videos, blogs, and other resources to learn more about related topics

Beyond SAS Programming 1

What if you want to ...

... use point-and-click SAS Studio tasks to generate code?

- Watch the video [Getting Started with SAS Studio](#).
- View additional free video tutorials on [using SAS Studio tasks](#).

... use the SAS windowing environment in this class?

- Watch the video [Writing and Submitting SAS Code: Choosing an Editor](#).
- Complete the SAS windowing environment activity on the Extended Learning Page.

... learn about using Enterprise Guide tasks to generate code?

- Visit the [Learn SAS Enterprise Guide](#) page for videos, tutorials, and training.
- Take the [SAS Enterprise Guide 1: Querying and Reporting](#) course.

43

Copyright © SAS Institute Inc. All rights reserved.

Links

- Watch the video [Getting Started with SAS Studio](#).
- View additional free video tutorials about [using SAS Studio tasks](#).
- Watch the video [Writing and Submitting SAS Code: Choosing an Editor](#).
- Visit the [Learn SAS Enterprise Guide](#) page for videos, tutorials, and training.
- Take the [SAS Enterprise Guide 1: Querying and Reporting](#) course.

Beyond SAS Programming 1

What if you want to ...

... use Jupyter Notebook to submit code to SAS?

- Read the blog post [How to run SAS programs in Jupyter Notebook](#).
- Read the instructions and download Jupyter kernel for SAS on the [SAS GitHub page](#).

... write code to take advantage of cloud-enabled SAS Viya?

- Watch the video [An Introduction to SAS Viya Programming for SAS 9 Programmers](#).
- Take the [Programming for SAS Viya](#) course after SAS Programming 1.

... integrate open source languages with SAS?

- Take the free [SAS Programming for R Users](#) course.
- Read the [Getting Started with SAS Viya for R documentation](#).

44
Copyright © SAS Institute Inc. All rights reserved.



Links

- Read the blog post [How to run SAS programs in Jupyter Notebook](#).
- Read instructions and download Jupyter kernel for SAS on the [SAS GitHub page](#).
- Watch the videos on [An Introduction to SAS Viya Programming for SAS 9 Programmers](#).
- Take the [Programming for SAS Viya](#) course after SAS Programming 1.
- Take the free [SAS Programming for R Users](#) course.
- Use the [Getting Started with SAS Viya for R documentation](#).

1.4 Solutions

Solutions to Activities and Questions

1.02 Activity – Correct Answer

#	Name	Member Type	File Size	Last Modified
1	CLASS_BIRTHDATE	DATA	128KB	01/15/2020 10:02:41
2	CLASS_TEACHERS	DATA	128KB	01/15/2020 10:02:41
3	CLASS_TEST2	DATA	128KB	01/15/2020 10:02:41
4	CLASS_TEST3	DATA	128KB	01/15/2020 10:02:41
5	CLASS_UPDATE	DATA	128KB	01/15/2020 10:02:41
6	EU_OCC	DATA	448KB	01/15/2020 10:02:41
7	NP_CODELOOKUP	DATA	320KB	01/15/2020 10:02:41
8	NP_FINAL	DATA	128KB	01/15/2020 10:02:41
9	NP_LARGE PARKS	DATA	128KB	01/15/2020 10:02:41
10	NP_MULTIYR	DATA	6MB	01/15/2020 10:02:42
11	NP_SPECIES	DATA	8MB	01/15/2020 10:02:43
12	NP_SUMMARY	DATA	128KB	01/15/2020 10:02:42
13	NP_TRAFFIC	DATA	320KB	01/15/2020 10:02:42
14	NP_WESTWEATHER	DATA	1MB	01/15/2020 10:02:42
15	STORM_2017	DATA	128KB	01/15/2020 10:02:42
16	STORM_BASINCODES	DATA	128KB	01/15/2020 10:02:43
17	STORM_DAMAGE	DATA	128KB	01/15/2020 10:02:43
18	STORM_DETAIL	DATA	7MB	01/15/2020 10:02:43
19	STORM_FINAL	DATA	576KB	01/15/2020 10:02:43
20	STORM_RANGE	DATA	384KB	01/15/2020 10:02:43
21	STORM_SUBBASINCODES	DATA	128KB	01/15/2020 10:02:43
22	STORM_SUMMARY	DATA	448KB	01/15/2020 10:02:43

Confirm that 22 SAS tables were created.



1.03 Activity – Correct Answer

Open p101a03.sas from the activities folder and perform the following tasks:

1. View the code. How many steps are in the program?
There are three steps: one DATA step and two PROC steps.
2. How many statements are in the PROC PRINT step?
four statements
3. How many global statements are in the program?
three TITLE statements
4. Run the program and view the log.
5. How many observations were read by the PROC PRINT step?
11 observations

1.04 Activity – Correct Answer

How many rows are in the `canadashoes` data? 37

```
/* Unbalanced quotation mark and  
misspelled PROC fixed.*/  
  
data canadashoes;  
  set sashelp.shoes;  
  where region="Canada";  
  Profit=Sales-Retruns;  
run;  
  
proc print data=canadashoes;  
run;
```

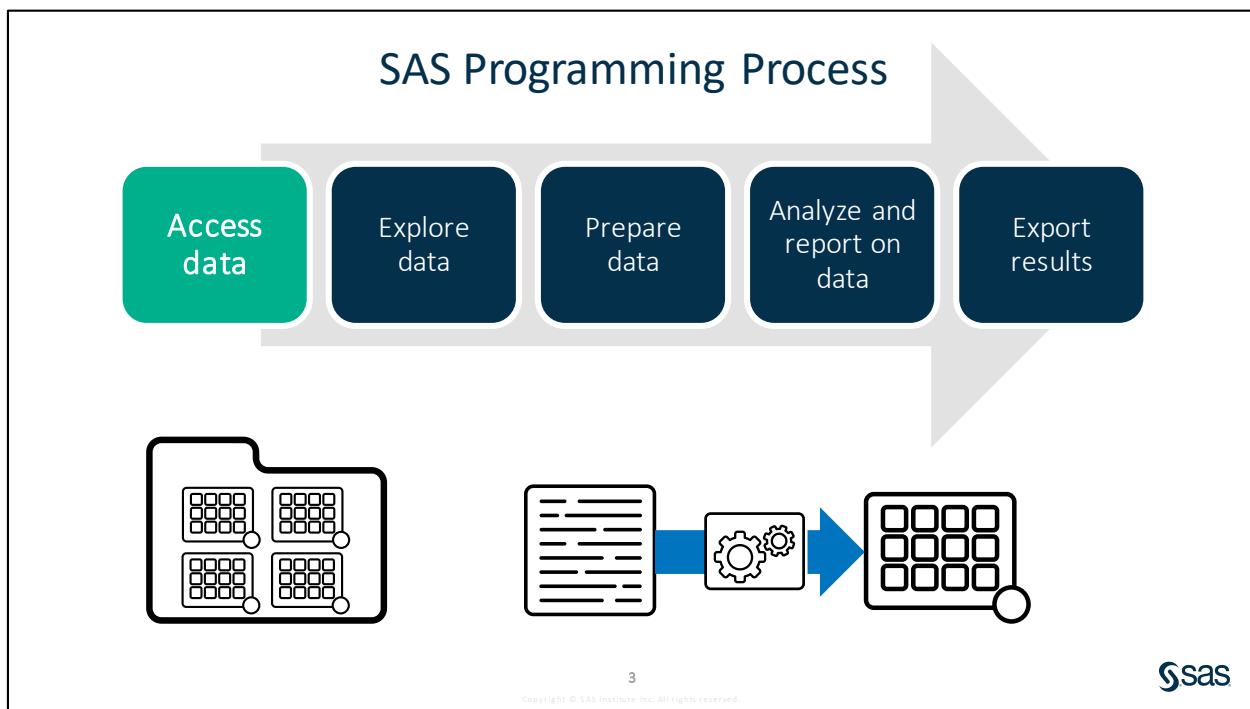
Formatting the code identifies the missing quotation mark.



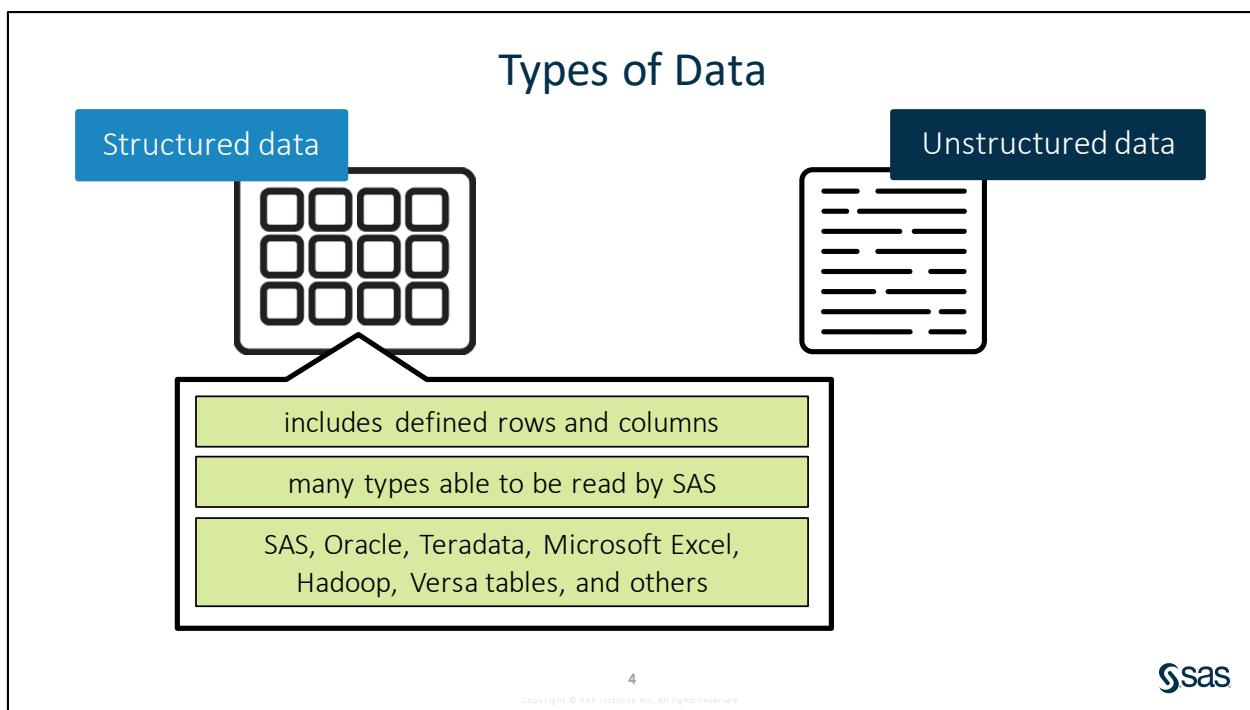
Lesson 2 Accessing Data

2.1 Understanding SAS Data	2-3
2.2 Accessing Data through Libraries.....	2-13
Demonstration: Exploring Automatic SAS Libraries	2-21
Demonstration: Using a Library to Read Excel Files.....	2-25
2.3 Importing Data into SAS	2-28
Demonstration: Importing a Comma-Delimited (CSV) File.....	2-31
Practice.....	2-35
2.4 Solutions	2-38
Solutions to Practices	2-38
Solutions to Activities and Questions.....	2-39

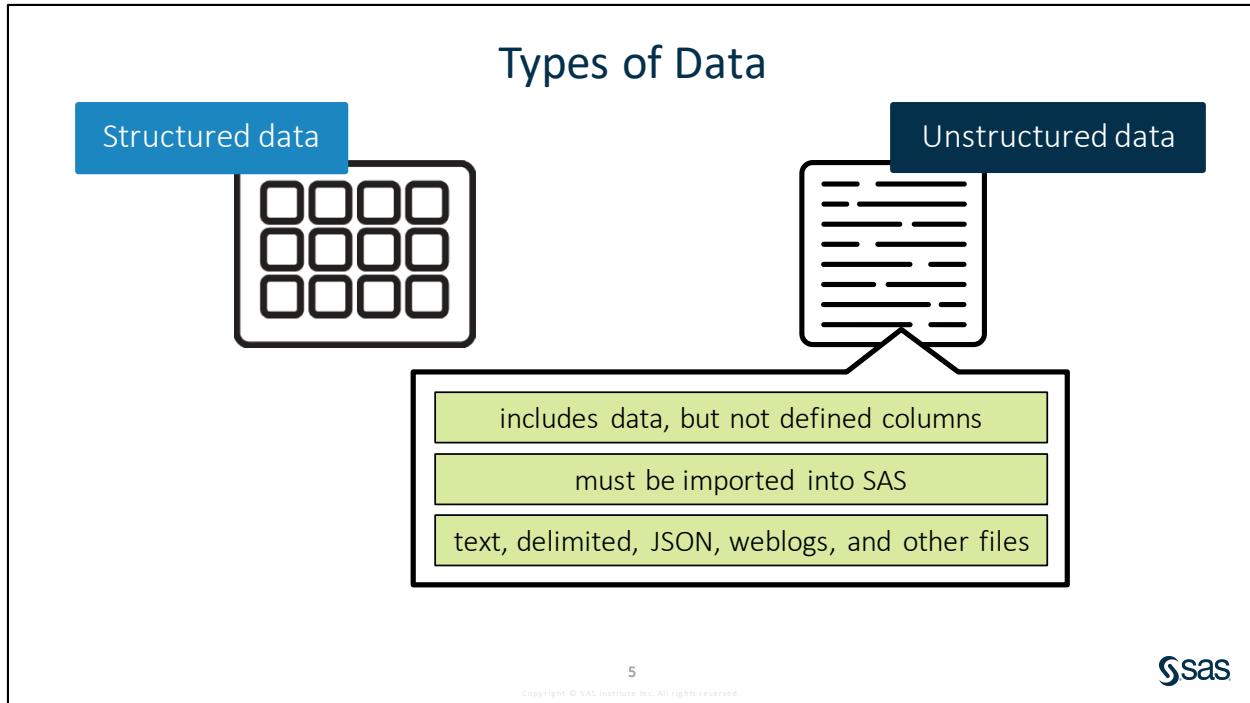
2.1 Understanding SAS Data



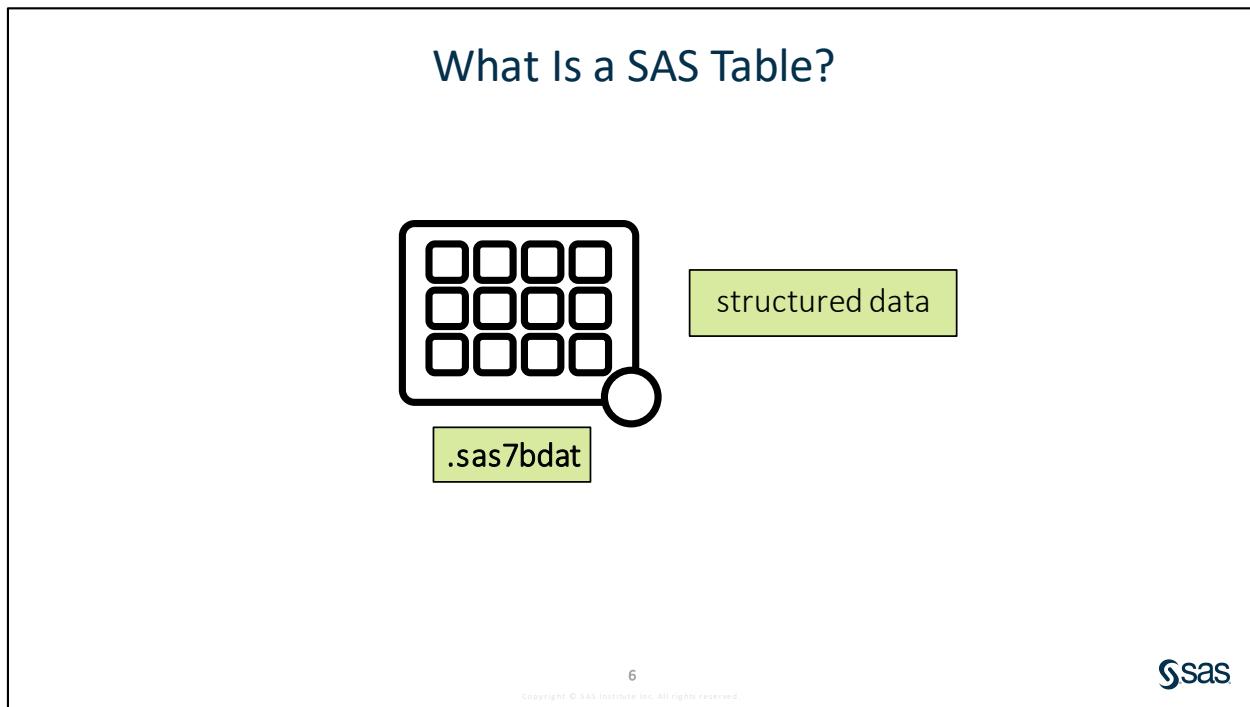
Accessing data is the first step in the SAS programming process. There are many types of data files, and SAS makes it easy to access data and use it for reporting and analysis.



SAS has engines to enable it to understand and read various types of structured data.



Unstructured data must be imported into SAS before you can analyze or report on it. SAS makes importing data easy too.



A *SAS table* is a structured data file that has defined columns and rows. SAS tables have the file extension *.sas7bdat*.

What Is a SAS Table?



7



Copyright © SAS Institute Inc. All rights reserved.

There are two parts to a SAS table: a *descriptor* portion and a *data* portion. The descriptor portion contains information about the attributes of the table, or metadata. The metadata includes general properties such as the table name, the number of rows, and the date and time that the table was created. The descriptor portion also includes the column definitions. The data portion of a SAS table contains the data values, stored in columns.

SAS Terminology



8



Copyright © SAS Institute Inc. All rights reserved.

Required Column Attributes for SAS Tables



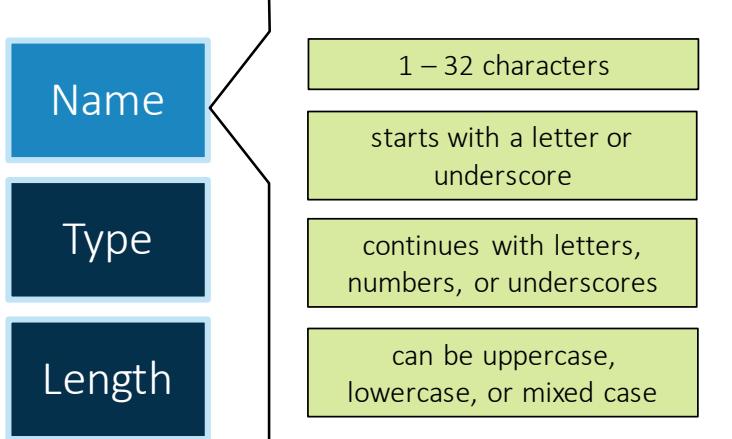
In SAS, all columns must have a name, type, and length.



9

Copyright © SAS Institute Inc. All rights reserved.

Required Column Attributes: Name



10

Copyright © SAS Institute Inc. All rights reserved.

Column names are stored in the case that you use when you create the column, and that is the way the column name appears in reports. After a column has been created, it can be typed in any case in your code without affecting the way that it is stored.

Note: These same naming conventions should be followed for SAS table names.

Depending on the environment used to submit your SAS code, SAS might allow for spaces and special symbols other than underscores in column and table names. If you use data sources other than SAS that have flexible column-name rules, SAS can make allowances for that. However, for simplicity and consistency, it is recommended to follow the standard SAS naming conventions.

2.01 Multiple Answer Question

Which column names are valid? (Select all that apply.)

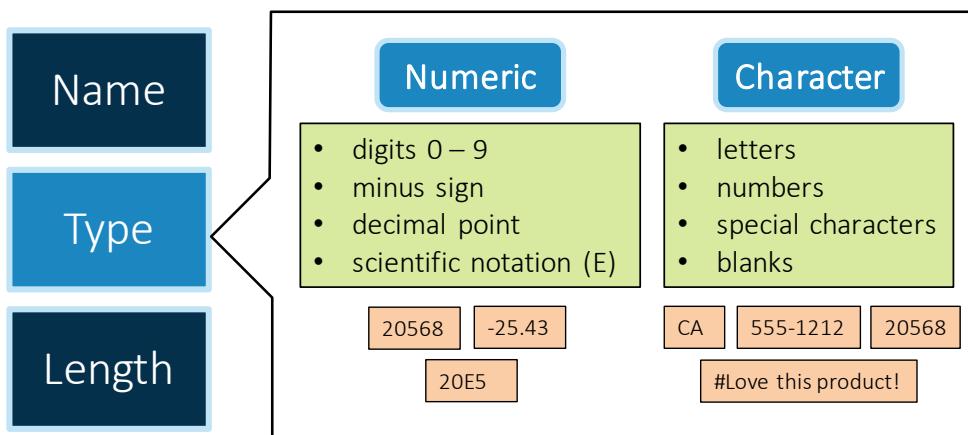
- a. month6
- b. 6month
- c. month#6
- d. month 6
- e. month_6
- f. Month6

11



Copyright © SAS Institute Inc. All rights reserved.

Required Column Attributes: Type

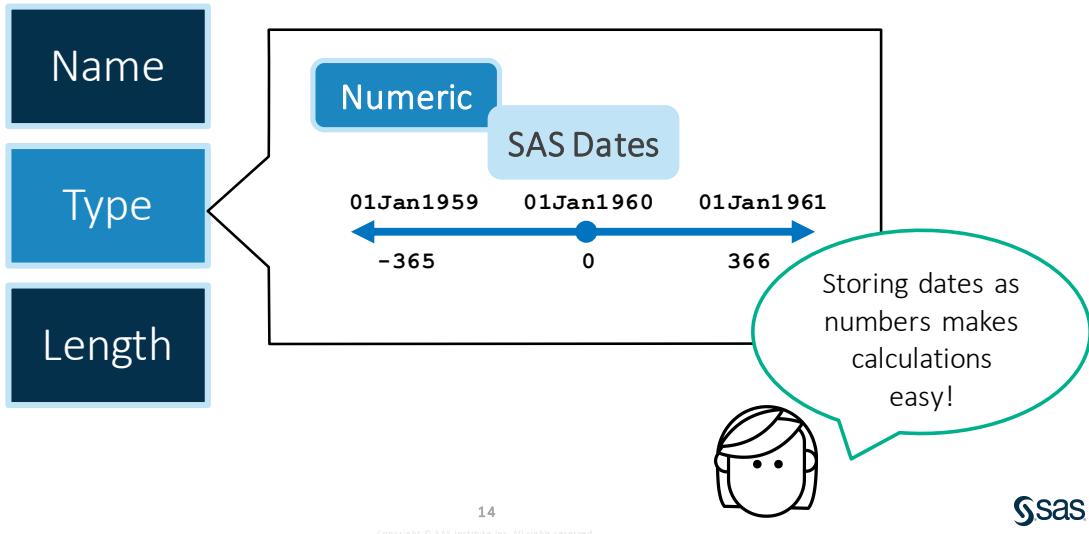


13



Copyright © SAS Institute Inc. All rights reserved.

Required Column Attributes: Type

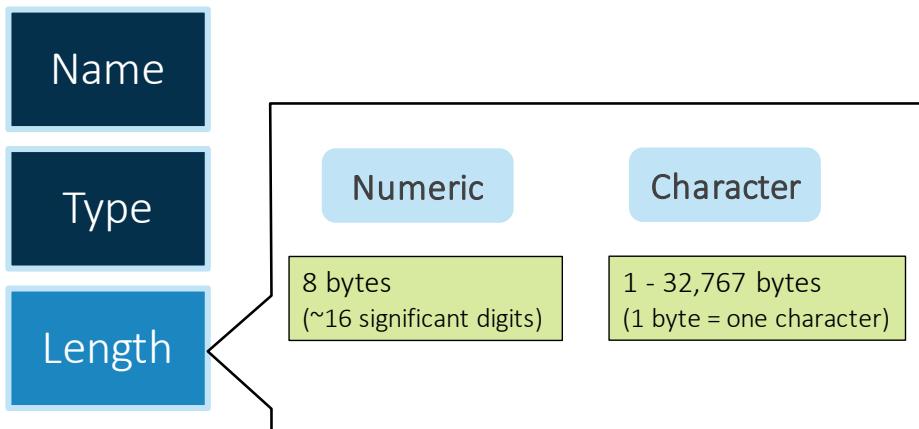


SAS date values represent the number of days between January 1, 1960, and a specified date. SAS can perform calculations on dates ranging from A.D. 1582 to A.D. 19,900.

SAS time values represent the number of seconds since midnight of the current day.

SAS datetime values represent the number of seconds between midnight on January 1, 1960, and an hour/minute/second within a specified date.

Required Column Attributes: Length



The column length is the number of bytes allocated to store column values. The length is related to the column type. Numeric columns, by default, are always stored as 8 bytes, which is enough for about 16 significant digits. Character columns can be any length between 1 and 32,767 bytes, and a byte stores one character. A column such as **Country Code** that is always a two-letter code might be assigned a length of 2. A column such as **Country Name** that could have a varying number of characters must have a length at least as long as the longest country name.

SAS uses floating-point representation to store numeric values. Floating-point representation supports a wide range of values (very large or very small numbers) with an adequate amount of numerical accuracy. For more information about how SAS stores numeric values, visit [SAS 9.4 Language Reference: Concepts](#).

2.02 Activity

1. Navigate to the location of your course files and open the **data** folder.
Enterprise Guide: Expand **Servers** ⇒ **Local** ⇒ **Files**.
SAS Studio: Expand **Files and Folders**.
2. Double-click the **storm_summary.sas7bdat** SAS table to view it.

How are missing character and numeric values represented in the data?

2.03 Question

Click Table Properties  above the `storm_summary` data to view the table and column attributes. Examine the length of the `Basin` column. Could *East Pacific* be properly stored as a data value in the `Basin` column?

- Yes
- No

Viewing Table and Column Attributes

```
PROC CONTENTS DATA=data-set;  
RUN;
```

```
proc contents data="s:/workshop/data/class_birthdate.sas7bdat";  
run;
```



PROC CONTENTS
creates a report
about the descriptor
portion of the data.

The path provided in the program must be relative to where SAS is running. If SAS is on a remote server, the path points to the server, not the local machine.

Viewing Table and Column Attributes

Data Set Name	s:/workshop/data/class_birthdate.sas7bdat	Observations	19		
Member Type	DATA	Variables	6		
Engine	BASE	Indexes	0		
Created	11/15/2017 11:52:18	Observation Length	48		
Last Modified	11/15/2017 11:52:18	Deleted Observations	0		
Protection		Compressed	NO		
Data Set Type		Engine/Host Dependent Information			
Label		Data Set Page Size	65536		
Data Representation	WINDOWS_64	Number of Data Set Pages	1		
Encoding	wlatin1_Weste	First Data Page	1		
		Max Obs per Page	1361		
		Obs in First Data Page	19		
		Number of Data Set Repairs	0		
		ExtendObsCounter	YES		
		Filename	s:/workshop\data\class_birthdate.sas7bdat		
		Release Created	9.0401M4		
		Host Created	X64_10PRO		
		Owner Name	CARYNTstever		
		File Size	128KB		
		File Size (bytes)	131072		
		Alphabetic List of Variables and Attributes			
		#	Variable	Type	Len
		3	Age	Num	8
		6	Birthdate	Num	8
		4	Height	Num	8
		1	Name	Char	8
		2	Sex	Char	1
		5	Weight	Num	8

21

Copyright © SAS Institute Inc. All rights reserved.

p102a04



The output of PROC CONTENTS is a listing of the information in the descriptor portion of the table. You can also think of this as the metadata or properties of the table. The first two sections of the report give general information about the table, including where the table is stored, when it was created and modified, and the number of rows and columns. The variable list provides the column names along with their type and length.

In the **class_birthdate** table, there are the numeric columns **Age**, **Birthdate**, **Height**, and **Weight** that all have a length of 8. There are also two character columns. **Name** is 8 bytes, meaning it can store names with up to eight characters. **Sex** has a length of 1, which is appropriate because it contains only one-letter codes.

2.04 Activity

Open **p102a04.sas** from the **activities** folder and perform the following task:

1. Write a PROC CONTENTS step to generate a report of the **storm_summary.sas7bdat** table properties. Highlight the step and run only the selected code.
2. How many observations are in the table?
3. How is the table sorted?

```
PROC CONTENTS DATA=data-set;  
RUN;
```

2.2 Accessing Data through Libraries

Using a Library to Read SAS Files

```
proc contents data="s:/workshop/data/class.sas7bdat";  
run;
```

where the
data is located

name and
type of data

25

Copyright © SAS Institute Inc. All rights reserved.

So far we have used a hardcoded file path to the SAS table that we want to access, and that file path has the two pieces of information that are required for SAS to read the file: where the data is located and what type of data it is. Because we have been reading a SAS table, providing a path to the data and file name in quotation marks works perfectly.



Discussion

What challenges might arise if you use a fixed path in your program?



Using a Library to Read SAS Files

```
proc contents data="s:/workshop/data/class.sas7bdat";
run;
```

I hope I don't need to write that file path again and again in a long program!

Think of the editing I'll need to do if the data changes location!

What if I want to access another type of data?



27

Copyright © SAS Institute Inc. All rights reserved.

Using a Library to Read SAS Files

create the library

```
LIBNAME libref engine "path";
```

name of library

what type of data it is

where the data is located

- eight-character maximum
- starts with a letter or underscore
- continues with letters, numbers, or underscores

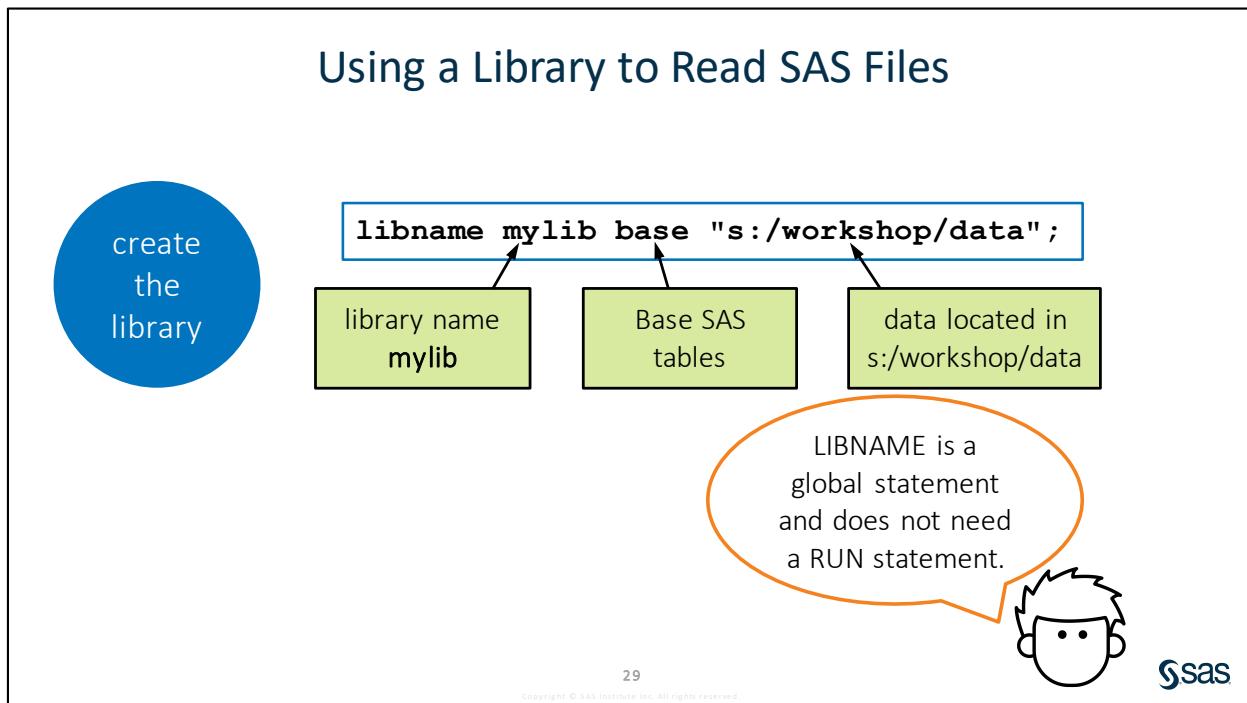
28

Copyright © SAS Institute Inc. All rights reserved.

SAS libraries provide a way to specify the two required pieces of information – the location and file type – in a very simple and efficient way. You can think of a library as a collection of data files that are the same type and in the same location.

A library is created with the LIBNAME statement. This is a global statement, and it does not need a RUN statement at the end.

- The statement begins with the keyword LIBNAME, followed by what is referred to as a *library reference*, or *libref*. The libref is the name of the library. The libref must be eight characters or less, must start with either a letter or underscore, and can include only letters, numbers, and underscores.
- After the libref, specify the engine, which is related to the type of data being accessed. The engine is a behind-the-scenes set of instructions that enables SAS to read structured data files directly, without having to do a separate, manual import into SAS. There are dozens of engines available, including Base for SAS tables, Excel, Teradata, Hadoop, and many others.
- Finally, provide the location or connection information for the data to be read. That can be a physical path or directory, or other options to connect to a database.



SAS complies with operating system permissions that are assigned to the data files referenced by the library. If you have Write access to the files, you are able to use SAS code to add, modify, or delete data files. If you have Read access but do not have Write access, you can read data files via the library, but you cannot make any changes to the files with SAS code.

To prevent SAS from making changes to tables in a library, add ACCESS=READONLY at the end of the LIBNAME statement.

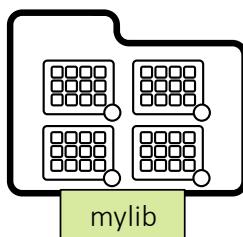
```
libname mylib base "s:/workshop/data" access=readonly;
```

Using a Library to Read SAS Files

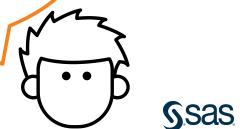
create
the
library

```
libname mylib base "s:/workshop/data";
```

```
libname mylib "s:/workshop/data";
```



The Base SAS engine is the default, so these two statements are the same.



30

Copyright © SAS Institute Inc. All rights reserved.

The path specified must be relative to where SAS is running. If SAS is local, you can specify a path to a folder of files on your own machine. If SAS is on a remote server, the path or folder must be to a location known to the server.

When the LIBNAME statement is submitted, all the information about the location and file type is associated with the library name, or libref.

Using a Library to Read SAS Files

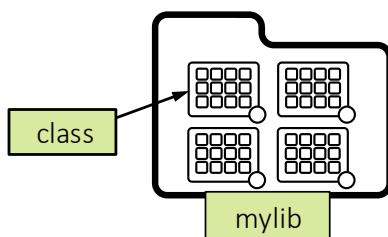
use the
library

libref.table-name

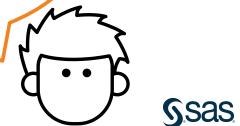
```
proc contents data=
```

mylib.class;

run;



mylib indicates the type of data and the location of the **class** table.



31

Copyright © SAS Institute Inc. All rights reserved.

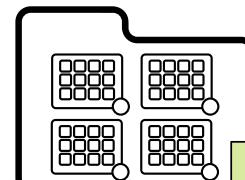
After the library is defined, it can be used as a shortcut to access tables in the program. To do this, specify the libref, a period, and the table name.

Using a Library to Read SAS Files

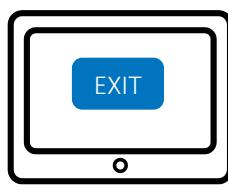
delete
library
reference

```
libname mylib clear;
```

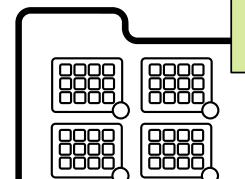
mylib



Your
data is
not
deleted.



mylib



32

Copyright © SAS Institute Inc. All rights reserved.

By default, a libref remains active until it is deleted or the SAS session ends. Remember that the libref is simply a pointer or shortcut to existing data, so although the libref might be deleted when SAS shuts down, the data remains in the same place. When SAS restarts, re-establish the library and libref by submitting the LIBNAME statement again before accessing the data. This is why SAS programs often begin with one or more LIBNAME statements to connect to the various data sources that are used in the code.

2.05 Activity (Required)

1. Open a new program. Write a LIBNAME statement to create a library named **PG1** that reads SAS tables in the **data** folder.

```
libname pg1 base "s:/workshop/data";
```

2. Run the code and verify that the library was successfully assigned in the log.
3. Go back to your program and save it as **libname.sas** in the main course files folder. Replace the file if it exists.

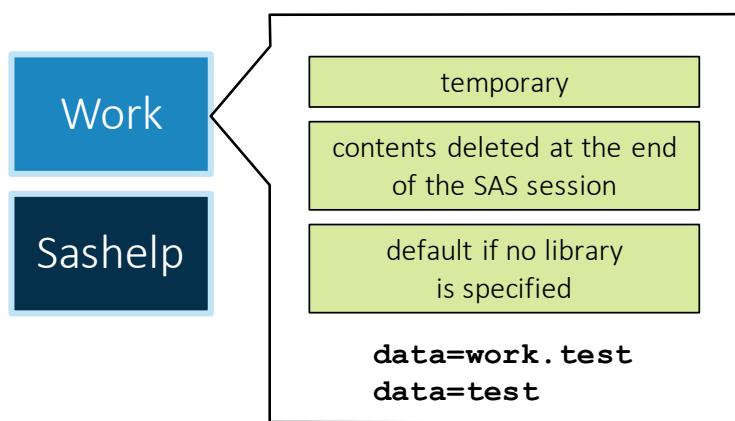
33

Note: The log might indicate that the **pg1** libref refers to the same physical library as another libref, such as **TMP0001** or **_TEMP0**. When a table is opened to view in the data grid, SAS creates a library that points to the folder where the table is located. You do not need to clear the libref that is created by SAS.

2.06 Activity

1. Enterprise Guide: Select **Libraries** in the resources pane and click  to refresh.
SAS Studio: Select **Libraries** in the navigation pane and expand **My Libraries**.
2. Expand the **PG1** library. Why are the Excel and text files in the **data** folder not included in the library?

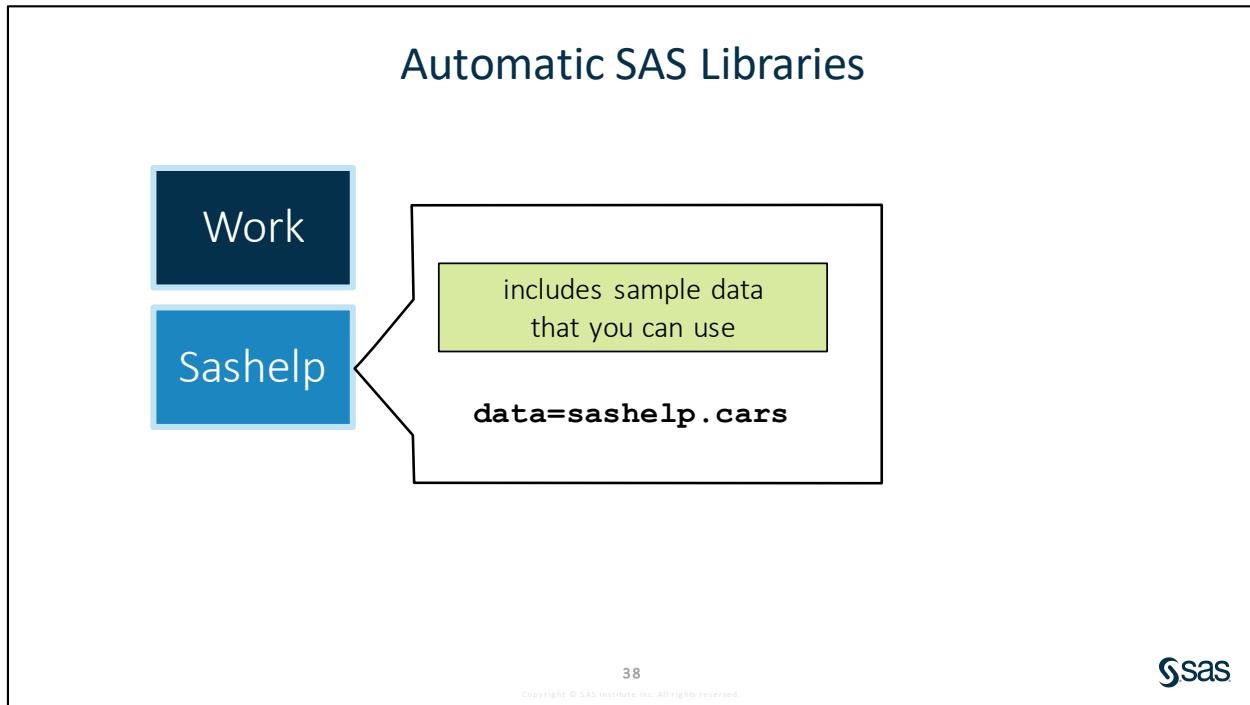
Automatic SAS Libraries



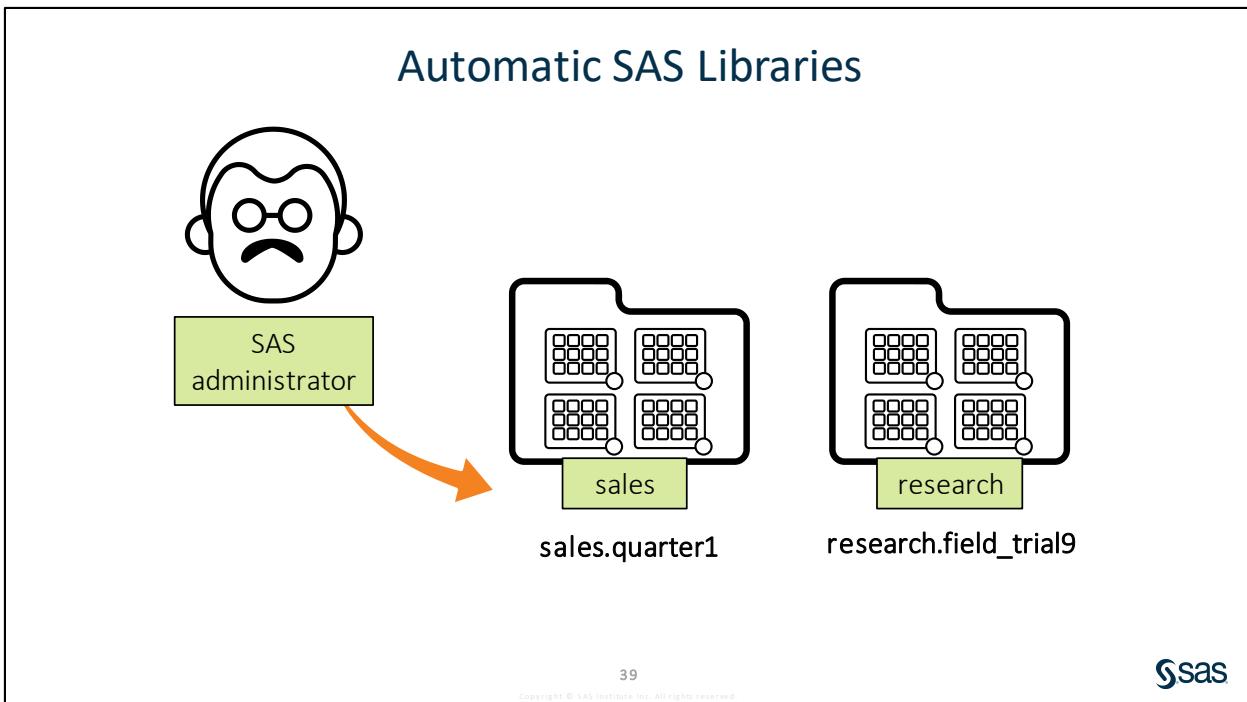
Work and **Sashelp** are also known as *SAS system libraries*. For more information about system libraries, [access this page in SAS Help](#).

The **Work** library is a temporary library that is automatically defined by SAS at the beginning of each SAS session. We say that the **Work** library is temporary because any tables written to the **Work** library are deleted at the end of each SAS session. This library is commonly used in SAS programs because it is a great way to create working files that you do not need to save permanently.

The **Work** library is also considered to be the default library. If a libref is not provided in front of a table name, SAS assumes that the library is **Work**. For example, **test** and **work.test** both reference the temporary table named **test** in the **Work** library.



Another library that SAS automatically defines is the **Sashelp** library. **Sashelp** contains a collection of sample tables and other files. We use several of the sample tables in **Sashelp** in the examples in this course.



If your SAS Platform has an administrator, other automatic libraries might be defined when you open your SAS interface. If libraries are defined for you, you do not need to submit a LIBNAME statement. You can use the libref that was created by your administrator and the table name to reference data in your program.



Exploring Automatic SAS Libraries

Scenario

Use the **Work** and **Sashelp** libraries that are automatically created by SAS. Determine what happens with libraries and tables when SAS restarts.

Files

- **p102d01.sas**
- **sashelp.class** – a sample table provided by SAS that includes basic information about 19 students

Notes

- **Work** and **Sashelp** are system libraries that are automatically defined by SAS.
- Tables stored in the **Work** library are deleted at the end of each SAS session.
- **Work** is the default library, so if a table name without a libref is provided in the program, the table is read from or written to the **Work** library.
- **Sashelp** contains a collection of sample tables and other files that include information about your SAS session.

Demo

1. Open the **p102d01.sas** program from the **demos** folder and find the **Demo** section. Run the demo program and use the navigation pane to examine the contents of the **Work** and **out** libraries.
2. Which table is in the **Work** library? Which table is in the **out** library?
3. Restart SAS.
 - a. Enterprise Guide: In the Servers list, select **Local**, right-click, and select **Disconnect**. Expand **Local** to start SAS again, and then expand **Libraries**.
 - b. SAS Studio: Select **More application options** ⇒ **Reset SAS Session**.
4. Discuss the following questions:
 - a. What is in the **Work** library?
 - b. Why are the **out** and **pg1** libraries not available?
 - c. Is **class_copy2** saved permanently?
 - d. What must be done to re-establish the **out** library?
5. To re-establish the **pg1** library, open and run the **libname.sas** program that was saved previously in the main course files folder.

Note: Whenever you restart SAS Studio or SAS Enterprise Guide, you need to run the **libname.sas** program to re-establish the **pg1** library.

End of Demonstration

Using a Library to Read Excel Files

```
LIBNAME libref XLSX "path/file-name.xlsx";
```

name of
the library

The XLSX engine
reads Excel files.

The path must include
the complete file
name and extension.

```
libname xlclass xlsx "s:/workshop/data/class.xlsx";
```

The XLSX engine requires a license
for SAS/ACCESS Interface to PC Files.

41

Copyright © SAS Institute Inc. All rights reserved.

p102d02



In addition to SAS data, libraries can be used to access many other types data. For example, a library using the XLSX engine can read data directly from Excel spreadsheets.

Remember that when SAS reads or writes data in a program, it must know where the data is located and what format is it in. The only change to the LIBNAME statement syntax is that we specify the XLSX engine, and a path that includes the complete Excel workbook file name and extension. You can think of the Excel workbook as a collection of tables. Each individual worksheet or named range is one table in the collection.

Note: The XLSX engine requires a license for SAS/ACCESS Interface to PC Files, and it also requires SAS 9.4M2 or later.

Using a Library to Read Excel Files

```
OPTIONS VALIDVARNAME=V7;
```

forces table and column names to follow SAS naming conventions

	First_Name	Last_Name	Days_Employed
1	Brad	Majors	136
2	Janet	Weiss	136
3	Everette	Scott	89
4	Frank	Furter	160

```
LIBNAME libref CLEAR;
```

clears the connection to the Excel file

42 Copyright © SAS Institute Inc. All rights reserved.
Sas
p102d02

There are two extra statements that are often used when reading Excel data. The first is the OPTIONS statement, a global statement for specifying system options. Excel does not have any rules for column headings, so they can be longer than 32 characters and include spaces or other special symbols. When SAS reads the Excel data, we can force column names to follow strict SAS naming conventions by using the VALIDVARNAME=V7 system option. Technically, this enforces the column naming rules established with SAS 7. With this option set, SAS replaces any spaces or special symbols in column names with underscores, and names greater than 32 characters are truncated.

In SAS Studio and Enterprise Guide, the VALIDVARNAME= option is set to ANY by default. ANY enables column names to contain special characters, including spaces. If a column name contains special characters, the column name must be expressed as a SAS name literal.

"var-name"n

The default value for VALIDVARNAME can also be changed in the interface options.

Enterprise Guide: Select **Tools** ⇒ **Options** ⇒ **Data General** and change **Valid variable names** to **Basic variable names**.

SAS Studio: Select **More application options** ⇒ **Preferences** ⇒ **Tables** and change **SAS variable name policy** to **V7**.

Note: The SAS windowing environment sets VALIDVARNAME=V7 by default.

When a connection is defined to data sources such as Excel or other databases, it is a good practice to clear, or delete, the libref at the end of the program. While the library is active, it might create a lock on the data preventing others from accessing the file, or it could maintain an active connection to the data sources that is unnecessary. To clear the library reference, use the LIBNAME statement again, name the libref, and use the keyword CLEAR.

Using a Library to Read Excel Files

```
options validvarname=v7;  
libname xlclass xlsx "s:/workshop/data/class.xlsx";
```

```
proc contents data=xlclass.class_birthdate;  
run;
```

```
libname xlclass clear;
```

name of the worksheet
that you want to read

43

Copyright © SAS Institute Inc. All rights reserved.

p102d02



- The OPTIONS statement enforces SAS naming rules for columns.
- The LIBNAME statement creates the **xlclass** library using the XLSX engine to read data from the **class.xlsx** Excel workbook located in s:/workshop/data.
- The PROC CONTENTS step reads the **class_birthdate** worksheet in the **class** workbook.
- The last LIBNAME statement clears the **xlclass** libref.



Using a Library to Read Excel Files

Scenario

Create a library to connect to an Excel workbook and reference an Excel worksheet in the program.

Files

- **p102d02.sas**
- **Storm.xlsx** – an Excel workbook with multiple worksheets that contain storm data

Syntax

```
OPTIONS VALIDVARNAME=V7;
LIBNAME libref XLSX "path/filename.xlsx";
LIBNAME libref CLEAR;
```

Notes

- The XLSX engine enables you to read data directly from Excel workbooks. The XLSX engine requires the SAS/ACCESS Interface to PC Files license.
- The VALIDVARNAME=V7 system option forces table and column names read from Excel to follow SAS naming conventions. Spaces and special symbols are replaced with underscores, and names greater than 32 characters are truncated.
- Date values are automatically converted to numeric SAS date values and formatted for easy interpretation.
- Worksheets from the Excel workbook can be referenced in a SAS program as *libref.worksheet-name*.
- When you define a connection to a data source other than a SAS data source, such as Excel or other databases, it is a good practice to delete the libref at the end of your program with the CLEAR option.

Demo

1. Open the **Storm.xlsx** file in Excel to view the data. Notice that, in the **Storm_Summary** worksheet, there are spaces in the **Hem NS** and **Hem EW** column headings. Close the Excel file after you finish viewing it.

Note: The file must be closed before you assign a library to the file.
2. Open **p102d02.sas** from the **demos** folder and find the **Demo** section. Complete the OPTIONS statement to ensure that column names follow SAS naming conventions.
3. Complete the LIBNAME statement to define a library named **xlstorm** that connects to the **Storm.xlsx** workbook.

4. Highlight the OPTIONS and LIBNAME statements and run the selected code. Use the navigation area to find the **xlstorm** library. Open the **storm_summary** table. Notice that the **Hem_NS** and **Hem_EW** columns include underscores. Close the **storm_summary** table.

```
*Complete the OPTIONS statement;
options validvarname=v7;

*Complete the LIBNAME statement;
*Update the path if necessary;

libname xlstorm xlsx "s:/workshop/data/storm.xlsx";
```

5. Modify the PROC CONTENTS statement to read the **storm_summary** table in the **xlstorm** library.
6. Add a statement to clear the **xlstorm** library. Highlight the entire demo program and run the selected code.

Note: In SAS Studio, if you do not submit the VALIDVARNAME=V7 option with the PROC CONTENTS step, the wrapper code by default resets the value of the VALIDVARNAME= option to ANY. This results in spaces in the **Hem EW** and **Hem NS** columns when the **storm_summary** table is read. To ensure that the VALIDVARNAME option remains set as V7, select **More application options** \Rightarrow **Preferences** and change the value of **SAS variable name policy** to **V7**. In Enterprise Guide, if you change the value of VALIDVARNAME with an OPTIONS statement, the wrapper code does not reset the option.

```
*Complete the DATA= option to reference the STORM_SUMMARY
worksheet;
proc contents data=xlstorm.storm_summary;
run;

libname xlstorm clear;
```

Alphabetic List of Variables and Attributes						
#	Variable	Type	Len	Format	Informat	Label
3	Basin	Char	2	\$2.	\$2.	Basin
8	EndDate	Num	8	DATE9.		EndDate
10	Hem_EW	Char	1	\$1.	\$1.	Hem EW
9	Hem_NS	Char	1	\$1.	\$1.	Hem NS
11	Lat	Num	8	BEST.		Lat
12	Lon	Num	8	BEST.		Lon
5	MaxWindMPH	Num	8	BEST.		MaxWindMPH
6	MinPressure	Num	8	BEST.		MinPressure
2	Name	Char	12	\$12.	\$12.	Name
1	Season	Num	8	BEST.		Season
7	StartDate	Num	8	DATE9.		StartDate
4	Type	Char	2	\$2.	\$2.	Type

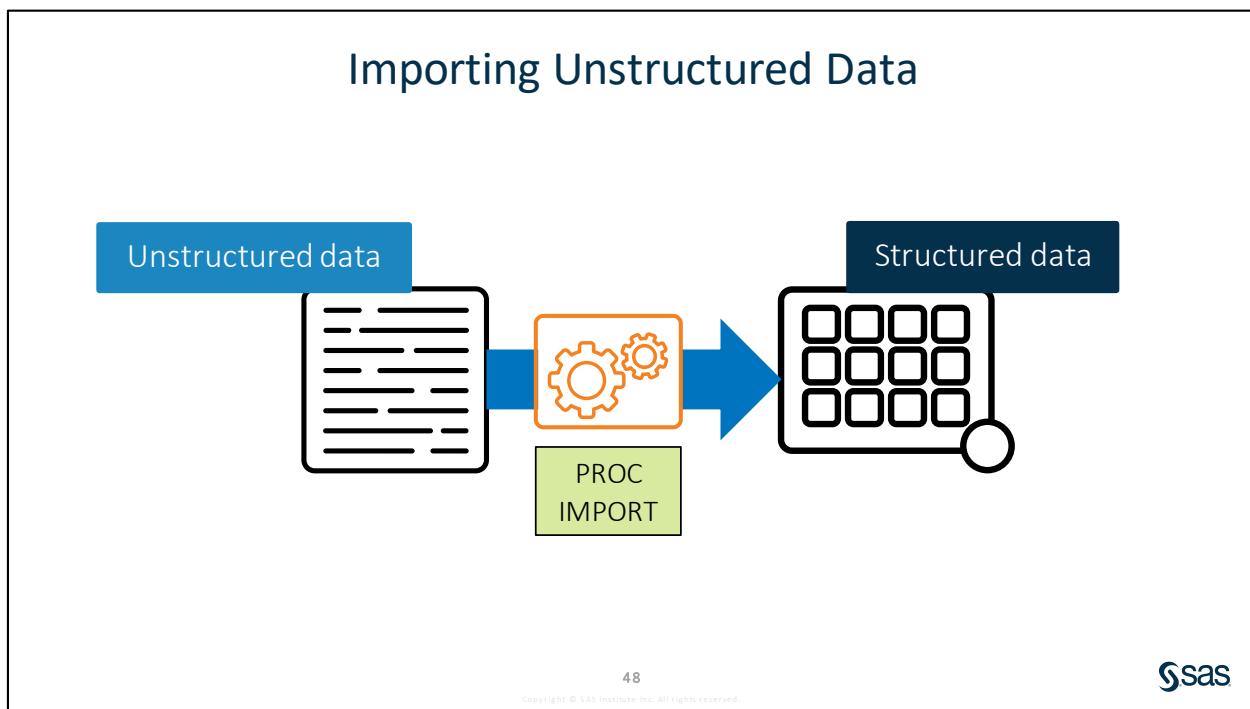
End of Demonstration

2.07 Activity

Open **p102a07.sas** from the **activities** folder and perform the following tasks:

1. If necessary, update the path of the course files in the LIBNAME statement.
2. Complete the PROC CONTENTS step to read the **parks** table in the **NP** library.
3. Run the program. Navigate to your list of libraries and expand the **NP** library. Confirm that three tables are included: **Parks**, **Species**, and **Visits**.
4. Examine the log. Which column names were modified to follow SAS naming conventions?
5. Uncomment the final LIBNAME statement and run it to clear the **NP** library.

2.3 Importing Data into SAS

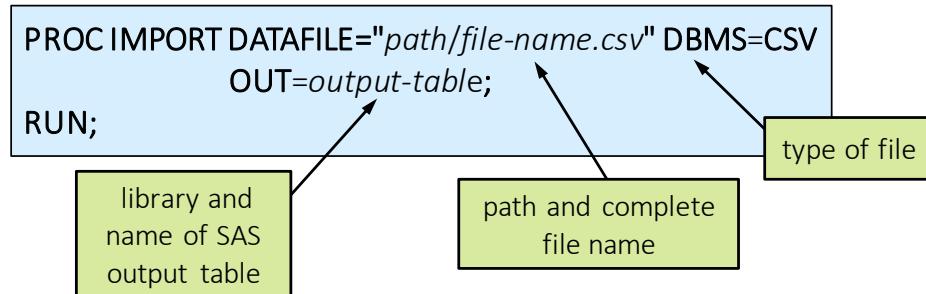


Libraries are an efficient and elegant way to directly access data and use it in a program. However, sometimes you need to access **unstructured** data and to do that, you need to import the file and create a copy as a SAS table.

Let's start with text files as an example. Text files are simply strings of characters to your computer. SAS cannot read text files directly with an engine. We must import the data into a structured format, such as a SAS table, in order to use the data in a program.

There are a number of ways to import data. If you are interested in a point-and-click approach, Enterprise Guide, SAS Studio, and the SAS windowing environment all offer an Import Wizard that enables you to read various file types, specify options, and create a new SAS table. But because this is a programming class, we are going to teach you a simple programming option: the IMPORT procedure.

Importing a Comma-Delimited (CSV) File



49

Copyright © SAS Institute Inc. All rights reserved.

p102d03



PROC IMPORT reads data from an external data source and writes it to a SAS table. SAS can import delimited files with any character acting as the delimiter. To import a comma-delimited file, use the DATAFILE= option to provide the path and complete file name, the DBMS= option to define the file type as CSV, and the OUT= option to provide the library and name of the SAS output table. By default, SAS assumes that column names are found in the first row of the file.

Here are some common DBMS identifiers that are included with Base SAS:

- CSV – comma-separated values.
- JMP – JMP files, JMP 7 or later.
- TAB – tab-delimited values.
- DLM – delimited files, default delimiter is a space. To use a different delimiter, use the DELIMITER= statement.

Here are additional DBMS identifiers included with SAS/ACCESS Interface to PC Files:

- XLSX – Microsoft Excel 2007, 2010 and later
- ACCESS – Microsoft Access 2000 and later

Other DBMS identifiers can be viewed [here](#) in the SAS Help Center.

Importing a Comma-Delimited (CSV) File

```
PROC IMPORT DATAFILE="path/file-name.csv" DBMS=CSV  
      OUT=output-table <REPLACE>;  
      <GUESSINGROWS=n | MAX;>  
RUN;
```

specifies the number of rows used to determine column type and length (default = 20)

replace the output table if it exists

The REPLACE option indicates that the SAS output table should be replaced if it already exists.

By default, SAS scans the first 20 rows of the data to make its best guess for the column attributes, including type and length. It is possible that SAS might incorrectly assume a column's type or length based on the values found in those initial rows. Use the GUESSINGROWS= option to provide a set number or use the keyword MAX to examine all rows. SAS scans the number of rows that you specify to determine type and length of each column in the imported table.



Importing a Comma-Delimited (CSV) File

Scenario

Using PROC IMPORT, import a comma-delimited file and create a new SAS table.

Files

- **p102d03.sas**
- **storm_damage.csv** – a comma-delimited file that includes a description and damage estimates for storms in the US with damages greater than one billion dollars

Syntax

```
PROC IMPORT DATAFILE="path/file-name.csv" DBMS=CSV
      OUT=output-table <REPLACE>;
      <GUESSINGROWS=n|MAX;>
RUN;
```

Notes

- The IMPORT procedure can be used to read delimited text files.
- The DBMS option identifies the file type. The CSV value is included with Base SAS.
- The OUT= option provides the library and name of the SAS output table.
- The REPLACE option is necessary to overwrite the SAS output table if it exists.
- SAS assumes that column names are in the first line of the text file and data begins on the second line.
- Date values are automatically converted to numeric SAS date values and formatted for easy interpretation.
- The GUESSINGROWS= option can be used to increase the number of rows that SAS scans to determine each column's type and length from the default of 20 rows to a maximum of 32,767.

Demo

The **storm_damage.csv** file is in the **data** folder. In this display of the data, notice that column names are in the first row, the data is comma-delimited, and there is a **Date** column. Data values that include commas are enclosed in quotation marks.

Event	Date	Summary	Cost
Hurricane Katrina	25AUG2005	"Category 3 hurricane in the Gulf of Mexico"	\$125 billion
Hurricane Harvey	25AUG2017	"Category 4 hurricane made landfall in Texas"	\$125 billion
Hurricane Maria	19SEP2017	"Category 4 hurricane made landfall in Puerto Rico"	\$90 billion
Hurricane Sandy	30OCT2012	"Extensive damage across the Northeastern United States"	\$75 billion
Hurricane Irma	06SEP2017	"Category 4 hurricane made landfall in Florida"	\$65 billion
Hurricane Andrew	23AUG1992	"Category 5 hurricane hit southern Florida"	\$26.5 billion
Hurricane Ike	12SEP2008	"Category 2 hurricane makes landfall in Texas"	\$25 billion
Hurricane Ivan	12SEP2004	"Category 3 hurricane makes landfall in the Florida Panhandle"	\$24 billion
Hurricane Wilma	24OCT2005	"Category 3 hurricane hit the Yucatan Peninsula"	\$20 billion

1. Open the **p102d03.sas** program in the **demos** folder and find the **Demo** section. Complete the PROC IMPORT step to read **storm_damage.csv** and create a temporary SAS table named **storm_damage_import**. Replace the table if it exists.
2. Complete the PROC CONTENTS step to examine the properties of **storm_damage_import**.
3. Highlight the demo program and submit the selected code.

```
*Complete the PROC IMPORT step;
proc import datafile="s:/workshop/data/storm_damage.csv" dbms=csv
            out=storm_damage_import replace;
run;

*Complete the PROC CONTENTS step;
proc contents data=storm_damage_import;
run;
```

#	Event	Date	Summary	Cost
1	Hurricane Katrina	25AUG2005	Category 3 hurricane initially im...	161300000000
2	Hurricane Harvey	25AUG2017	Category 4 hurricane made lan...	125000000000
3	Hurricane Maria	19SEP2017	Category 4 hurricane made lan...	90000000000
4	Hurricane Sandy	30OCT2012	Extensive damage across sever...	709000000000
5	Hurricane Irma	06SEP2017	Category 4 hurricane made lan...	500000000000
6	Hurricane Andrew	23AUG1992	Category 5 hurricane hits Florid...	483000000000
7	Hurricane Ike	12SEP2008	Category 2 hurricane makes lan...	351000000000
8	Hurricane Ivan	12SEP2004	Category 3 hurricane makes lan...	273000000000
9	Hurricane Wilma	24OCT2005	Category 3 hurricane hits SW FL	245000000000

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Informat
4	Cost	Num	8	BEST12.	BEST32.
2	Date	Num	8	DATE9.	DATE9.
1	Event	Char	22	\$22.	\$22.
3	Summary	Char	764	\$764.	\$764.

End of Demonstration

2.08 Activity

Open **p102a08.sas** from the **activities** folder and perform the following tasks:

1. This program imports a tab-delimited file. Run the program twice and carefully read the log. What is different about the second submission?
2. Fix the program and rerun it to confirm that the import is successful.

```
proc import datafile="s:/workshop/data/storm_damage.tab"
            dbms=tab out=storm_damage_tab;
run;
```

52


Copyright © SAS Institute Inc. All rights reserved.

Importing an Excel File

name of sheet
that you want
to import

```
PROC IMPORT DATAFILE="path/file-name.xlsx" DBMS=XLSX
            OUT=output-table <REPLACE>;
            SHEET=sheet-name;
            RUN;
```

type of file

```
proc import datafile="s:/workshop/data/class.xlsx"
            dbms=xlsx
            out=work.class_test_import_replace;
            sheet=class_test;
run;
```

54


Copyright © SAS Institute Inc. All rights reserved.

The XLSX library engine can read and write Excel data directly, but you might prefer to import a copy of your Excel data as a SAS table and use that SAS table in your program. If SAS/ACCESS to PC Files is licensed, PROC IMPORT can accomplish this. Change the DATAFILE= value and the DBMS option to reference XLSX and use the SHEET= option to tell SAS which worksheet you want to read. PROC IMPORT can read only one spreadsheet at a time, and by default it reads the first worksheet.

Note: If the Excel file is open when PROC IMPORT runs, an error occurs.

Beyond SAS Programming 1

What if you want to ...

... create libraries that are assigned automatically?

- Try the challenge practices for SAS Studio or Enterprise Guide.

... learn about accessing other types of data with SAS/ACCESS products?

- Take one of the [SAS/ACCESS courses](#).
- Dive into the [SAS/ACCESS documentation](#).

... read complex text files?

- Watch the free videos and try the examples provided in the **Reading Raw Data** section of the Extended Learning Page.

56
Copyright © SAS Institute Inc. All rights reserved.



Links

- SAS/ACCESS courses (<http://support.sas.com/training/us/paths/dmgt.html#acc>)
- SAS/ACCESS documentation (<https://support.sas.com/documentation/onlinedoc/access/>)



Practice

If you restarted your SAS session, open and submit the **libname.sas** program in the course files.

Level 1

1. Importing Excel Data from a Single Worksheet

Create a table that contains a copy of the data that is in an Excel workbook. The Excel workbook contains a single worksheet.

- Open **p102p01.sas** from the **practices** folder. Complete the PROC IMPORT step to read **eu_sport_trade.xlsx**. Create a SAS table named **eu_sport_trade** and replace the table if it exists.
- Modify the PROC CONTENTS code to display the descriptor portion of the **eu_sport_trade** table. Submit the program, and then view the output data and the results.

	Sport_Product	Geo_Code	Country	Year	Amt_Import	Amt_Export
1	BALL	BE	Belgium	2000	26,115,000	7,915,000
2	BALL	BE	Belgium	2001	23,456,000	10,546,000
3	BALL	BE	Belgium	2002	33,800,000	14,738,000
4	BALL	BE	Belgium	2003	33,206,000	24,711,000
5	BALL	BE	Belgium	2004	33,360,000	15,890,000
6	BALL	BE	Belgium	2005	41,006,000	40,189,000
7	BALL	BE	Belgium	2006	43,169,000	35,566,000

Alphabetic List of Variables and Attributes						
#	Variable	Type	Len	Format	Informat	Label
6	Amt_Export	Num	8	COMMA15.		Amt_Export
5	Amt_Import	Num	8	COMMA15.		Amt_Import
3	Country	Char	37	\$37.	\$37.	Country
2	Geo_Code	Char	2	\$2.	\$2.	Geo_Code
1	Sport_Product	Char	7	\$7.	\$7.	Sport_Product
4	Year	Num	8	BEST.		Year

Level 2

2. Importing Data from a CSV File

Create a table from a comma-delimited CSV file.

np_traffic.csv

```
ParkName,UnitCode,ParkType,Region,TrafficCounter,ReportingDate,TrafficCount
Big Hole NB,BIHO,National Battlefield,Pacific West,TRAFFIC COUNT AT BATTLE ROAD,31JAN2016,0
Big Hole NB,BIHO,National Battlefield,Pacific West,TRAFFIC COUNT AT BATTLE ROAD,29FEB2016,0
Big Hole NB,BIHO,National Battlefield,Pacific West,TRAFFIC COUNT AT BATTLE ROAD,31MAR2016,0
Big Hole NB,BIHO,National Battlefield,Pacific West,TRAFFIC COUNT AT BATTLE ROAD,30APR2016,183
Big Hole NB,BIHO,National Battlefield,Pacific West,TRAFFIC COUNT AT BATTLE ROAD,31MAY2016,289
```

- Create a new program. Write a PROC IMPORT step to read the **np_traffic.csv** file and create the **traffic** SAS table. Add a PROC CONTENTS step to view the descriptor portion of the newly created table. Submit the program.
- Examine the data interactively. Scroll down to row 37. Notice that the values of **ParkName** and **TrafficCounter** seem to be truncated. Modify the program to resolve this issue.
- Submit the program and verify that **ParkName** and **TrafficCounter** are no longer truncated.

37	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	31JAN2016	0
38	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	29FEB2016	0
39	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	31MAR2016	0
40	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	30APR2016	0
41	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	31MAY2016	5808
42	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	30JUN2016	3747
43	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	31JUL2016	3655
44	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	31AUG2016	4605
45	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	30SEP2016	3605
46	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	31OCT2016	4422

Challenge

3. Importing Data with a Specific Delimiter

Create a table from **np_traffic.dat**. The values in the text file are delimited with a pipe (that is, a vertical bar).

ParkName UnitCode ParkType Region TrafficCounter ReportingDate TrafficCount
Big Hole NB BIHO National Battlefield Pacific West TRAFFIC COUNT AT BATTLE ROAD 31JAN2016 0
Big Hole NB BIHO National Battlefield Pacific West TRAFFIC COUNT AT BATTLE ROAD 29FEB2016 0
Big Hole NB BIHO National Battlefield Pacific West TRAFFIC COUNT AT BATTLE ROAD 31MAR2016 0
Big Hole NB BIHO National Battlefield Pacific West TRAFFIC COUNT AT BATTLE ROAD 30APR2016 183
Big Hole NB BIHO National Battlefield Pacific West TRAFFIC COUNT AT BATTLE ROAD 31MAY2016 289

- Access the [SAS Procedures Guide](#). Expand **Procedures** and find the **IMPORT Procedure** section. Review the syntax and examples to determine how to read a file that is delimited with a specific symbol.
- Use PROC IMPORT to import the **np_traffic.dat** file and create the temporary **traffic2** SAS table.

Partial Results (rows 37-46 of 2,784)

37	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	31JAN2016	0
38	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	29FEB2016	0
39	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	31MAR2016	0
40	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	30APR2016	0
41	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	31MAY2016	5808
42	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	30JUN2016	3747
43	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	31JUL2016	3655
44	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	31AUG2016	4605
45	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	30SEP2016	3605
46	City of Rocks NRES	CIRO	National Reserve	Pacific West	TRAFFIC COUNT AT CIRCLE CREEK ENTRANCE	31OCT2016	4422

4. SAS Studio: Assigning a Library Automatically at Start-Up

Recall that when SAS shuts down, library references are deleted. It might be helpful to have certain libraries that are automatically defined when SAS starts.

- In the Files and Folders section of the navigation pane, navigate to the folder where your course data is stored (for example, **s:/workshop/data**). Right-click the folder and select **Create \Rightarrow Library**. If necessary, enter **PG1** in the **Name** field and click the **Re-create this library at start-up** check box. Click **OK**.
- A LIBNAME statement is added to an autoexec program that runs each time SAS starts. The program can be accessed and edited by selecting **More application options \Rightarrow Edit Autoexec File**.

- c. To test the library, select **More application options** \Rightarrow **Reset SAS Session**. Expand the **Libraries** section of the navigation pane and verify that the **pg1** library is available.
- 5. SAS Enterprise Guide: Assigning a Library Automatically at Start-Up**

Recall that when SAS shuts down, library references are deleted. It might be helpful to have certain libraries that are automatically defined when SAS starts.

- a. Select **Tools** \Rightarrow **Options** \Rightarrow **SAS Programs**. Select the **Submit SAS code when server is connected** check box and click **Edit**. You can include any SAS code that you want to execute each time that SAS starts. Enter a LIBNAME statement, click **Save**, and then click **OK**.

Note: Change the path if necessary to match the location of your course data.

```
libname pg1 base "s:/workshop/data";
```

- b. To test the library, select **Local** in the Servers list, right-click, select **Disconnect**, and then click **Yes**. Expand **Local** to start SAS again, and then expand **Libraries** to confirm that **pg1** is available.

End of Practices

2.4 Solutions

Solutions to Practices

1. Importing Excel Data from a Single Worksheet

```
*Modify the path if necessary;
proc import datafile="s:/workshop/data/eu_sport_trade.xlsx"
            dbms=xlsx
            out=eu_sport_trade
            replace;
run;

proc contents data=eu_sport_trade;
run;
```

2. Importing Data from a CSV File

```
*Modify the path if necessary;
proc import datafile="s:/workshop/data/np_traffic.csv"
            dbms=csv
            out=traffic
            replace;
guessingrows=max;
run;

proc contents data=traffic;
run;
```

3. Importing Data with a Specific Delimiter

```
*Modify the path if necessary;
proc import datafile="s:/workshop/data/np_traffic.dat"
            dbms=dlm
            out=traffic2
            replace;
guessingrows=3000;
delimiter="|";
run;
```

4. SAS Studio: Assigning a Library Automatically at Start-Up

See the practice for detailed steps.

5. SAS Enterprise Guide: Assigning a Library Automatically at Start-Up

See the practice for detailed steps.

End of Solutions

Solutions to Activities and Questions

2.01 Multiple Answer Question – Correct Answers

Which column names are valid? (Select all that apply.)

- a. month6
- b. 6month
- c. month#6
- d. month 6
- e. month_6
- f. Month6

Month6 and month6
are actually the
same column name.



12

Copyright © SAS Institute Inc. All rights reserved.

2.02 Activity – Correct Answer

1. Navigate to the location of your course files and open the **data** folder.
Enterprise Guide: Expand **Servers** ⇒ **Local** ⇒ **Files**
SAS Studio: Expand **Files and Folders**
2. Double-click the **storm_summary.sas7bdat** SAS table to view it.

How are missing character and numeric values represented in the data?

Season	Name	Basin	Type	MaxWindMPH
1	1980	NA	TS	35
2	1980	SP	NR	.
3	1980 AGATHA	EP	TS	115
4	1980 ALBINE	SI	ET	.

blank for
character

period for
numeric

17

Copyright © SAS Institute Inc. All rights reserved.



2.03 Question – Correct Answer

Click Table Properties  above the `storm_summary` data to view the table and column attributes. Examine the length of the `Basin` column. Could *East Pacific* be properly stored as a data value in the `Basin` column?

- Yes
- No

Basin is two bytes, so *East Pacific* would be truncated, and the value would be *Ea.*



19

Copyright © SAS Institute Inc. All rights reserved.



2.04 Activity – Correct Answer

1. Write a PROC CONTENTS step to generate a report of the `storm_summary.sas7bdat` table properties. Highlight the step and run only the selected code.
2. How many observations are in the table? **3118**
3. How is the table sorted? **Season, Name**

```
proc contents data="s:/workshop/data/storm_summary.sas7bdat";
run;
```

Use the location of your course files.

Data Set Name	s:/workshop/data/storm_summary.sas7bdat	Observations	3118
Member Type	DATA	Variables	12
Engine	V9	Indexes	0

Sort Information	
Sortedby	Season Name
Validated	YES
Character Set	ANSI

23

Copyright © SAS Institute Inc. All rights reserved.



2.05 Activity – Correct Answer

1. Open a new program. Write a LIBNAME statement to create a library named **PG1** that reads SAS tables in the **data** folder.

```
libname pg1 base "s:/workshop/data";
```

2. Run the code and verify that the library was successfully assigned in the log.

```
25      libname pg1 base "s:/workshop/data";
NOTE: Libref PG1 was successfully assigned as follows:
      Engine:      BASE
      Physical Name: s:\workshop\data
```

3. Go back to your program and save it as **libname.sas** in the main course files folder. Replace the file if it exists.

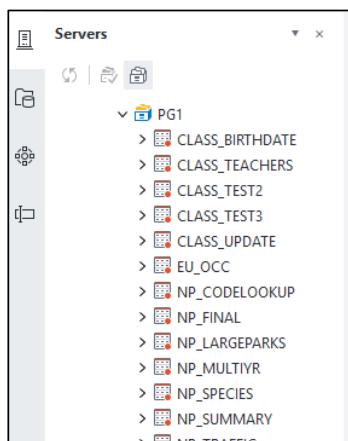
34

Copyright © SAS Institute Inc. All rights reserved.



2.06 Activity – Correct Answer

Why are the Excel and text files in the **data** folder not included in the library?



The **PG1** library uses
the **BASE** engine,
so it reads only
Base SAS tables.



36

Copyright © SAS Institute Inc. All rights reserved.

2.07 Activity – Correct Answer

4. Which column names were modified to follow SAS naming conventions?

```
proc contents data=np.parks;
run;
```

```
35      proc contents data=np.parks;
NOTE: Variable Name Change. Park Code -> Park_Code
NOTE: Variable Name Change. Park Name -> Park_Name
36          run;
```

46



2.08 Activity – Correct Answer

Open **p102a08.sas** from the **activities** folder and perform the following tasks:

- This program imports a tab-delimited file. Run the program twice and carefully read the log. What is different about the second submission?

```
NOTE: Import cancelled. Output dataset WORK.HUR_DAMAGE_TAB already exists.
Specify REPLACE option to overwrite it.
```

- Fix the program and rerun it to confirm that the import is successful.

```
proc import datafile="s:/workshop/data/storm_damage.tab"
            dbms=tab out=storm_damage_tab replace;
run;
```

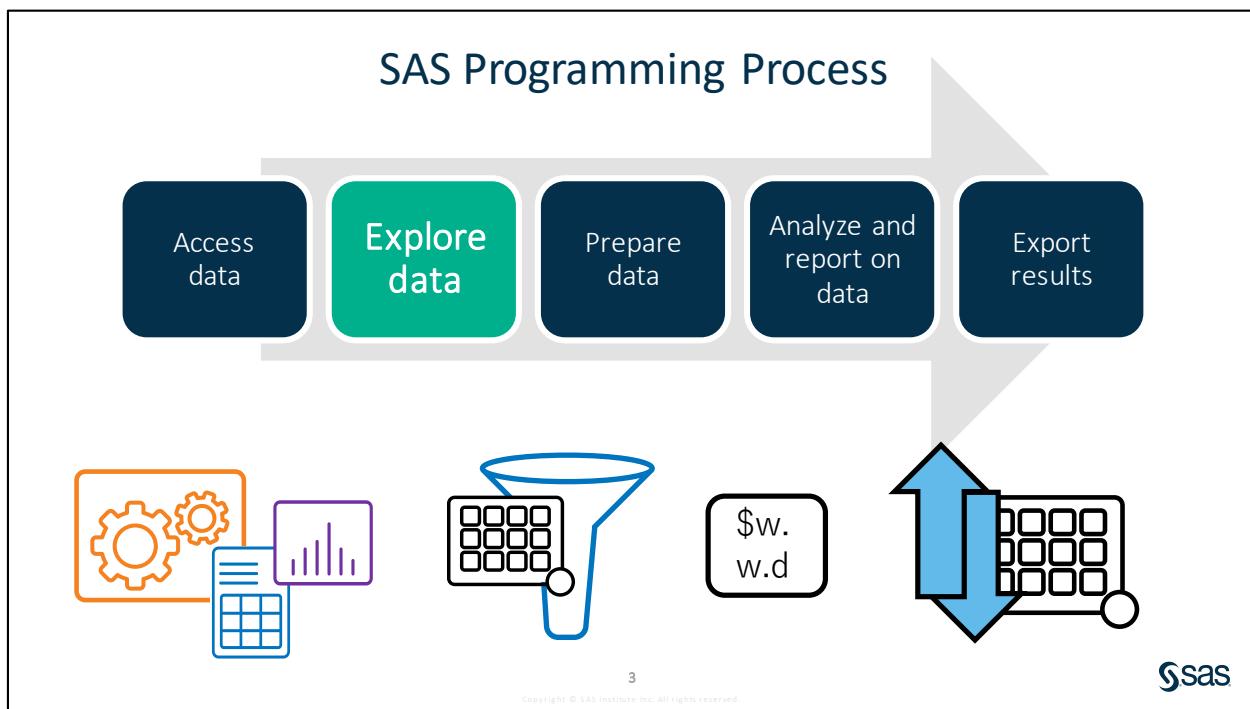
53



Lesson 3 Exploring and Validating Data

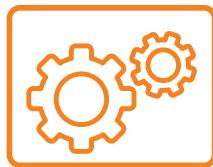
3.1 Exploring Data	3-3
Demonstration: Exploring Data with SAS Procedures.....	3-10
Practice.....	3-14
3.2 Filtering Rows.....	3-18
Demonstration: Filtering Rows with Basic Operators	3-22
Demonstration: Filtering Rows Using Macro Variables.....	3-30
Practice.....	3-33
3.3 Formatting Columns	3-37
Demonstration: Formatting Data Values in Results	3-41
3.4 Sorting Data and Removing Duplicates	3-43
Demonstration: Identifying and Removing Duplicate Values	3-49
Practice.....	3-52
3.5 Solutions	3-54
Solutions to Practices	3-54
Solutions to Activities and Questions.....	3-57

3.1 Exploring Data



Exploring data can include learning about the columns and values that you have, as well as validating data to look for incorrect or inconsistent values. In this lesson, you learn to use some procedures that give you some of this insight. You also learn to subset the data so that you can focus on particular segments, format data so that you can easily understand it, sort data, and identify and clean up duplicate values.

Exploring Data with Procedures



PRINT
MEANS
UNIVARIATE
FREQ

4
Copyright © SAS Institute Inc. All rights reserved.



After accessing data, the next step is to understand it. PROC CONTENTS can be used to confirm column attributes, but often the data is too large or complex for a visual review to be sufficient.

The PRINT, MEANS, UNIVARIATE, and FREQ procedures can be used to quickly and easily explore data.

PRINT Procedure

```
PROC PRINT DATA=input-table (OBS=n);
RUN;
```

use to specify the last observation (row) to read

By default, PROC PRINT lists all columns and rows in the input table.



5

Copyright © SAS Institute Inc. All rights reserved.

p103d01



Setup for the Question

1. Go to support.sas.com/documentation. Click **Programming: SAS 9.4 and Viya**.
2. Under **Syntax - Quick Links**, click **By Name** in the **Procedures** group and find **PRINT**. Examine the syntax and the table of procedure tasks and examples.

PRINT Procedure

[Syntax](#) [Overview](#) [Concepts](#) [Using](#) [Examples](#)

Interaction: A common practice is to sort a data set using PROC SORT before you use the PROC PRINT BY statement. If you sort a CAS table with VARCHAR variables using PROC SORT, VARCHAR variables are converted to CHAR variables.

Note: PROC PRINT supports the VARCHAR data type for CAS tables.

Tips: Each password and encryption key option must be coded on a separate line to ensure that they are properly blotted in the log. Supports the Output Delivery System. For details, see [Output Delivery System: Basic Concepts in SAS Output Delivery System: User's Guide](#).

You can use the ATTRIB, FORMAT, LABEL, TITLE, and WHERE statements. See [SAS DATA Step Statements: Reference](#). For more information, see [Statements with the Same Function in Multiple Procedures](#).

Syntax

```
PROC PRINT <option(s)>;
  BY <DESCENDING> variable-1 <<DESCENDING> variable-2 ...> <NOTSORTED>;
  PAGENTRY variable;
```

Copyright © SAS Institute Inc. All rights reserved.



3.01 Multiple Choice Question

Which statement in PROC PRINT selects variables that appear in the report and determines their order?

- a. BY
- b. ID
- c. SUM
- d. VAR



PRINT Procedure

```
proc print data=sashelp.cars (obs=10);
  var Make Model Type MSRP;
run;
```

Obs	Make	Model	Type	MSRP
1	Acura	MDX	SUV	\$36,945
2	Acura	RSX Type S 2dr	Sedan	\$23,820
3	Acura	TSX 4dr	Sedan	\$26,990
4	Acura	TL 4dr	Sedan	\$33,195
5	Acura	3.5 RL 4dr	Sedan	\$43,755
6	Acura	3.5 RL w/Navigation 4dr	Sedan	\$46,100
7	Acura	NSX coupe 2dr manual S	Sports	\$89,765
8	Audi	A4 1.8T 4dr	Sedan	\$25,940
9	Audi	A41.8T convertible 2dr	Sedan	\$35,940
10	Audi	A4 3.0 4dr	Sedan	\$31,840

9

Copyright © SAS Institute Inc. All rights reserved.

p103d01



This PROC PRINT step lists the first 10 rows, or observations, from the **sashelp.cars** table and displays only the **Make**, **Model**, **Type**, and **MSRP** columns.

MEANS Procedure

```
PROC MEANS DATA=input-table;
  VAR col-name(s);
  RUN;
```

use to specify the
numeric columns
to analyze

By default, PROC MEANS generates simple summary statistics for each numeric column in the input data.

10
Copyright © SAS Institute Inc. All rights reserved.

p103d01



MEANS Procedure

```
proc means data=sashelp.cars;
  var EngineSize Horsepower MPG_City MPG_Highway;
run;
```

The MEANS Procedure						
Variable	Label	N	Mean	Std Dev	Minimum	Maximum
EngineSize	Engine Size (L)	428	3.1967290	1.1085947	1.3000000	8.3000000
Horsepower		428	215.8855140	71.8360316	73.0000000	500.0000000
MPG_City	MPG (City)	428	20.0607477	5.2382176	10.0000000	60.0000000
MPG_Highway	MPG (Highway)	428	26.8434579	5.7412007	12.0000000	66.0000000

11

Copyright © SAS Institute Inc. All rights reserved.

p103d01 

This PROC MEANS step calculates the default statistics – frequency count (N), mean, standard deviation, minimum, and maximum – for each of the columns that is listed in the VAR statement.

By examining the PROC MEANS results, you can identify average values or values that might be outside of an expected range.

UNIVARIATE Procedure

```
PROC UNIVARIATE DATA=input-table;
  VAR col-name(s);
  RUN;
```

use to specify the
numeric columns
to analyze

By default, PROC UNIVARIATE generates summary statistics for each numeric column in the input data.



12

Copyright © SAS Institute Inc. All rights reserved.

p103d01 

UNIVARIATE Procedure

```
proc univariate data=sashelp.cars;
  var MPG_Highway;
run;
```

The UNIVARIATE Procedure Variable: MPG_Highway (MPG [Highway])		
Moments		
N	428	Sum Weights
Mean	26.8434579	Sum Observations
Std Deviation	5.74120072	Variance
Skewness	1.25239527	Kurtosis
Uncorrected SS	322479	Corrected SS
Coeff Variation	21.3877092	Std Error Mean

Basic Statistical Measures	
Location	Variability
Mean	26.84346
Median	26.00000
Mode	26.00000
	Range
	Interquartile Range

Tests for Location: Mu0=0		
Test	Statistic	p Value
Student's t	t 96.7292	Pr > t < .0001
Sign	M 214	Pr >= M < .0001
Signed Rank	S 45903	Pr >= S < .0001

Quantiles (Definition 5)	
Level	Quantile
100% Max	66
99%	44
95%	36
90%	34
75% Q3	29
50% Median	26
25% Q1	24
10%	20
5%	18
1%	16
0% Min	12

Extreme Observations	
Lowest	Highest
Value	Obs
12	167
13	119
14	252
16	217
16	216
Value	Obs
44	156
46	405
51	150
51	374
66	151

13
Copyright © SAS Institute Inc. All rights reserved.

p103d01 

This PROC UNIVARIATE step analyzes **MPG_Highway** and provides several summary statistics, including the five lowest and highest extreme values and their observation numbers.

FREQ Procedure

```
PROC FREQ DATA=input-table;
  TABLES col-name(s);
RUN;
```

use to specify the frequency tables to include in the results



By default, PROC FREQ creates a frequency table for each column in the input table.

14
Copyright © SAS Institute Inc. All rights reserved.

p103d01 

FREQ Procedure

```
proc freq data=sashelp.cars;
  tables Origin Type DriveTrain;
run;
```

The FREQ Procedure

Origin	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Asia	158	36.92	158	36.92
Europe	123	28.74	281	65.65
USA	147	34.35	428	100.00

Type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Hybrid	3	0.70	3	0.70
SUV	60	14.02	63	14.72
Sedan	262	61.21	325	75.93
Sports	49	11.45	374	87.38
Truck	24	5.61	398	92.99
Wagon	30	7.01	428	100.00

DriveTrain	Frequency	Percent	Cumulative Frequency	Cumulative Percent
All	92	21.50	92	21.50
Front	226	52.80	318	74.30
Rear	110	25.70	428	100.00

15

Copyright © SAS Institute Inc. All rights reserved.

p103d01 

This PROC FREQ step creates a separate table for **Origin**, **Type**, and **DriveTrain**. Each table includes a list of the distinct values for the column along with a frequency count, percent, and cumulative frequency and percent. This is a great way to validate the data in your columns. For example, you might notice unexpected values or values that appear in both uppercase and lowercase.



Exploring Data with SAS Procedures

Scenario

Use the PRINT, MEANS, UNIVARIATE, and FREQ procedures to explore and validate data.

Files

- **p103d01.sas**
- **storm_summary** – a SAS table that contains one row per storm for the 1980 through 2016 storm seasons

Syntax

```

PROC PRINT DATA=input-table(OBS=n);
  VAR col-name(s);
RUN;

PROC MEANS DATA=input-table;
  VAR col-name(s);
RUN;

PROC UNIVARIATE DATA=input-table;
  VAR col-name(s);
RUN;

PROC FREQ DATA=input-table;
  TABLES col-name(s);
RUN;

```

Notes

- PROC PRINT lists all columns and rows in the input table by default. The OBS= data set option limits the number of rows read from the input data. The VAR statement limits and orders the columns that are listed.
- PROC MEANS generates simple summary statistics for each numeric column in the input data by default. The VAR statement limits the columns to analyze.
- PROC UNIVARIATE also generates summary statistics for each numeric column in the data by default, but it includes more detailed statistics related to distribution and extreme values. The VAR statement limits the columns to analyze.
- PROC FREQ creates a frequency table for each column in the input table by default. You can limit the columns that are analyzed by using the TABLES statement.

Demo

1. Open **p103d01.sas** from the **demos** folder and find the **Demo** section of the program. Complete the PROC PRINT statement to list the data in **pg1.storm_summary**. Print the first 10 observations. Highlight the step and run the selected code.

```

proc print data=pg1.storm_summary (obs=10);
run;

```

Obs	Season	Name	Basin	Type	MaxWindMPH	MinPressure	StartDate	EndDate	Hem_NS	Hem_EW	Lat	Lon
1	1980		na	TS	35	.	17JUL1980	18NOV1980	N	W	25.7	-91.2
2	1980		SP	NR	.	998	27MAR1980	30MAR1980	S	E	19.1	137.0
3	1980	AGATHA	EP	TS	115	.	09JUN1980	15JUN1980	N	W	12.8	-118.7
4	1980	ALBINE	SI	ET	.	.	27NOV1979	06DEC1979	S	E	19.1	137.0
5	1980	ALEX	WP	TS	40	998	09OCT1980	14OCT1980	N	E	27.2	140.5
6	1980	ALLEN	NA	TS	190	899	31JUL1980	11AUG1980	N	W	21.8	-86.4
7	1980	AMY	SI	NR	132	915	04JAN1980	12JAN1980	S	E	-19.4	119.6
8	1980	BERENICE	SI	TS	.	.	15DEC1979	21DEC1979	S	E	-19.4	119.6
9	1980	BETTY	WP	ET	115	925	28OCT1980	08NOV1980	N	E	14.3	127.5
10	1980	BLAS	EP	TS	58	.	16JUN1980	19JUN1980	N	W	12.3	-110.5

2. Add a VAR statement to include only the following columns: **Season**, **Name**, **Basin**, **MaxWindMPH**, **MinPressure**, **StartDate**, and **EndDate**. Add **list first 10 rows** as a comment before the PROC PRINT statement. Highlight the step and run the selected code.

Enterprise Guide Note: To easily add column names, use the autocomplete prompts to view and select columns. You can either double-click on a column to add it in the program, or start to type the column name and press the spacebar when the correct column is highlighted.

SAS Studio Note: To easily add column names, place your cursor after the keyword VAR. Use the Library section of the navigation pane to find the pg1 library. Expand the **storm_summary** table to see a list of column names. Hold down the Ctrl key and select the columns in the order in which you want them to appear in the statement. Drag the selected columns to the VAR statement.

```

/*list first 10 rows*/
proc print data=pg1.storm_summary(obs=10);
  var Season Name Basin MaxWindMPH MinPressure StartDate
      EndDate;
run;
```

Obs	Season	Name	Basin	MaxWindMPH	MinPressure	StartDate	EndDate
1	1980		na	35	.	17JUL1980	18NOV1980
2	1980		SP	.	998	27MAR1980	30MAR1980
3	1980	AGATHA	EP	115	.	09JUN1980	15JUN1980
4	1980	ALBINE	SI	.	.	27NOV1979	06DEC1979
5	1980	ALEX	WP	40	998	09OCT1980	14OCT1980
6	1980	ALLEN	NA	190	899	31JUL1980	11AUG1980
7	1980	AMY	SI	132	915	04JAN1980	12JAN1980
8	1980	BERENICE	SI	.	.	15DEC1979	21DEC1979
9	1980	BETTY	WP	115	925	28OCT1980	08NOV1980
10	1980	BLAS	EP	58	.	16JUN1980	19JUN1980

3. Copy the PROC PRINT step and paste it at the end of the program. Change **PRINT** to **MEANS**. Remove the OBS= data set option to analyze all observations. Modify the VAR statement to calculate summary statistics for **MaxWindMPH** and **MinPressure**. Add **calculate summary statistics** as a comment before the PROC MEANS statement. Highlight the step and run the selected code.

```
/*calculate summary statistics*/
proc means data=pg1.storm_summary;
    var MaxWindMPH MinPressure;
run;
```

Variable	N	Mean	Std Dev	Minimum	Maximum
MaxWindMPH	3095	79.3179321	31.6853937	6.0000000	213.0000000
MinPressure	2922	961.8545517	288.6582966	-9999.00	1012.00

4. Copy the PROC MEANS step and paste it at the end of the program. Change **MEANS** to **UNIVARIATE**. Add **examine extreme values** as a comment before the PROC UNIVARIATE statement. Highlight the step and run the selected code.

```
/*examine extreme values*/
proc univariate data=pg1.storm_summary;
    var MaxWindMPH MinPressure;
run;
```

The UNIVARIATE Procedure			
Variable: MaxWindMPH			
Moments			
N	3095	Sum Weights	3095
Mean	79.3179321	Sum Observations	245489
Std Deviation	31.6853937	Variance	1003.96417
Skewness	0.5963944	Kurtosis	-0.3710172
Uncorrected SS	22577945	Corrected SS	3106265.15
Coeff Variation	39.947327	Std Error Mean	0.56954597

Extreme Observations			
Lowest		Highest	
Value	Obs	Value	Obs
6	2659	184	702
17	1960	184	1477
23	2757	184	2164
23	1366	190	6
23	1103	213	3017

Missing Values			
Missing Value	Count	Percent Of	
		All Obs	Missing Obs
.	23	0.74	100.00

5. Copy the PROC UNIVARIATE step and paste it at the end of the program. Change **UNIVARIATE** to **FREQ**. Change the VAR statement to a TABLES statement to produce frequency tables for **Basin**, **Type**, and **Season**. Add **list unique values and frequencies** as a comment before the PROC FREQ statement. Highlight the step and run the selected code.

```
/*list unique values and frequencies*/
proc freq data=pg1.storm_summary;
  tables Basin Type Season;
run;
```

The FREQ Procedure				
Basin	Frequency	Percent	Cumulative Frequency	Cumulative Percent
EP	671	21.52	671	21.52
NA	472	15.14	1143	36.66
NI	84	2.69	1227	39.35
SI	588	18.86	1815	58.21
SP	359	11.51	2174	69.72
WP	928	29.76	3102	99.49
na	16	0.51	3118	100.00
Type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
DS	293	9.40	293	9.40
ET	761	24.44	1054	22.84

End of Demonstration



Practice

If you restarted your SAS session, open and submit the **libname.sas** program in the course files.

Level 1

1. Exploring Data with Procedures

The **pg1.np_summary** table contains public use statistics from the National Park Service. Use the PRINT, MEANS, UNIVARIATE, and FREQ procedures to explore the data for possible inconsistencies.

- Open **p103p01.sas** from the **practices** folder. Complete the PROC PRINT statement to list the first 20 observations in **pg1.np_summary**.
- Add a VAR statement to include only the following variables: **Reg**, **Type**, **ParkName**, **DayVisits**, **TentCampers**, and **RVCampers**. Highlight the step and run the selected code.

Do you observe any possible inconsistencies in the data?

Obs	Reg	Type	ParkName	DayVisits	TentCampers	RVCampers
1	A	NM	Cape Krusenstern National Monument	15,000	0	0
2	A	NP	Kenai Fjords National Park	346,534	1,514	0
3	A	NP	Kobuk Valley National Park	15,500	0	0
4	A	PRE	Yukon-Charley Rivers National Preserve	1,146	0	0
5	A	PRE	Bering Land Bridge National Preserve	2,642	0	0
6	A	PRESERVE	Noatak National Preserve	17,000	0	0

- Copy the PROC PRINT step and paste it at the end of the program. Change **PRINT** to **MEANS** and remove the OBS= data set option. Modify the VAR statement to calculate summary statistics for **DayVisits**, **TentCampers**, and **RVCampers**. Highlight the step and run the selected code.

What is the minimum value for tent campers? Is that value unexpected?

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
DayVisits	Recreational Day Visitors	135	966022.48	1568838.29	1146.00	11312786.00
TentCampers	Tent Campers	135	23870.81	60590.83	0	490431.00
RVCampers	RV Campers	135	14761.33	40977.10	0	376744.00

- Copy the PROC MEANS step and paste it at the end of the program. Change **MEANS** to **UNIVARIATE**. Highlight the step and run the selected code.

Are there negative values for any of the columns?

- Copy the PROC UNIVARIATE step and paste it at the end of the program. Change **UNIVARIATE** to **FREQ**. Change the VAR statement to a TABLES statement to produce frequency tables for **Reg** and **Type**. Highlight the step and run the selected code.

Are there any lowercase codes? Are there any codes that occur only once in the table?

Region Code					Type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Reg	Frequency	Percent	Cumulative Frequency	Cumulative Percent					
A	6	4.44	6	4.44	NM	63	46.67	63	46.67
IM	52	38.52	58	42.96	NP	51	37.78	114	84.44
MW	18	13.33	76	56.30	NPRE	1	0.74	115	85.19
NC	1	0.74	77	57.04	NS	10	7.41	125	92.59
NE	13	9.63	90	66.67	PRE	3	2.22	128	94.81
PW	23	17.04	113	83.70	PRESERVE	4	2.96	132	97.78
SE	22	16.30	135	100.00	RIVERWAYS	1	0.74	133	98.52
					RVR	2	1.48	135	100.00

- f. Add comments before each step to document the program. Save the program as **np_validate.sas** in the output folder.

Level 2

2. Using Procedures to Validate Data

The **pg1.np_summary** table contains information about US national parks, monuments, preserves, rivers, and seashores. Valid values for the columns **Reg** and **Type** are as follows:

Reg	Description	Type	Description
A	Alaska	NM	National Monument
IM	Intermountain	NP	National Park
MW	Midwest	NS	National Seashore
NC	National Capital	PRE	National Preserve
NE	Northeast	RVR	National River
PW	Pacific West		
SE	Southeast		

- Create a new program. Write a PROC FREQ step to produce frequency tables for the **Reg** and **Type** columns in the **pg1.np_summary** table. Submit the step and look for invalid values.
- Write a PROC UNIVARIATE step to generate statistics for the **Acres** column in the **pg1.np_summary** table. Notice the observation numbers for the smallest park and the largest park.
- View the **pg1.np_summary** table to identify the name of the smallest and largest parks.

Challenge

3. Generating Extreme Observations Output

The **pg1.eu_occ** table includes monthly occupancy counts for European countries between January 2004 and September 2017.

The SAS Output Delivery System (ODS) gives you options for controlling the type and format of the output that is generated by SAS code. The ODS SELECT statement is used to specify output objects for results. The ODS SELECT statement can be used to generate a report containing only the Extreme Observations output.

Note: To specify an output object, you need to know which output objects your SAS program produces. The ODS TRACE statement writes to the SAS log a trace record that includes the path, the label, and other information about each output object that your SAS program produces. You can find documentation about the ODS TRACE and ODS SELECT statements in the SAS Help Facility and in the online documentation.

- a. Create a new program. Write a PROC UNIVARIATE step to examine **Camp** in the **pg1.eu_occ** table.
- b. Add the ODS TRACE statements before and after PROC UNIVARIATE as follows.

```
ods trace on;
proc univariate data=pg1.eu_occ;
  var camp;
run;
ods trace off;
```

- c. Submit the program and notice the trace information in the SAS log. Determine the name of the Extreme Observations output object.
- d. Delete the ODS TRACE statements. Add an ODS SELECT statement immediately before the PROC UNIVARIATE step and provide the name of the Extreme Observation output object.

Note: This method can be used with other procedures that create multiple tables (such as PROC CONTENTS) to select a portion of the output.

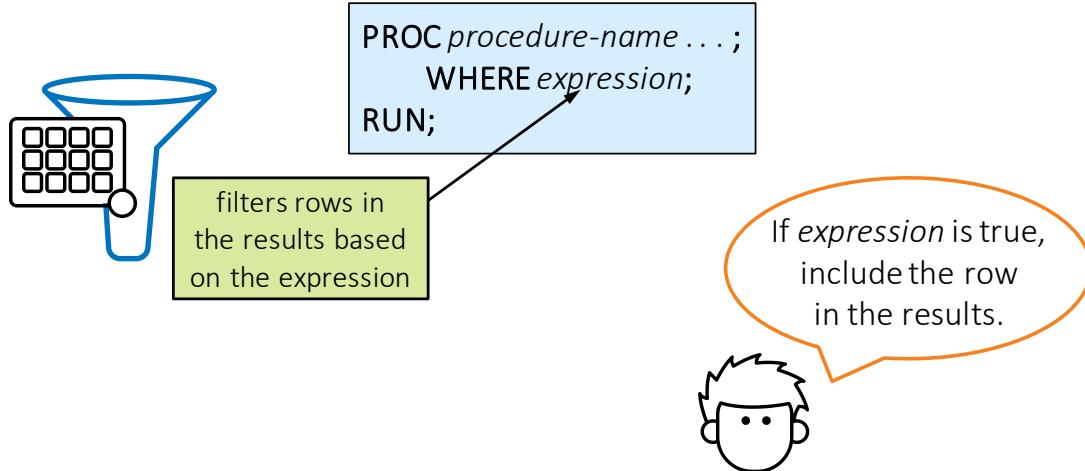
- e. Using the SAS documentation or the syntax Help in the editor, identify the option that specifies the number of extreme observations that are listed in the table. Use the option to change the number of extreme observations from five to 10. Submit the program.

The UNIVARIATE Procedure				
Variable: Camp (Nights Spent at Camp Grounds or RV Parks)				
Extreme Observations				
Lowest		Highest		
Value	Obs	Value	Obs	
0	3204	36451329	1701	
0	3203	37544738	1665	
0	3202	38075450	1677	
0	3201	38118264	1653	
0	3200	44896837	1712	
0	3199	45472015	1676	
0	3198	45863306	1688	
0	3197	46045595	1664	
0	3196	46513414	1700	
0	3195	47550694	1652	

End of Practices

3.2 Filtering Rows

Filtering Rows with the WHERE Statement



What if you want to filter the rows that appear in a PROC PRINT report? Or what if you want to calculate summary statistics for only a subset of the data based on a condition? You can use the powerful and flexible WHERE statement to subset your data. The WHERE statement can be used in PROC PRINT, MEANS, FREQ, UNIVARIATE and many others.

Using Basic Operators in an Expression

WHERE *expression;*

column operator value

= or EQ

< or LT

^= or ~= or NE

>= or GE

> or GT

<= or LE

Type = "SUV"

Type EQ "SUV"

MSRP <= 30000

MSRP LE 30000

20



Copyright © SAS Institute Inc. All rights reserved.

The WHERE statement consists of the keyword WHERE followed by one or more expressions. An expression tests the value of a column against a condition. The expression evaluates as true or false for each row.

Note: Either the symbol or letters can be used to represent these operators in an expression.

Specifying Values in an Expression

WHERE *expression;*

column operator value

Character values are case sensitive and must be enclosed in double or single quotation marks.

Numeric values must be standard numeric (that is, no symbols).

Type = "SUV"

Type = 'Wagon'

MSRP <= 30000

21

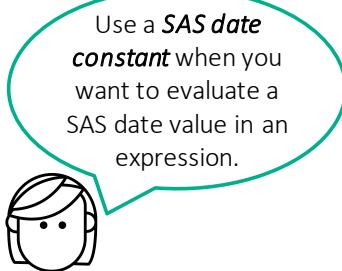


Copyright © SAS Institute Inc. All rights reserved.

Specifying Values in an Expression

`WHERE expression;`

column operator value



"ddmmmyyyy"d

`where date > "01JAN2015"d;`

`where date > "1jan15";`

22

Copyright © SAS Institute Inc. All rights reserved.



Dates are stored as numeric values, so the expression is evaluated based on a numeric comparison. If you want to compare a date column to a fixed date, then you can use the SAS date constant notation. SAS turns the string date into the numeric equivalent in order to evaluate the expression.

Combining Expressions

```
proc print data=sashelp.cars;
  var Make Model Type MSRP MPG_City MPG_Highway;
  where Type="SUV" and MSRP <= 30000;
run;
```

Expressions can be combined with AND or OR.

Obs	Make	Model	Type	MSRP	MPG_City	MPG_Highway
48	Buick	Rendezvous CX	SUV	\$26,545	19	26
67	Chevrolet	Tracker	SUV	\$20,255	19	22
121	Ford	Explorer XLT V6	SUV	\$29,670	15	20
122	Ford	Escape XLS	SUV	\$22,515	18	23
152	Honda	Pilot LX	SUV	\$27,560	17	22

23

Copyright © SAS Institute Inc. All rights reserved.

p103d02



Using the IN Operator

```
WHERE col-name IN (value-1,...,value-n);
WHERE col-name NOT IN (value-1,...,value-n);
```

Values can be character or numeric.

```
where Type="SUV" or Type="Truck" or Type="Wagon";
```

```
where Type in ("SUV", "Truck", "Wagon");
```

```
where Type in ("SUV" "Truck" "Wagon");
```

All three of these statements have the same result.



sas

24

Copyright © SAS Institute Inc. All rights reserved.

The OR keyword can be used to provide multiple values, such as in this example. Notice that each condition has to include TYPE=. This can be tedious if there are several valid values that must be listed. A more efficient approach in this scenario is to use the IN operator to compare to a list of values.

The IN operator works with both numeric and character values. Remember that character values are case sensitive and must be enclosed in quotation marks. The keyword NOT can be used to reverse the logic of the IN operator.



Filtering Rows with Basic Operators

Scenario

Use the WHERE statement and basic operators to subset rows in a procedure.

Files

- **p103d02.sas**
- **storm_summary** – a SAS table that contains one row per storm for the 1980 through 2016 storm seasons

Syntax

```
WHERE expression;

Basic Operators:
  =, EQ
  ^=, ~=, NE
  >, GT
  <, LT
  >=, GE
  <=, LE
  IN(value1, ..., valuen)

SAS Date Constant
  "ddmmmyyyy"d ("01JAN2015"d)
```

Notes

- The WHERE statement is used to filter rows. If the expression is true, rows are read. If the expression is false, they are not.
- Character values are case sensitive and must be enclosed in quotation marks.
- Numeric values are not in quotation marks and must include only digits, decimal points, and negative signs.
- Compound conditions can be created with AND or OR.
- The logic of an operator can be reversed with the NOT keyword.
- When an expression includes a fixed date value, use the SAS date constant syntax: “ddmmmyyyy”d.
 - dd represents a one- or two-digit day
 - mmm represents a three-letter month in uppercase, lowercase, or mixed case
 - yyyy represents a two- or four-digit year

Demo

1. Open **p103d02.sas** from the **demos** folder and find the **Demo** section of the program. Write a PROC PRINT step to list the data in **pg1.storm_summary**.
2. Write a WHERE statement to include rows with **MaxWindMPH** values greater than or equal to 156 (Category 5 storms). Highlight the PROC PRINT step and run the selected code.

```
where MaxWindMPH >= 156;
```

3. Modify the WHERE statement for each of the conditions below. Highlight the PROC PRINT step and run the selected code after each condition.

- a. **Basin** equal to *WP* (West Pacific)

```
where Basin = "WP";
```

- b. **Basin** equal to *SI* or *NI* (South Indian or North Indian)

```
where Basin in ("SI" "NI");
```

- c. **StartDate** on or after January 1, 2010

```
where StartDate >= "01jan2010"d;
```

- d. **Type** equal to *TS* (tropical storm) and **Hem_EW** equal to *W* (west)

```
where Type = "TS" and Hem_EW = "W";
```

- e. **MaxWindMPH** greater than 156 or **MinPressure** less than 920

```
where MaxWindMPH > 156 or MinPressure < 920;
```

4. In the final WHERE statement, are missing values included for **MinPressure**? How can you exclude missing values?

```
where MaxWindMPH>156 or 0<MinPressure<920;
```

End of Demonstration

Using Special WHERE Operators

```
WHERE col-name IS MISSING;  

WHERE col-name IS NOT MISSING;
```

```
where age is missing;  

where name is not missing;
```

These operators work
for both character
and numeric
missing values.



IS NULL is another special operator that can be used with DBMS data. It distinguishes between null and missing values. IS NULL and IS MISSING are the same when they are used with a SAS table.

Using Special WHERE Operators

```
WHERE col-name BETWEEN value-1 AND value-2;
```

```
where age between 20 and 39;
```

includes rows with values
between and including the
endpoints that you specify

For character values,
the range is based
on the alphabet.



Using Special WHERE Operators

`WHERE col-name LIKE "value";`

`where City like "New%" ;`

New York
New Delhi
Newport
Newcastle
New

`where City like "Sant_%" ;`

Santa Clara
Santa Cruz
Santo Domingo
Santo Tomas

wildcard
for any
number of
characters

wildcard
for a single
character

3.02 Activity

Open **p103a02.sas** from the **activities** folder and perform the following tasks:

1. Uncomment each WHERE statement one at a time and run the step to observe the rows that are included in the results.
2. Comment all previous WHERE statements. Add a new WHERE statement to print storms that begin with Z. How many storms are included in the results?

Efficiently Changing the Filter Value

```
proc print data=sashelp.cars;
  where Type="Wagon";
  var Type Make Model MSRP;
run;
proc means data=sashelp.cars;
  where Type="Wagon";
  var MSRP MPG_Highway;
run;
proc freq data=sashelp.cars;
  where Type="Wagon";
  tables Origin Make;
run;
```

Wagon \Rightarrow SUV

How can you
easily replace this
value everywhere
in the program?



```
proc print data=sashelp.cars;
  where Type="SUV";
  var Type Make Model MSRP;
run;
proc means data=sashelp.cars;
  where Type="SUV";
  var MSRP MPG_Highway;
run;
proc freq data=sashelp.cars;
  where Type="SUV";
  tables Origin Make;
run;
```

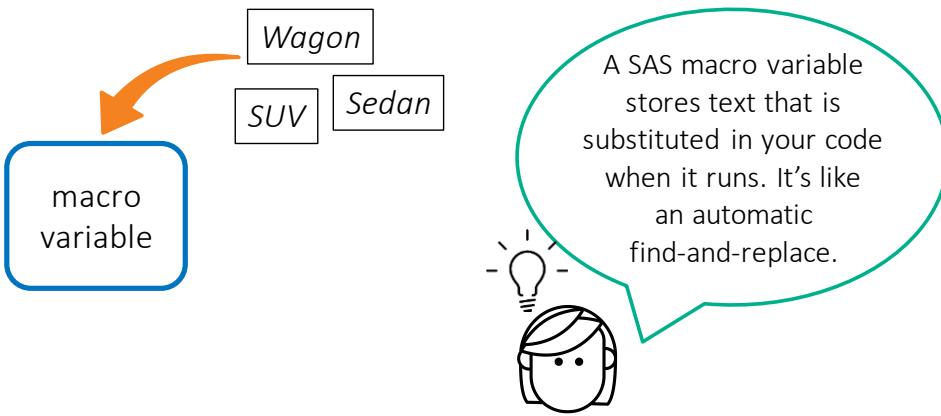
31



Copyright © SAS Institute Inc. All rights reserved.

Suppose you have a program with multiple procedures, and you want to filter each where the value of **Type** is *Wagon*. After you look at the results, you decide that you want similar reports where **Type=SUV** and **Type=Sedan**. Find and replace is an option, but it would be preferable to change that repeating value in one place.

Efficiently Changing the Filter Value



32



The SAS macro language enables you to design dynamic programs that are easy to update or modify. A macro variable enables you to store text and use it in a program.

Creating and Using SAS Macro Variables

create
the
macro
variable

```
%let CarType=Wagon;

proc print data=sashelp.cars;
  where Type="Wagon";
  var Type Make Model MSRP;
run;
proc means data=sashelp.cars;
  where Type="Wagon";
  var MSRP MPG_Highway;
run;
proc freq data=sashelp.cars;
  where Type="Wagon";
  tables Origin Make;
run;
```

%LET macro-variable=value;

creates a macro variable
named **CarType** that
stores the text **Wagon**

33

Copyright © SAS Institute Inc. All rights reserved.

p103d03 

The first step is to create the macro variable, and we do that with the %LET statement. All macro statements begin with a % sign.

Creating and Using SAS Macro Variables

use the
macro
variable

```
%let CarType=Wagon;

proc print data=sashelp.cars;
  where Type="&CarType";
  var Type Make Model MSRP;
run;
proc means data=sashelp.cars;
  where Type="&CarType";
  var MSRP MPG_Highway;
run;
proc freq data=sashelp.cars;
  where Type="&CarType";
  tables Origin Make;
run;
```

¯o-var

Use the macro variable
in place of the value in
the program.

34

Copyright © SAS Institute Inc. All rights reserved.

p103d03 

The next step is to use the macro variable in the program. In each place where *Wagon* is specified, replace it with the macro variable that holds the value *CarType*. To reference a macro variable in a program, precede the name with an ampersand.

Note: It is recommended that you do not include quotation marks when you define the macro variable value. Use quotation marks when necessary after the macro variable is resolved.

Creating and Using SAS Macro Variables

use the
macro
variable

```
%let CarType=Wagon;

proc print data=sashelp.cars;
  where Type="Wagon";
  var Type Make Model MSRP;
run;
proc means data=sashelp.cars;
  where Type="Wagon";
  var MSRP MPG_Highway;
run;
proc freq data=sashelp.cars;
  where Type="Wagon";
  tables Origin Make;
run;
```



SAS replaces
&CarType with
Wagon when the
program runs.

35

Copyright © SAS Institute Inc. All rights reserved.

p103d03 

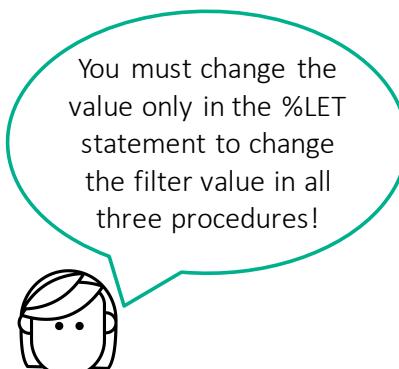
The ampersand triggers SAS to look up the text string stored in the **CarType** macro variable and replace it with *Wagon* before it executes the code.

Creating and Using SAS Macro Variables

use the
macro
variable

```
%let CarType=SUV;

proc print data=sashelp.cars;
  where Type="SUV";
  var Type Make Model MSRP;
run;
proc means data=sashelp.cars;
  where Type="SUV";
  var MSRP MPG_Highway;
run;
proc freq data=sashelp.cars;
  where Type="SUV";
  tables Origin Make;
run;
```



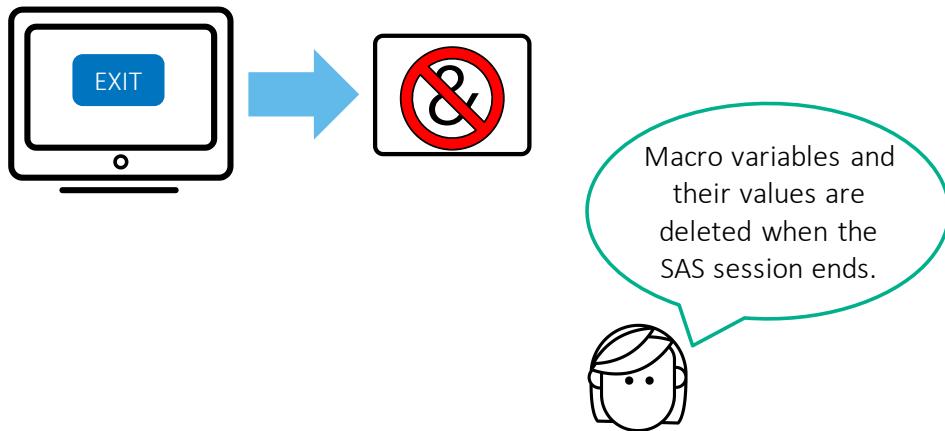
You must change the
value only in the %LET
statement to change
the filter value in all
three procedures!

36

Copyright © SAS Institute Inc. All rights reserved.

p103d03 

Creating and Using SAS Macro Variables



sas

Like libraries, macro variables are temporary, so when your SAS session ends, they are deleted. If macro variable references are included in a program, the macro variables must be created before they are referenced.



Filtering Rows Using Macro Variables

Scenario

Modify a program to use SAS macro variables to filter data in multiple procedures.

Files

- **p103d03.sas**
- **storm_summary** – a SAS table that contains one row per storm for the 1980 through 2016 storm seasons

Syntax

```
%LET macrovar=value;
WHERE numvar=&macrovar;
WHERE charvar="&macrovar";
WHERE datevar="&macrovar"d;
```

Notes

- A macro variable stores a text string that can be substituted into a SAS program.
- The %LET statement defines the macro variable name and assigns a value.
- Macro variable names must follow SAS naming rules.
- Macro variables can be referenced in a program by preceding the macro variable name with an & (ampersand).
- If a macro variable reference is used inside quotation marks, double quotation marks must be used.

Demo

1. Open **p103d03.sas** from the **demos** folder and find the **Demo** section of the program. Highlight the demo program and run the selected code.
2. Write three %LET statements to create macro variables named **WindSpeed**, **BasinCode**, and **Date**. Set the initial values of the variables to match the WHERE statement.
3. Modify the WHERE statement to reference the macro variables. Highlight the demo program and run the selected code. Verify that the same results are produced.

```
%let WindSpeed=156;
%let BasinCode=NA;
%let Date=01JAN2000;

proc print data=pg1.storm_summary;
  where MaxWindMPH=&WindSpeed and Basin="&BasinCode" and
        StartDate="&Date"d;
  var Basin Name StartDate EndDate MaxWindMPH;
run;
```

```

proc means data=pg1.storm_summary;
  where MaxWindMPH=&WindSpeed and Basin=&BasinCode" and
        StartDate=>"&Date"d;
  var MaxWindMPH MinPressure;
run;

```

Obs	Basin	Name	StartDate	EndDate	MaxWindMPH
1946	NA	ISABEL	06SEP2003	20SEP2003	167
2024	NA	IVAN	02SEP2004	24SEP2004	167
2086	NA	EMILY	11JUL2005	21JUL2005	161
2113	NA	KATRINA	23AUG2005	31AUG2005	173
2144	NA	RITA	18SEP2005	26SEP2005	178
2164	NA	WILMA	15OCT2005	26OCT2005	184
2262	NA	DEAN	13AUG2007	23AUG2007	173
2269	NA	FELIX	31AUG2007	06SEP2007	173

The MEANS Procedure					
Variable	N	Mean	Std Dev	Minimum	Maximum
MaxWindMPH	8	172.0000000	7.1113591	161.0000000	184.0000000
MinPressure	8	908.3750000	16.1416719	882.0000000	929.0000000

4. Change the values of the macro variables to values that you select. Possible values for **Basin** include *NA*, *WP*, *SP*, *WP*, *NI*, and *SI*. Highlight the demo program and run the selected code.

End of Demonstration

3.03 Activity

Open **p103a03.sas** from the **activities** folder and perform the following tasks:

1. Change the value in the %LET statement from **NA** to **SP**.

2. Run the program and carefully read the log.

Which procedure did not produce a report?

What is different about the WHERE statement in that step?



Practice

If you restarted your SAS session, open and submit the **libname.sas** program in the course files.

Level 1

4. Filtering Rows in a Listing Report Using Character Data

The **pg1.np_summary** table contains public use statistics from the National Park Service. The park type codes are inconsistent for national preserves. Examine these inconsistencies by producing a report that lists any national preserve.

- a. Open **p103p04.sas** from the **practices** folder. Add a WHERE statement to print only the rows where **ParkName** includes *Preserve*.

Note: **ParkName** contains character values. These values are case sensitive.

- b. Submit the program and view the results. Which codes are used for preserves?

Note: If you use double quotation marks in the WHERE statement, you receive a warning in the log. To eliminate the warning, use single quotation marks.

Obs	Type	ParkName
4	PRE	Yukon-Charley Rivers National Preserve
5	PRE	Bering Land Bridge National Preserve
6	PRESERVE	Noatak National Preserve
58	PRESERVE	Big Thicket National Preserve
74	PRE	Tallgrass Prairie National Preserve
113	PRESERVE	Mojave National Preserve
127	NPRE	Little River Canyon National Preserve
135	PRESERVE	Big Cypress National Preserve

5. Creating a Listing Report for Missing Data

Use PROC PRINT and the WHERE statement to examine the **pg1.eu_occ** table.

- a. Create a new program. Write a PROC PRINT step to read the **pg1.eu_occ** table. Use a WHERE statement to list rows where **Hotel**, **ShortStay**, and **Camp** are missing. Run the program. How many rows are included?
- b. Modify the WHERE statement to list rows with **Hotel** values greater than 40,000,000. Run the program. Which months are included in the report?

Obs	Geo	Country	YearMon	Hotel	ShortStay	Camp
1322	ES	Spain	2017M08	46720017	14976087	10627605
1323	ES	Spain	2017M07	43651610	12905564	8040503
1334	ES	Spain	2016M08	46502956	14449380	10425182
1335	ES	Spain	2016M07	42948946	12349138	7525715
1346	ES	Spain	2015M08	44813404	13742674	9708860
1358	ES	Spain	2014M08	43038904	12906393	9651606

Level 2

6. Using Macro Variables to Subset Data in Procedures

- Create a new program. Write a PROC FREQ step to analyze rows from **pg1.np_species**. Include only rows where **Species_ID** starts with YOSE (Yosemite National Park) and **Category** equals *Mammal*. Generate frequency tables for **Abundance** and **Conservation_Status**.
- Write a PROC PRINT step to list the same subset of rows from **pg1.np_species**. Include **Species_ID**, **Category**, **Scientific_Name**, and **Common_Names** in the report. Run the program.

Abundance	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Abundant	1	7.14	1	7.14
Common	3	21.43	4	28.57
Rare	6	42.86	10	71.43
Uncommon	4	28.57	14	100.00
Frequency Missing = 2				

Conservation_Status	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Species of Concern	3	100.00	3	100.00
Frequency Missing = 13				

Obs	Species_ID	Category	Scientific_Name	Common_Names
17152	YOSE-1003	Mammal	Sus scrofa	Pig, Pig (Feral), Wild Boar, Wild Boar
17153	YOSE-1006	Mammal	Urocyon cinereoargenteus	Gray Fox
17154	YOSE-1007	Mammal	Vulpes vulpes necator	Sierra Nevada Red Fox
17155	YOSE-1014	Mammal	Martes americana	American Marten, Marten
17156	YOSE-1019	Mammal	Taxidea taxus	Badger
17157	YOSE-1023	Mammal	Ursus arctos	Brown Bear, Grizzly Bear
17158	YNOCE-1002	Mammal	Mustela californicus	California Mink

- Create a macro variable named **ParkCode** to store YOSE, and another macro variable named **SpeciesCat** to store *Mammal*. Modify the code to reference the macro variables. Run the program and confirm that the same results are generated.

Note: The macro variable values are case sensitive when they are used in a WHERE statement.

- d. Change the values of the macro variables to ZION (Zion National Park) and Bird. Run the program.

Abundance	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Abundant	1	3.45	1	3.45
Common	8	27.59	9	31.03
Occasional	8	27.59	17	58.62
Rare	4	13.79	21	72.41
Uncommon	8	27.59	29	100.00
Frequency Missing = 17				

Conservation_Status	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Endangered	1	14.29	1	14.29
In Recovery	1	14.29	2	28.57
Species of Concern	5	71.43	7	100.00
Frequency Missing = 39				

Obs	Species_ID	Category	Scientific_Name	Common_Names
17471	ZION-1094	Bird	Haliaeetus leucocephalus	Bald Eagle
17472	ZION-1109	Bird	Aythya collaris	Ring-Necked Duck
17473	ZION-1115	Bird	Clangula hyemalis	Long-Tailed Duck
17474	ZION-1117	Bird	Lophodytes cucullatus	Hooded Merganser
17475	ZION-1129	Bird	Calypte costae	Costa's Hummingbird
17476	ZION-1133	Bird	Selasphorus platycercus	Broad-Tailed Hummingbird
17477	ZION-1125	Bird	Anas diazi	Eastern Migratory Duck

Challenge

7. Eliminating Case Sensitivity in WHERE Conditions

Character comparisons in a WHERE statement are case sensitive. Use SAS functions to make comparisons case insensitive.

- Open pg1.np_traffic. Notice that the case of **Location** values is inconsistent.
 - Create a new program. Write a PROC PRINT step that lists **ParkName**, **Location**, and **Count**. Print rows where **Count** is not equal to 0 and **Location** includes *MAIN ENTRANCE*. Submit the program. Use the log to confirm that 38 rows are listed.
- Note:** If you use double quotation marks in the WHERE statement, you receive a warning in the log. To eliminate the warning, use single quotation marks.
- The UPCASE function can be used to eliminate case sensitivity in character WHERE expressions. Use the UPCASE function on the **Location** column to include any case of *MAIN ENTRANCE*. Run the program and verify that 40 rows are listed.

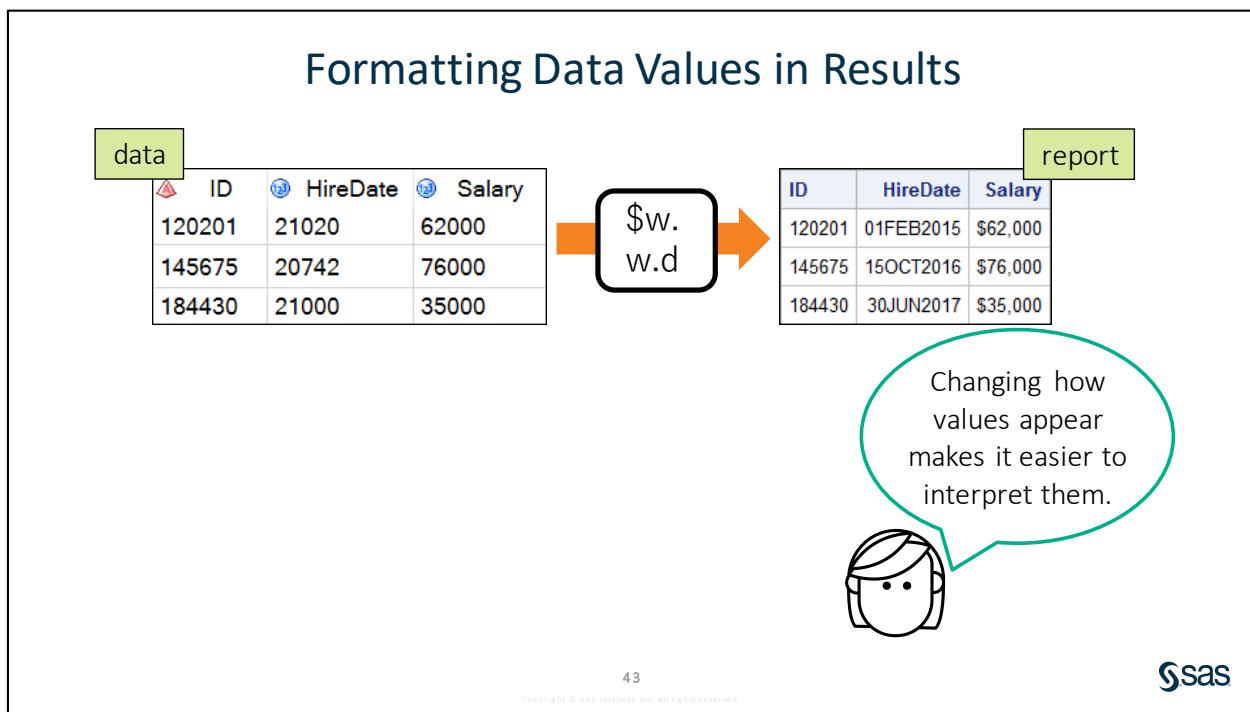
UPCASE(column)

Note: The UPCASE function in a WHERE statement does not permanently convert the values of the column to uppercase.

Obs	ParkName	Location	Count
2	Abraham Lincoln Birthplace National Historical Park	TRAFFIC COUNT AT MAIN ENTRANCE	1,302
8	Allegheny Portage Railroad National Historic Site	Traffic Count at Main Entrance	784
22	Andersonville National Historic Site	TRAFFIC COUNT AT MAIN ENTRANCE	2,146
101	Booker T. Washington National Monument	TRAFFIC COUNT AT MAIN ENTRANCE	364
125	Cabrillo National Monument	TRAFFIC COUNT AT MAIN ENTRANCE	30,994
130	Canyon De Chelly National Monument	TRAFFIC COUNT AT MAIN ENTRANCE	151,500

End of Practices

3.3 Formatting Columns



Sometimes when you are exploring data, it can be difficult to interpret the raw values in the data. For example, it is impossible to visually evaluate SAS date values such as **HireDate** in their raw form. Therefore, in your report, you might want to display the value in a date format that is easy to understand. Numeric columns such as **Salary** store only digits and decimal points, but you might want to display those numbers with commas or currency symbols to make them easier to interpret quickly.

Formatting Data Values in Results

```
PROC PRINT DATA=input-table;
  FORMAT col-name(s) format;
RUN;
```

indicates a character format

`<$>format-name<w>.<d>`

the number of decimal places for numeric formats

total width of the formatted value

All formats include a period.

Common Formats for Numeric Values

Format Name	Example Value	Format Applied	Formatted Value
<code>w.d</code>	12345.67	5.	12346
<code>w.d</code>	12345.67	8.1	12345.7
COMMA <code>w.d</code>	12345.67	COMMA8.1	12,345.7
DOLLAR <code>w.d</code>	12345.67	DOLLAR10.2	\$12,345.67
DOLLAR <code>w.d</code>	12345.67	DOLLAR10.	\$12,346
YEN <code>w.d</code>	12345.67	YEN7.	¥12,346
EUROX <code>w.d</code>	12345.67	EUROX10.2	€12.345,67

Note: International formats just add the symbol to the values. The formats do not convert values from one currency to another.

3.04 Activity

1. Go to support.sas.com/documentation. Click **Programming: SAS 9.4 and Viya**.
2. In the **Syntax - Quick Links** section, under **Language Elements**, select **Formats**.
3. What does the **Zw.d** format do?

46

Copyright © SAS Institute Inc. All rights reserved.



SAS documentation link: <http://support.sas.com/documentation>

Common Formats for Date Values

Value	Format	Formatted Value
21199	DATE7.	15JAN18
21199	DATE9.	15JAN2018
21199	MMDDYY10.	01/15/2018
21199	DDMMYY8.	15/01/18
21199	MONYY7.	JAN2018
21199	MONNAME.	January
21199	WEEKDATE.	Monday, January 15, 2018

48

Copyright © SAS Institute Inc. All rights reserved.



Formatting Multiple Columns

```
proc print data=pg1.class_birthdate;
  format Height Weight 3. Birthdate date9. ;
run;
```

Name	Sex	Age	Height	Weight	Birthdate
Alfred	M	14	69	112.5	16370
Alice	F	13	56.5	84	16756
Barbara	F	13	65.3	98	16451
Carol	F	14	62.8	102.5	16256

Name	Sex	Age	Height	Weight	Birthdate
Alfred	M	14	69	113	26OCT2004
Alice	F	13	57	84	16NOV2005
Barbara	F	13	65	98	15JAN2005
Carol	F	14	63	103	04JUL2004

Here we are printing **class_birthdate**. You can format several columns using either the same format or different formats in a single FORMAT statement. Here we are formatting the columns **height** and **weight** with 3., which rounds the value to the nearest whole number, and we are formatting **Birthdate** with the DATE9. format. These formats impact the way that the values are displayed in the procedure results, but they do not change the raw data values themselves.



Formatting Data Values in Results

Scenario

Use the FORMAT statement in a procedure to display data values as dates and currency.

Files

- **p103d04.sas**
- **storm_damage** – a SAS table that contains a description and damage estimates for storms in the US with damages greater than one billion dollars

Syntax

```
FORMAT col-name format;
<$>format-name<w>.<d>
```

Notes

- Formats are used to change how values are displayed in data and reports.
- Formats do not change the underlying data values.
- Formats can be applied in a procedure using the FORMAT statement.
- Visit [SAS Language Elements documentation](#) to access a list of available SAS formats.

Demo

1. Open **p103d04.sas** from the **demos** folder and find the **Demo** section of the program. Write a PROC PRINT step to list the data in **pg1.storm_damage**. Highlight the step and run the selected code.
2. Add a FORMAT statement to apply the MMDDYY10. format to **Date** and DOLLAR16. to **Cost**. Highlight the step and run the selected code.

```
proc print data=pg1.storm_damage;
  format Date mmddyy10. Cost dollar16. ;
run;
```

Obs	Event	Date	Summary	Cost
1	Hurricane Katrina	08/25/2005	Category 3 hurricane initially impacts the U.S. as a Category 1 near Miami, FL, then as a strong Category 3 along the eastern LA-western MS coastlines, resulting in severe storm surge damage (maximum surge probably exceeded 30 feet) along the LA-MS-AL coasts, wind damage, and the failure of parts of the levee system in New Orleans. Inland effects included high winds and some flooding in the states of AL, MS, FL, TN, KY, IN, OH, and GA.	\$161,300,000,000
2	Hurricane Harvey	08/25/2017	Category 4 hurricane made landfall near Rockport, Texas causing widespread damage. Harvey's devastation was most pronounced due to the large region of extreme rainfall producing historic flooding across Houston and surrounding areas.	\$125,000,000,000

3. Change the width of MMDDYY to **8** and DOLLAR to **14**. Highlight the step and run the selected code. Change MMDDYY to **6** and DOLLAR to **10**. Highlight the step and run the selected code again. What happens to the formatted values?

End of Demonstration

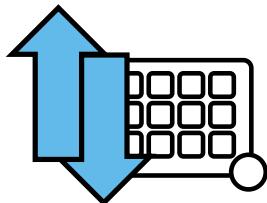
3.05 Activity

Open **p103a05.sas** from the **activities** folder and perform the following tasks:

1. Highlight the PROC PRINT step and run the selected code. Notice how the values of **Lat**, **Lon**, **StartDate**, and **EndDate** are displayed in the report.
2. Change the width of the DATE format to 7 and run the PROC PRINT step. How does the display of **StartDate** and **EndDate** change?
3. Change the width of the DATE format to 11 and run the PROC PRINT step. How does the display of **StartDate** and **EndDate** change?
4. Highlight the PROC FREQ step and run the selected code. Notice that the report includes the number of storms for each **StartDate**.
5. Add a FORMAT statement to apply the MONNAME. format to **StartDate** and run the PROC FREQ step. How many rows are in the report?

3.4 Sorting Data and Removing Duplicates

Sorting Data



improve visual arrangement
of the data

identify and remove
duplicate rows

prepare data for certain
data processing steps

55

Copyright © SAS Institute Inc. All rights reserved.



Sorting data can be a helpful or necessary step in exploring your data. You might want to sort on groups or measures so that you can visually examine the high or low values. You might use sorting as a way to identify and remove duplicate rows. Also, sorting might be required for certain data processing steps.

Sorting Data

```
PROC SORT DATA=input-table <OUT=output-table>;  
  BY <DESCENDING> col-name(s);  
RUN;
```

overrides the default ascending sort order

columns (character, numeric, or both) to sort by

If you don't use the OUT= option, SAS overwrites the input table.



Sas

So how does PROC SORT work? First, SAS rearranges the rows in the input table. Then SAS creates a table that contains the rearranged rows either by replacing the original table or by creating a new table. By default, SAS replaces the original SAS table unless the OUT= option specifies an output table. Keep in mind that PROC SORT does not generate printed output, so you have to open or print the sorted table if you want to look at it.

Similar to PROC PRINT and other procedures, use the DATA= option to specify the input table. Next, use the OUT= option in the PROC SORT statement to prevent permanently sorting the input table. If you do not include the OUT= option, PROC SORT changes the sort order of the input table.

Sorting Data

```
proc sort data=pg1.class_test2 out=test_sort;
  by Name;
run;
```

ascending order
by Name

Name	Subject	TestScore
Alfred	Math	82
Alfred	Reading	79
Alice	Math	71
Alice	Reading	67
Barbara	Math	96
Barbara	Reading	86
Carol	Math	61
Carol	Reading	57

57



Sorting Data

```
proc sort data=pg1.class_test2 out=test_sort;
  by Name TestScore;
run;
```

ascending order
by Name and then
within Name by
ascending TestScore

Name	Subject	TestScore
Alfred	Reading	79
Alfred	Math	82
Alice	Reading	67
Alice	Math	71
Barbara	Reading	86
Barbara	Math	96
Carol	Reading	57
Carol	Math	61

58



Sorting Data

```
proc sort data=pg1.class_test2 out=test_sort;
  by Subject descending TestScore;
run;
```

ascending order
by Subject and then
within Subject by
descending TestScore

Name	Subject	TestScore
Judy	Math	97
Barbara	Math	96
Louise	Math	92
James	Math	90
Joyce	Math	88
William	Math	87
Henry	Math	85
Jane	Math	84

3.06 Activity

Open **p103a06.sas** from the **activities** folder and perform the following tasks:

1. Modify the OUT= option in the PROC SORT statement to create a temporary table named **storm_sort**.
2. Complete the WHERE and BY statements to answer the following question: Which storm in the North Atlantic basin (*NA* or *na*) had the strongest **MaxWindMPH**?

```
PROC SORT DATA=input-table <OUT=output-table>;
  WHERE expression;
  BY <DESCENDING> col-name(s);
RUN;
```

Identifying and Removing Duplicate Rows

```
PROC SORT DATA=input-table <OUT=output-table>
    NODUPKEY <DUPOUT=output-table>;
    BY _ALL_;
    RUN;
```

sorts by all columns to ensure that duplicate rows are adjacent

removes adjacent rows with duplicate BY values

output table of duplicates

62


Copyright © SAS Institute Inc. All rights reserved.

Identifying and Removing Duplicate Rows

```
proc sort data=pg1.class_test3
    out=test_clean
    nodupkey
    dupout=test_dups;
    by _all_;
run;
```

pg1.class_test3

Name	Subject	TestScore
Judy	Math	97
Judy	Reading	91
Barbara	Math	96
Barbara	Reading	86
Barbara	Math	96
Louise	Math	92

test_clean

Name	Subject	TestScore
Alice	Math	71
Alice	Reading	67
Barbara	Math	96
Barbara	Reading	86
Carol	Math	61
Carol	Reading	57

test_dups

Name	Subject	TestScore
Barbara	Math	96

63


Copyright © SAS Institute Inc. All rights reserved.

Identifying and Removing Duplicate Key Values

```
PROC SORT DATA=input-table <OUT=output-table>
  NODUPKEY <DUPOUT=output-table>;
  BY <DESCENDING> col-name(s);
RUN;
```

keeps only the first occurrence of each unique value of the BY variable

This removes duplicate values of the column listed in the BY statement.



Identifying and Removing Duplicate Key Values

```
proc sort data=pg1.class_test2
  out=test_clean
  dupout=test_dups
  nodupkey;
  by Name;
run;
```

Name	Subject	TestScore
Judy	Math	97
Judy	Reading	91
Barbara	Math	96
Barbara	Reading	86
Louise	Math	92
Louise	Reading	99
James	Math	90
James	Reading	85

Name	Subject	TestScore
Alfred	Math	82
Alice	Math	71
Barbara	Math	96
Carol	Math	61
Henry	Math	85
James	Math	90
Jane	Math	84
Janet	Math	75

Name	Subject	TestScore
Alfred	Reading	79
Alice	Reading	67
Barbara	Reading	86
Carol	Reading	57
Henry	Reading	86
James	Reading	85
Jane	Reading	76
Janet	Reading	71



Identifying and Removing Duplicate Values

Scenario

Use the NODUPKEY option in PROC SORT to identify and remove duplicates.

Files

- **p103d05.sas**
- **storm_detail** – a SAS table that contains multiple rows per storm for the 2000 through 2016 storm seasons. Each row represents one measurement for each six hours of a storm.

Syntax

Remove duplicate rows:

```
PROC SORT DATA=input-table <OUT=output-table>
    NODUPKEY <DUPOUT=output-table>;
    BY _ALL_;
    RUN;
```

Remove duplicate key values:

```
PROC SORT DATA=input-table <OUT=output-table>
    NODUPKEY <DUPOUT=output-table>;
    BY <DESCENDING> col-name(s);
    RUN;
```

Notes

- The NODUPKEY option keeps only the first row for each unique value of the column or columns listed in the BY statement.
- Using `_ALL_` in the BY statement sorts by all columns and ensures that duplicate rows are adjacent in the sorted table and are removed.
- The DUPOUT= option creates an output table in which the duplicates are removed.

Demo

1. Open **p103d05.sas** from the **demos** folder and find the **Demo** section of the program. Modify the first PROC SORT step to sort by all columns and remove any duplicate rows. Write the removed rows to a table named **storm_dups**. Highlight the step and run the selected code. Confirm that there are 50,757 rows in **storm_clean** and 7 rows in **storm_dups**.

```
proc sort data=pg1.storm_detail out=storm_clean
    nodupkey dupout=storm_dups;
    by _all_;
run;
```

2. The second PROC SORT step is filtering for nonmissing values of **Name** and **Pressure** and then sorting by descending **Season**, **Basin**, **Name**, and **Pressure**. Run the second PROC SORT step and confirm that the first row for each storm represents the minimum value of **Pressure**.

Note: Because storm names can be reused in multiple years and basins, unique storms are grouped by sorting by **Season**, **Basin**, and **Name**.

3. Modify the third PROC SORT step to sort the **min_pressure** table from the previous PROC SORT step, and keep the first row for each storm. You do not need to keep the removed duplicates. Highlight the step and run the selected code.

```
proc sort data=min_pressure nodupkey;
  by descending Season Basin Name;
run;
```

Season	Basin	Sub_basin	Name	ISO_time	Type	Latitude	Longitude	Wind	Pressure	Hem_NS	Hem_EW	Region
2016	EP	MM	AGATHA	03JUL2016:06...	TS	16.8	-122.1	45	1002	N	W	Pacific
2016	EP	MM	BLAS	06JUL2016:00...	TS	14.3	-121.2	120	947	N	W	Pacific
2016	EP	MM	CELIA	11JUL2016:18...	TS	15.1	-125.8	85	972	N	W	Pacific
2016	EP	MM	DARBY	16JUL2016:18...	TS	17.9	-124.4	105	958	N	W	Pacific
2016	EP	MM	ESTELLE	17JUL2016:18...	TS	16.5	-112.2	60	990	N	W	Pacific
2016	EP	MM	FRANK	27JUL2016:00...	TS	21.6	-118.2	75	979	N	W	Pacific
2016	EP	MM	GEORGETTE	25 JUL 2016:06	TS	16.6	-126.4	115	952	N	W	Pacific

End of Demonstration

Beyond SAS Programming 1

What if you want to ...

... discover other procedures for exploring your data?

- Visit the [SAS 9.4 Procedures Help page](#).
- Browse or ask questions in the [SAS Procedures community](#) and see responses from other SAS programmers.

... dive deeper into the SAS macro language?

- Take the [SAS Macro 1](#) course.
- Read the [SAS Macro Programming Made Easy](#) book.

... create custom formats based on your data?

- Learn about [PROC FORMAT in SAS Help](#).
- Take the [SAS Programming 2](#) course.

67
Copyright © SAS Institute Inc. All rights reserved.



Links

- Visit the [SAS 9.4 Procedures Help page](#).
- Browse or ask questions in the [SAS Procedures community](#) and see responses from other SAS programmers.
- Take the [SAS Macro 1](#) course.
- Read the [SAS Macro Programming Made Easy](#) book.
- Learn about [PROC FORMAT in SAS Help](#).
- Take the [SAS Programming 2](#) course.



Practice

If you restarted your SAS session, open and submit the **libname.sas** program in the course files.

Level 1

8. Sorting Data and Creating an Output Table

Create the **np_sort** table that contains data for national parks ordered by regional code and decreasing numbers of daily visitors.

- Open **p103p08.sas** from the **practices** folder. Modify the PROC SORT step to read **pg1.np_summary** and create a temporary sorted table named **np_sort**.
- Add a BY statement to order the data by **Reg** and descending **DayVisits**.
- Add a WHERE statement to select **Type** equal to **NP**. Submit the program.

	Reg	Type	ParkName	DayVisits	OtherLodging	OtherCamping	TentCa
1	A	NP	Kenai Fjords National Park	346,534	0	0	
2	A	NP	Kobuk Valley National Park	15,500	0	0	
3	IM	NP	Grand Canyon National Park	5,969,811	600,307	24,257	20
4	IM	NP	Rocky Mountain National Park	4,517,585	0	0	12
5	IM	NP	Zion National Park	4,295,127	86,456	0	13
6	IM	NP	Yellowstone National Park	4,257,177	579,227	583,068	10
7	IM	NP	Grand Teton National Park	3,270,076	196,577	297,084	
8	IM	NP	Glacier National Park	2,946,681	99,550	0	11

Level 2

9. Sorting Data to Remove Duplicate Rows

The **pg1.np_largenparks** table contains gross acreage for large national parks. There are duplicate rows for some locations.

- Open and review the **pg1.np_largenparks** table. Notice that there are exact duplicate rows for some parks.
- Create a new program. Write a PROC SORT step that creates two tables (**park_clean** and **park_dups**), and removes the duplicate rows. Submit the program.

park_clean

	UnitCode	AreaName	State	Reg	GrossAcres
1	ACAD	ACADIA NP	ME	NE	49057.36
2	AMIS	AMISTAD NRA	TX	IM	58500.00
3	APIS	APOSTLE ISLANDS NL	WI	MW	69377.43
4	ARCH	ARCHEES NP	UT	IM	76678.98
5	ASIS	ASSATEAGUE ISLAND NS	MD-VA 2/	NE	41346.50
6	BADL	BADLANDS NP	SD	MW	242755.94
7	PAND	PANDELLER NM	NM	IM	33676.67

park_dups

	UnitCode	AreaName	State	Reg	GrossAcres
1	ASIS	ASSATEAGUE ISLAND NS	MD-VA 2/	NE	41346.50
2	BLCA	BLACK CANYON OF GUNNISON NP	CO	IM	30749.75
3	BLRI	BLUE RIDGE PKWY	NC-VA	NT	98774.04
4	BRCA	BRYCE CANYON NP	UT	IM	35835.08
5	BUFF	BUFFALO N RVR	AR	MW	94293.31
6	CACO	CAPE COD NS	MA	NE	43608.43
7	CALO	CAPE LOOKOUT NC	NC	SC	20242.20

Challenge**10. Creating a Lookup Table from a Detailed Table**

The **pg1.eu_occ** table includes multiple rows from each country code and country name. Create a lookup table that includes a single row for each country code and name.

- a. Create a new program. Write a PROC SORT step to sort **pg1.eu_occ** and create an output table named **countrylist**. Remove duplicate key values. Sort by **Geo** and then **Country**.
- b. To read only **Geo** and **Country** from the **pg1.eu_occ** table, you can use the KEEP= data set option. Add the KEEP= option immediately after the input table and list **Geo** and **Country**.

data-set (KEEP=varlist)

- c. Run the program and verify that only one row per country is included.

	Geo	Country
1	AT	Austria
2	BE	Belgium
3	BG	Bulgaria
4	CY	Cyprus
5	CZ	Czech Republic
6	DE	Germany
7	DK	Denmark

End of Practices

3.5 Solutions

Solutions to Practices

1. Exploring Data with Procedures

```

/*Parts A and B*/
/*list first 20 rows*/
proc print data=pg1.np_summary(obs=20);
  var Reg Type ParkName DayVisits TentCampers RVCampers;
run;

/*Part C*/
/*calculate summary statistics*/
proc means data=pg1.np_summary;
  var DayVisits TentCampers RVCampers;
run;

/*Part D*/
/*examine extreme values*/
proc univariate data=pg1.np_summary;
  var DayVisits TentCampers RVCampers;
run;

/*Part E*/
/*list unique values and frequency counts*/
proc freq data=pg1.np_summary;
  tables Reg Type;
run;

```

- b. Do you observe any possible inconsistencies in the data?

Yes. The Type column has inconsistencies. Notice that national preserve locations have the code PRES and PRESERVE.

- c. What is the minimum value for tent campers? Is that value unexpected?

The minimum value is zero. No, because it is possible that a park had zero tent campers.

- d. Are there negative values for any of the columns?

No

- e. Are there any lowercase codes? Are there any codes that occur only once in the table?

There are no lowercase codes. NC, NPRE, and RIVERWAYS occur once in the table.

2. Using Procedures to Validate Data

```
*Part A;
proc freq data=pg1.np_summary;
  tables Reg Type;
run;

*Part B;
proc univariate data=pg1.np_summary;
  var Acres;
run;
```

- a. What invalid values exist for **Reg**? None

What invalid values exist for **Type**? **NPRE**, **PRESERVE**, **RIVERWAYS**

- c. What are the smallest and largest parks? **Observation 78 (African Burial Ground Monument, .35 acres)** and **observation 6 (Noatak National Preserve, 6,587,071.39 acres)**

3. Generating Extreme Observations Output

```
*Part A and B;
ods trace on;
proc univariate data=pg1.eu_occ;
  var camp;
run;
ods trace off;

*Part D and E;
ods select extremeobs;
proc univariate data=pg1.eu_occ nextrobs=10;
  var camp;
run;
```

4. Filtering Rows in a Listing Report Using Character Data

```
proc print data=pg1.np_summary;
  var Type ParkName;
  where ParkName like '%Preserve%';
run;
```

5. Creating a Listing Report for Missing Data

```
*Part A;
proc print data=pg1.eu_occ;
  where Hotel is missing and ShortStay is missing and
        Camp is missing;
run;

*Part B;
proc print data=pg1.eu_occ;
  where Hotel > 40000000;
run;
```

- a. How many rows are included? **101**

- b. Which months are included in the report? **The months are July or August.**

6. Using Macro Variables to Subset Data in Procedures

```
%let ParkCode=ZION;
%let SpeciesCat=Bird;

proc freq data=pg1.np_species;
  tables Abundance Conservation_Status;
  where Species_ID like "&ParkCode%" and
        Category="&SpeciesCat";
run;

proc print data=pg1.np_species;
  var Species_ID Category Scientific_Name Common_Names;
  where Species_ID like "&ParkCode%" and
        Category="&SpeciesCat";
run;
```

7. Eliminating Case Sensitivity in WHERE Conditions

```
proc print data=pg1.np_traffic;
  var ParkName Location Count;
  where Count ne 0 and upcase(Location) like '%MAIN ENTRANCE%';
run;
```

8. Sorting Data and Creating an Output Table

```
proc sort data=pg1.np_summary out=np_sort;
  by Reg descending DayVisits;
  where Type="NP";
run;
```

9. Sorting Data to Remove Duplicate Rows

```
proc sort data=pg1.np_largeparks
  out=park_clean
  dupout=park_dups
  nodupkey;
  by _all_;
run;
```

10. Creating a Lookup Table from a Detailed Table

```
proc sort data=pg1.eu_occ(keep=geo country) out=countryList
  nodupkey;
  by geo Country;
run;
```

End of Solutions

Solutions to Activities and Questions

3.01 Multiple Choice Question – Correct Answer

Which statement in PROC PRINT selects variables that appear in the report and determines their order?

- a. BY
- b. ID
- c. SUM
- d. VAR

Table of Procedure Tasks and Examples	
Statement	Task
PROC PRINT	Print observations in a data set
BY	Produce a separate section of the report for each BY group
ID	Identify observations by the formatted values of the variables that you list instead of by observation numbers
PAGEBY	Control page ejects that occur before a page is full
SUMBY	Limit the number of sums that appear in the report
SUM	Total values of numeric variables
VAR	Select variables that appear in the report and determine their order

8



3.02 Activity – Correct Answer

Add a new WHERE statement to print storms that begin with Z. How many storms are included?

```
proc print data=pg1.storm_summary(obs=50);
  commented where statements
  where name like "Z%";
run;
```

NOTE: There were 24 observations read from the data set PG1.STORM_SUMMARY.
WHERE name like 'Z%';

30



3.03 Activity – Correct Answer

Which procedure did not produce a report? PROC FREQ

What is different about the WHERE statement in that step?

Single quotation marks were used around the macro variable &BasinCode rather than double quotation marks.

```

48      proc freq data=pg1.storm_summary;
49          where Basin='&BasinCode';
50          tables Type;
51      run;

NOTE: No observations were selected from data
      set PG1.STORM_SUMMARY.
NOTE: There were 0 observations read from the
      data set PG1.STORM_SUMMARY.
WHERE 0 /* an obviously FALSE WHERE clause */ ;

```

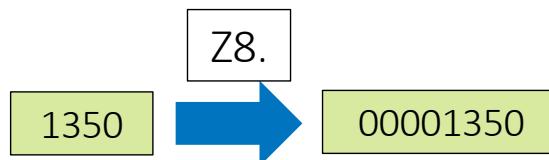
Double quotation marks must be used around macro variables.



3.04 Activity – Correct Answer

1. Go to support.sas.com/documentation. Click Programming: SAS 9.4 and Viya.
2. In the Syntax - Quick Links section, under Language Elements, select Formats.
3. What does the Z.w.d format do?

The format displays standard numeric data with leading zeros.



continued...

3.05 Activity – Correct Answer

2. Change the width of the DATE format to 7 and run the PROC PRINT step. How does the display of **StartDate** and **EndDate** change?
3. Change the width of the DATE format to 11 and run the PROC PRINT step. How does the display of **StartDate** and **EndDate** change?

StartDate	EndDate
17JUL80	18NOV80
27MAR80	30MAR80
09JUN80	15JUN80
27NOV79	06DEC79
09OCT80	14OCT80

StartDate	EndDate
17-JUL-1980	18-NOV-1980
27-MAR-1980	30-MAR-1980
09-JUN-1980	15-JUN-1980
27-NOV-1979	06-DEC-1979
09-OCT-1980	14-OCT-1980

DATE7. displays a two-digit year.

DATE11. displays a four-digit year and adds dashes.

52

Copyright © SAS Institute Inc. All rights reserved.



3.05 Activity – Correct Answer

5. Add a FORMAT statement to apply the MONNAME. format to **StartDate** and run the PROC FREQ step. How many rows are in the report?

```
proc freq data=pg1.storm_summary order=freq;
  tables StartDate;
  format startdate monname.;

run;
```

The FREQ Procedure				
StartDate	Frequency	Percent	Cumulative Frequency	Cumulative Percent
August	483	15.49	483	15.49
September	482	15.46	965	30.95
July	351	11.26	1316	42.21
October	324	10.39	1640	52.60
January	250	8.02	1890	60.62
February	224	7.18	2114	67.80
June	198	6.35	2312	74.15
November	187	6.00	2499	80.15
December	183	5.87	2682	86.02
March	181	5.81	2863	91.82
May	131	4.20	2994	96.02
April	124	3.98	3118	100.00

The new report has 12 rows.

Formats are an easy way to group data in procedures!



Copyright © SAS Institute Inc. All rights reserved.

53

3.06 Activity – Correct Answer

Open **p103a06.sas** from the **activities** folder and perform the following tasks:

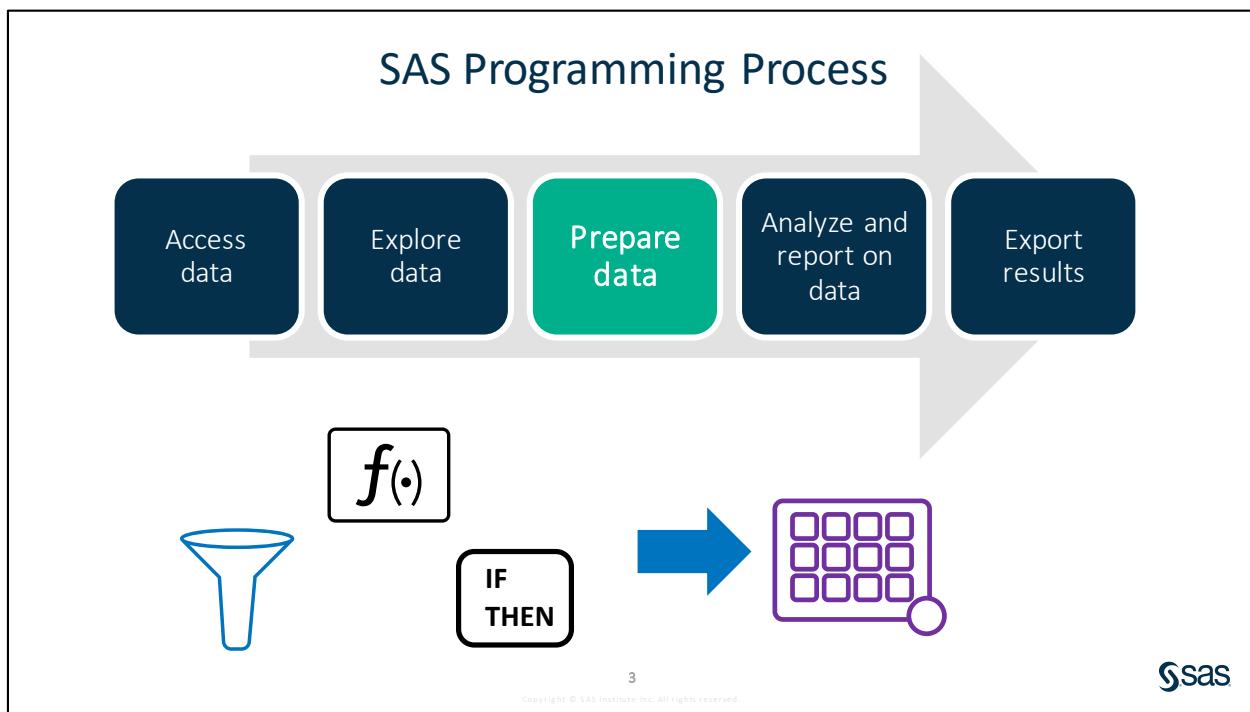
1. Modify the OUT= option in the PROC SORT statement to create a temporary table named **storm_sort**.
2. Complete the WHERE and BY statements to answer the following question: Which storm in the North Atlantic Basin (**NA** or **na**) had the strongest **MaxWindMPH**? Allen

```
proc sort data=pg1.storm_summary out=storm_sort;
  where Basin in("NA" "na");
  by descending MaxWindMPH;
run;
```

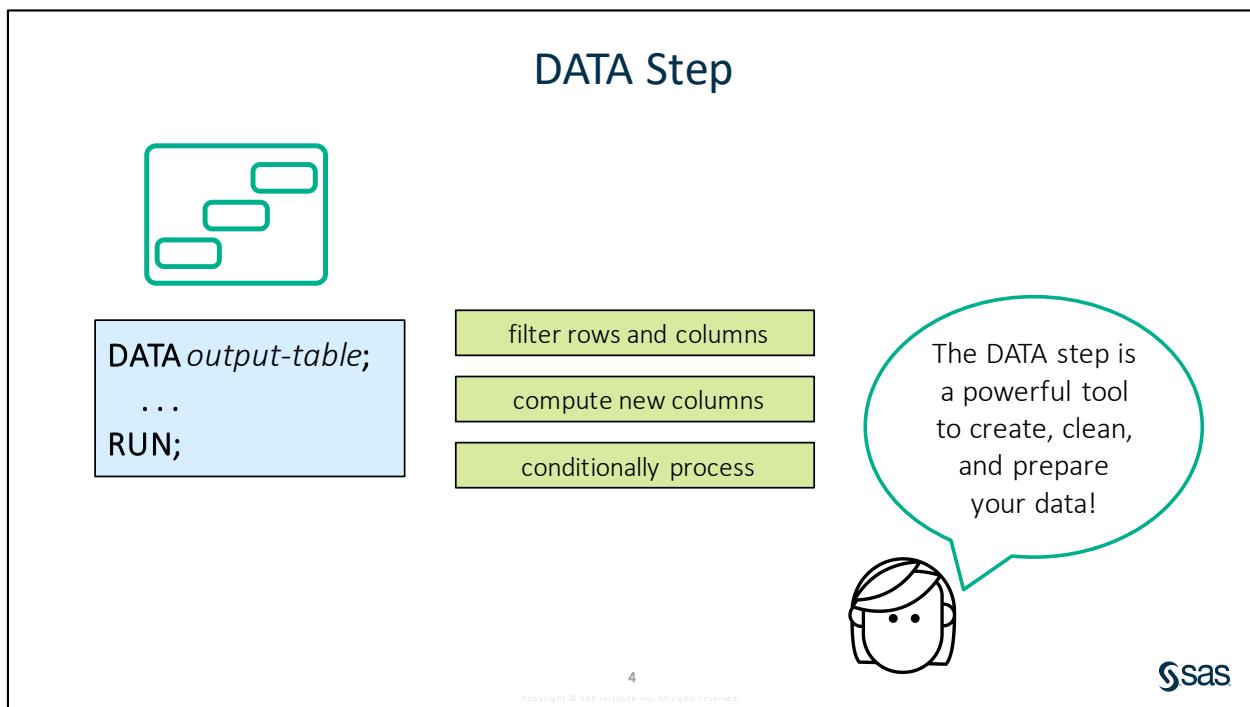
Lesson 4 Preparing Data

4.1	Reading and Filtering Data.....	4-3
	Practice.....	4-12
4.2	Computing New Columns	4-14
	Demonstration: Using Expressions to Create New Columns.....	4-16
	Demonstration: Using Character Functions.....	4-22
	Demonstration: Using Date Functions.....	4-26
	Practice.....	4-28
4.3	Conditional Processing	4-30
	Demonstration: Conditional Processing with IF-THEN.....	4-31
	Demonstration: Processing Multiple Statements with IF-THEN/DO.....	4-42
	Practice.....	4-46
4.4	Solutions	4-48
	Solutions to Practices	4-48
	Solutions to Activities and Questions.....	4-51

4.1 Reading and Filtering Data

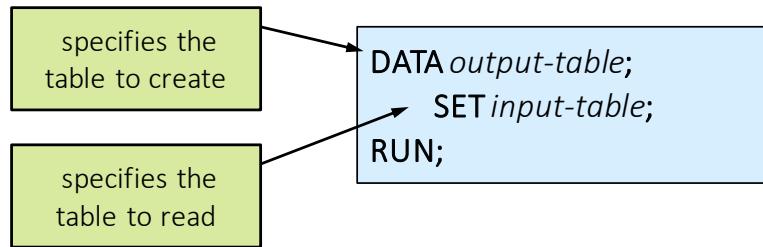


After you explore your data, you likely want to make some adjustments based on what you find and what you need. In this lesson, you learn various ways to subset data, and you use expressions and functions to compute new columns. You also learn how to use conditional processing to obtain the results that you want in your output data.



The DATA step is a robust, yet simple programming tool that can do everything from simple querying to providing structure to messy weblogs. In this class, you become familiar with the most common data manipulation actions, such as filtering rows and columns, computing new columns, and performing conditional processing. Beyond these features, the DATA step also enables you to merge or join tables, read complex raw data, and perform repetitive processing with DO loops or arrays. These topics and many others are covered in [SAS Programming 2: Data Manipulation Techniques](#) and [other advanced programming courses](#).

Using the DATA Step to Create a SAS Data Set



When you work with data, you want to preserve your existing data and create a copy that you can work on, so let's start with a simple DATA step that does just that.

- The DATA statement names the table that you want to create, or the *output table*. This can be a temporary table if you use the **Work** library or a permanent table if you use any other library. Be aware that if the table you list in the DATA statement exists and you have Write access to it, the DATA step overwrites that table.
- The SET statement names the existing table that you are reading from, or the *input table*. When I reference a data source as *libref.table*, then based on a previous LIBNAME statement, SAS knows where to find the data source and how to read it.
- The DATA step ends with a RUN statement.

Using the DATA Step to Create a SAS Data Set

creates the table
myclass in the
Work library

reads the table
class in the
Sashelp library

```
data myclass;
  set sashelp.class;
run;
```

	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Alice	F	13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84
10	John	M	12	59	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90
13	Louise	F	12	56.3	77
14	Mary	F	15	66.5	112
15	Philip	M	16	72	150
16	Robert	M	12	64.8	128
17	Ronald	M	15	67	133
18	Thomas	M	11	57.5	85
19	William	M	15	66.5	112

6

Copyright © SAS Institute Inc. All rights reserved.



DATA Step Processing

Compilation

- Check syntax for errors.
- Identify column attributes.
- Establish new table metadata.

Execution

- Read and write data.
- Perform data manipulations, calculations, and so on.



What happens
behind the
scenes when a
DATA step runs?

7

Copyright © SAS Institute Inc. All rights reserved.



How does the DATA step work behind the scenes? In this course, you need to have only a high-level understanding of the process. The DATA step has two phases: compilation and execution. In the compilation phase, SAS checks for syntax errors in the program and establishes the table metadata, such as column name, type, and length. In the execution phase, the data is read, processed, and written one row at time.

DATA Step Processing

Execution

- 1) Read a row from the input table.
- 2) Sequentially process statements.
- 3) At the end, write the row to the output table.
- 4) Loop back to the top of the DATA step to read the next row from the input table.

```
data myclass;  
  set sashelp.class;  
  ...other statements...  
run;
```



Automatic
looping makes
processing
data easy!

sas

8

Copyright © SAS Institute Inc. All rights reserved.

DATA step execution is like an automatic loop. The first time through the DATA step, the SET statement reads the first row from the input table and then processes any other statements in sequence, manipulating the values within that row. When SAS reaches the RUN statement, there is an implied OUTPUT action, and the new row is written to the output table. The DATA step then automatically loops back to the top and executes the statements in order again, this time reading, manipulating, and outputting the second row. That implicit loop continues until all rows are read from the input table.

As you learn more about the DATA step, it is helpful to have a deep understanding of this behind-the-scenes processing. The SAS Programming 2: Data Manipulation Techniques course addresses more complex DATA step code and covers the details of the compile and execute phases.

4.01 Activity

Open **p104a01.sas** from the **activities** folder and perform the following tasks:

1. Complete the DATA step to create a temporary table named **storm_new** and read **pg1.storm_summary**. Run the program and read the log.
2. Define a library named **out** pointing to the **output** folder in the main course files folder.
3. Change the program to save a permanent version of **storm_new** in the **out** library. Run the modified program.

```
LIBNAME libref"path";
DATA output-table;
    SET input-table;
RUN;
```

Keep this program open for the next activity.

9



4.02 Multiple Answer Question

The table listed in the SET statement must be read via a library. Which data sources can be used in the SET statement?

- a. SAS tables
- b. Excel spreadsheets
- c. DBMS tables
- d. comma-delimited files

11



Filtering Rows in the DATA Step

```
DATA output-table;
  SET input-table;
  WHERE expression;
  RUN;
```

filters rows based on the expression

The DATA step reads rows only from the input table where the *expression* is true.



13

Copyright © SAS Institute Inc. All rights reserved.

p104d01 

The same WHERE syntax that works in a procedure to subset data for a report or analysis works in the DATA step to filter rows. Only those rows from the input table that meet the criteria in the WHERE statement are processed by the DATA step and written to the output table.

Filtering Rows in the DATA Step

```
data myclass;
  set sashelp.class;
  where age >= 15;
run;
```

	Name	Sex	Age	Height	Weight
Janet	F		15	62.5	112.5
Mary	F		15	66.5	112
Philip	M		16	72	150
Ronald	M		15	67	133
William	M		15	66.5	112

NOTE: There were 5 observations read from the data set SASHHELP.CLASS.

WHERE age>= 15;

NOTE: The data set WORK.MYCLASS has 5 observations and 5 variables.

14

Copyright © SAS Institute Inc. All rights reserved.

p104d01 

Subsetting Columns in the DATA Step

`DROP col-name <col-name>;`

`KEEP col-name <col-name>;`

Choose the statement based on the number of columns that you want to specify.



15

Copyright © SAS Institute Inc. All rights reserved.

p104d01

To specify the columns to include in the output data, use either the DROP statement or the KEEP statement followed by the column names from the input table to drop or keep.

Subsetting Columns in the DATA Step

These statements have the same result in the output table.

```
data myclass;
  set sashelp.class;
  keep name age height;
  drop sex weight;
run;
```

	Name	Age	Height
Alfred	14	69	
Alice	13	56.5	
Barbara	13	65.3	
Carol	14	62.8	
Henry	14	63.5	

16

Copyright © SAS Institute Inc. All rights reserved.

p104d01

4.03 Activity

Modify the program that you opened in the previous activity or open **p104a03.sas** from the **activities** folder and perform the following tasks:

1. Change the name of the output table to **storm_cat5**.
2. Include only Category 5 storms (**MaxWindMPH** greater than or equal to 156) with **StartDate** on or after 01JAN2000.
3. Add a statement to include the following columns in the output data: **Season**, **Basin**, **Name**, **Type**, and **MaxWindMPH**. How many Category 5 storms occurred since January 1, 2000?

Formatting Columns in the DATA Step

```
DATA output-table;
  SET input-table;
  FORMAT col-name format;
  RUN;
```

name of the column that you want to format

name of the format that you want to apply

Formats in the DATA step are permanently assigned to the columns.



The **FORMAT** statement is used in procedures to change how data values are displayed in a report or analysis.

We can use the same **FORMAT** statement in the DATA step, but the impact is a little different. A **FORMAT** statement in the DATA step **permanently** assigns a format to a column in the properties of the new table. The raw data values are still stored in the table, but anytime you view the data or use it in procedures, the formats are automatically applied.

Formatting Columns in the DATA Step

```
data myclass;
  set sashelp.class;
  format height 4.1 weight 3.;
run;
```

rounds values of height
to one decimal place
and weight to the
nearest whole number

sashelp.class

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5

myclass

Name	Sex	Age	Height	Weight
Alfred	M	14	69.0	113
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	103

20

Copyright © SAS Institute Inc. All rights reserved.

p104d01





Practice

If you restarted your SAS session, open and submit the **libname.sas** program in the course files.

Level 1

1. Creating a SAS Table

The **pg1.eu_occ** SAS table contains monthly occupancy rates for European countries from January 2004 through September 2017.

- Open the **pg1.eu_occ** table and examine the column names and values.
- Open **p104p01.sas** from the **practices** folder. Modify the code to create a temporary table named **eu_occ2016** and read **pg1.eu_occ**.
- Complete the WHERE statement to select only the stays that were reported in 2016. Notice that **YearMon** is a character column and the first four positions represent the year.
- Complete the FORMAT statement in the DATA step to apply the COMMA17. format to the **Hotel**, **ShortStay**, and **Camp** columns.
- Complete the DROP statement to exclude **Geo** from the output table.

	Country	YearMon	Hotel	ShortStay	Camp
1	Austria	2016M12	6,670,483	1,468,847	117,579
2	Austria	2016M11	3,600,616	681,867	28,303
3	Austria	2016M10	5,727,389	985,402	146,108
4	Austria	2016M09	7,726,801	1,443,829	620,032
5	Austria	2016M08	11,399,594	3,022,261	1,897,979
6	Austria	2016M07	9,996,416	2,633,484	1,608,971
7	Austria	2016M06	6,444,485	1,287,244	569,242
8	Austria	2016M05	5,619,330	1,118,179	445,622

Level 2

2. Creating a Permanent SAS Table

The **np_species** table includes one row for each species that is found in each national park.

- Create a new program. Write a DATA step to read the **pg1.np_species** table and create a new permanent table named **fox**. Write the new table to the **output** folder.
- Include only the rows where **Category** is *Mammal* and **Common_Names** includes *Fox*.
- Exclude the **Category**, **Record_Status**, **Occurrence**, and **Nativeness** columns. Run the program.
- Notice that *Fox Squirrels* are included in the output table. Add a condition in the WHERE statement to exclude rows that include *Squirrel*.

- e. Sort the **fox** table by **Common_Names**.

	Species_ID	Family	Scientific_Name	Common_Names	Abundance	Seasonality	Conservation_Status
1	GAAR-1004	Canidae	Alopex lagopus	Arctic Fox	Unknown		
2	CHIS-1000	Canidae	Urocyon littoralis	Channel Islands Gray Fox	Rare	Breeder	
3	BLCA-1005	Canidae	Urocyon cinereoargenteus	Common Gray Fox	Rare		
4	CARE-1006	Canidae	Urocyon cinereoargenteus	Common Gray Fox	Common	Breeder	
5	CONG-1004	Canidae	Urocyon cinereoargenteus	Common Gray Fox	Common	Breeder	
6	YOSE-1006	Canidae	Urocyon cinereoargenteus	Gray Fox	Common	Breeder	

Challenge

3. Creating a SAS Table Using Macro Variables

The **np_species** table includes one row for each species that is found in each national park.

- Write a new program that creates a temporary table named **Mammal** that includes only the mammals from the **pg1,np_species** table. Do not include **Abundance**, **Seasonality**, or **Conservation_Status** in the output table.
- Use PROC FREQ to determine how many species there are for each unique value of **Record_Status**.

Record_Status	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Approved	561	90.63	561	90.63
In Review	58	9.37	619	100.00

- Modify the program to use a macro variable to change **Mammal** to other values of **Category**. Change the macro variable value to **Bird** and run the program.

Note: Use PROC FREQ to determine the unique values of **Category**.

Record_Status	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Approved	2060	96.22	2060	96.22
In Review	81	3.78	2141	100.00

End of Practices

4.2 Computing New Columns

Using Expressions to Create New Columns

assignment statement

```
DATA output-table;  
  SET input-table;  
  new-column = expression;  
RUN;
```

arithmetic expression
or constant

The assignment
statement can
create or update
a column.



sas

23

Copyright © SAS Institute Inc. All rights reserved.

Often your data does not have **all** the columns that you need, and you might want to calculate or derive new columns from existing columns. Fortunately, this is easy to do in the DATA step. To create new columns, you use an assignment statement. You simply type the name of the new column, an equal sign, and then the expression that creates a new data value.

Using Expressions to Create New Columns

```
data cars_new;
  set sashelp.cars;
  where Origin ne "USA";
  Profit = MSRP-Invoice;
  Source = "Non-US Cars";
  format Profit dollar10.;
  keep Make Model MSRP Invoice Profit Source;
run;
```

Make	Model	MSRP	Invoice	Profit	Source
Acura	MDX	\$36,945	\$33,337	\$3,608	Non-US Cars
Acura	RSX Type S 2dr	\$23,820	\$21,761	\$2,059	Non-US Cars
Acura	TSX 4dr	\$26,990	\$24,647	\$2,343	Non-US Cars
Acura	TL 4dr	\$33,195	\$30,299	\$2,896	Non-US Cars
Acura	3.5 RL 4dr	\$43,755	\$39,014	\$4,741	Non-US Cars
Acura	3.5 RL w/Navi...	\$46,100	\$41,100	\$5,000	Non-US Cars
Acura	NSX coupe 2d...	\$89,765	\$79,978	\$9,787	Non-US Cars

The column name
is stored in the
case that you use
to create it.



p104d02

In this example, the WHERE statement includes rows where **Origin** is not equal to **USA**. The first assignment statement creates the new column **Profit** using a simple arithmetic expression. SAS creates the numeric column **Profit** and generates a value for every row in the output table by subtracting **Invoice** from **MSRP**. The second assignment statement creates a column named **Source** and assigns the character string *Non-US Cars*. Notice that because there is a KEEP statement, you must explicitly list the new columns so that they are included in the **cars_new** table.



Using Expressions to Create New Columns

Scenario

Read an existing SAS table, and create temporary and permanent copies.

Files

- **p104d02.sas**
- **storm_summary** – a SAS table that contains one row per storm for the 1980 through 2016 storm seasons

Syntax

```
DATA output-table;
  SET input-table;
  new-column = expression;
RUN;
```

Notes

- The name of the column to be created or updated is listed on the left side of the equal sign.
- Provide an expression on the right side of the equal sign.
- SAS automatically defines the required attributes (name, type, and length) if the column is new.
- A new numeric column has a length of 8.
- The length of a new character column is determined based on the length of the assigned string.
- Character strings must be enclosed in quotation marks and are case sensitive.

Demo

1. Open **p104d02.sas** from the **demos** folder and find the **Demo** section of the program. Add an assignment statement to create a numeric column named **MaxWindKM** by multiplying **MaxWindMPH** by 1.60934.
2. Add a FORMAT statement to round **MaxWindKM** to the nearest whole number.
3. Add an assignment statement to create a new character column named **StormType** that is equal to *Tropical Storm*. Highlight the DATA step and run the selected code.

```
data tropical_storm;
  set pg1.storm_summary;
  drop Hem_EW Hem_NS Lat Lon;
  where Type="TS";
  *Add assignment and FORMAT statements;
  MaxWindKM=MaxWindMPH*1.60934;
  format MaxWindKM 3. ;
  StormType="Tropical Storm";
run;
```

	Season	Name	Basin	Type	MaxWindMPH	MinPressure	StartDate	EndDate	MaxWindKM	StormType
1	1980	na	TS		35	.	17JUL1980	18NOV1980	56	Tropical Storm
2	1980	AGATHA	EP	TS	115	.	09JUN1980	15JUN1980	185	Tropical Storm
3	1980	ALEX	WP	TS	40	998	09OCT1980	14OCT1980	64	Tropical Storm
4	1980	ALLEN	NA	TS	190	899	31JUL1980	11AUG1980	306	Tropical Storm
5	1980	BERENICE	SI	TS	.	.	15DEC1979	21DEC1979	.	Tropical Storm
6	1980	BLAS	EP	TS	58	.	16JUN1980	19JUN1980	93	Tropical Storm
7	1980	CARMEN	WP	TS	69	985	05APR1980	07APR1980	111	Tropical Storm
8	1980	CARLY	WP	TS	52	996	28OCT1980	02NOV1980	94	Tropical Storm

End of Demonstration

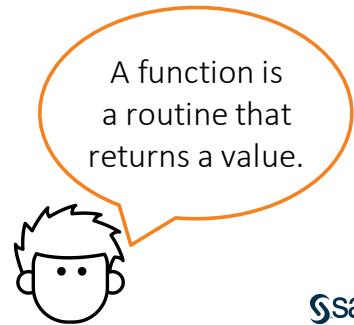
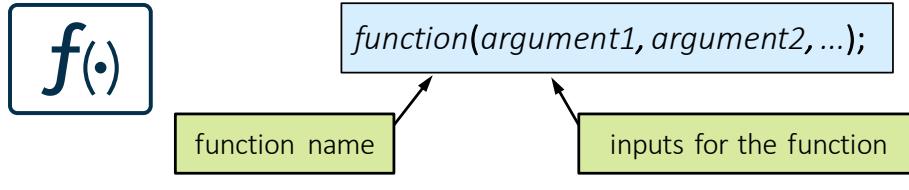
4.04 Activity

Open **p104a04.sas** from the **activities** folder and perform the following tasks:

1. Add an assignment statement to create **StormLength** that represents the number of days between **StartDate** and **EndDate**.
2. Run the program. In 1980, how long did the storm named Agatha last?

```
data storm_length;
  set pg1.storm_summary;
  drop Hem_EW Hem_NS Lat Lon;
  *Add assignment statement;
run;
```

Functions



Arithmetic calculations and character constants are a good start for creating new columns, but often you need more elaborate or flexible methods for generating the new data values. SAS offers hundreds of functions that can be used in countless ways to manipulate numeric, character, and date values.

The syntax for a function is the function name, followed by the arguments enclosed in parentheses. The arguments are separated by commas. The arguments consist of the input that the function needs to perform its specific routine and return a value.

Functions

$f(\cdot)$

```
DATA output-table;
  SET input-table;
  new-column=function(arguments);
RUN;
```

Functions can be used
in an assignment
statement to create
or update a column.

29


Copyright © SAS Institute Inc. All rights reserved.

Numeric Functions

Functions
SUM (num1, num2, ...)
MEAN (num1, num2, ...)
MEDIAN (num1, num2, ...)
RANGE (num1, num2, ...)
MIN (num1, num2, ...)
MAX (num1, num2, ...)
N (num1, num2, ...)
NMISS (num1, num2, ...)

These functions
ignore
missing values
in the data.



30



SAS has a collection of summary statistics functions, including SUM, MEAN, MEDIAN, and RANGE. Each of these functions can have an unlimited number of arguments, and each argument provides either a numeric constant or numeric column in the data. The function calculates the summary statistic from the values of the arguments for each row in the data. One interesting note about these summary functions is that if any of the input values are missing, the missing value or values are ignored, and the calculation is based on the known values.

Numeric Functions

```
data cars_new;
  set sashelp.cars;
  MPG_Mean=mean(MPG_City, MPG_Highway);
  format MPG_Mean 4.1;
  keep Make Model MPG_City MPG_Highway MPG_Mean;
run;
```

The MEAN function calculates an average for each row.



Make	Model	MPG_City	MPG_Highway	MPG_Mean
Acura	MDX	17	23	20.0
Acura	RSX Type S 2dr	24	31	27.5
Acura	TSX 4dr	22	29	25.5
Acura	TL 4dr	20	28	24.0
Acura	3.5 RL 4dr	18	24	21.0
Acura	3.5 RL w/Navi...	18	24	21.0
Acura	NSX coupe 2d...	17	24	20.5
Audi	A4 1.8T 4dr	22	31	26.5

31

Copyright © SAS Institute Inc. All rights reserved.

p104d03



In this example code, an assignment statement creates a column named **MPG_Mean**. The MEAN function is used with the arguments **MPG_City** and **MPG_Highway** to supply values for **MPG_Mean**. Notice that the FORMAT statement rounds the displayed values of **MPG_Mean** to one decimal place.

4.05 Activity

Open p104a05.sas from the activities folder and perform the following tasks:

1. Open the **pg1.storm_range** table and examine the columns. Notice that each storm has four wind speed measurements.
2. Create a new column named **WindAvg** that is the mean of **Wind1**, **Wind2**, **Wind3**, and **Wind4**.
3. Create a new column **WindRange** that is the range of **Wind1**, **Wind2**, **Wind3**, and **Wind4**.

```
data storm_windavg;
  set pg1.storm_range;
  *Add assignment statements;
run;
```

32

Copyright © SAS Institute Inc. All rights reserved.



Character Functions

Function	What It Does
UPCASE (char) LOWCASE (char)	Changes letters in a character string to uppercase or lowercase
PROPCASE (char, <delimiters>)	Changes the first letter of each word to uppercase and other letters to lowercase
CATS (char1, char2, ...)	Concatenates character strings and removes leading and trailing blanks from each argument
SUBSTR (char, position, <length>)	Returns a substring from a character string

Note: The default delimiters for the PROPCASE function are a blank, forward slash, hyphen, open parenthesis, period, and tab. To use a different list of delimiters, specify a list of characters in a single set of quotation marks as the second argument in the function.

Character Functions

```
data cars_new;
  set sashelp.cars;
  Type=upcase(Type);
  keep Make Model Type;
run;
```

Type is an existing column.

Make	Model	Type
Acura	MDX	SUV
Acura	RSX Type S 2dr	SEDAN
Acura	TSX 4dr	SEDAN
Acura	TL 4dr	SEDAN
Acura	3.5 RL 4dr	SEDAN
Acura	3.5 RL w/Navigation 4dr	SEDAN
Acura	NSX coupe 2dr manual S	SPORTS
Audi	A4 1.8T 4dr	SEDAN

As a simple example, let's look at the UPCASE function. It requires one argument: a character column. The UPCASE function returns the uppercase equivalent of the input data values. In this case, we are not creating a new column in the output data. We are converting the values in the **Type** column to uppercase in the **cars_new** data.



Using Character Functions

Scenario

Use character functions to manipulate existing character values.

Files

- **p104d03.sas**
- **storm_summary** – a SAS table that contains one row per storm for the 1980 through 2016 storm seasons

Syntax

```
UPCASE(char)
PROPCASE(char, <delimiters>)
CATS(char1, char2, ...)
SUBSTR(char, position, <length>)
```

Notes

- The UPCASE function converts character values to uppercase.
- The PROPCASE function changes the first letter of each word to uppercase and other letters to lowercase.
- The CATS function concatenates character values and removes any leading or trailing blanks.
- The SUBSTR function extracts a string from a character value.

Demo

1. Open **p104d03.sas** from the **demos** folder and find the **Demo** section of the program. Add an assignment statement to convert **Basin** to all uppercase letters using the UPCASE function.
2. Add an assignment statement to convert **Name** to proper case using the PROPCASE function.
3. Add an assignment statement to create **Hemisphere**, which concatenates **Hem_NS** and **Hem_EW** using the CATS function.
4. Add an assignment statement to create **Ocean**, which extracts the second letter of **Basin** using the SUBSTR function. Highlight the DATA step and run the selected code.

```
data storm_new;
  set pg1.storm_summary;
  drop Type Hem_EW Hem_NS MinPressure Lat Lon;
  *Add assignment statements;
  Basin=upcase(Basin) ;
  Name=propcase(Name) ;
  Hemisphere=cats(Hem_NS, Hem_EW) ;
  Ocean=substr(Basin,2,1) ;
run;
```

	Season	Name	Basin	MaxWindMPH	StartDate	EndDate	Hemisphere	Ocean
1	1980		NA	35	17JUL1980	18NOV1980	NW	A
2	1980		SP	.	27MAR1980	30MAR1980	SE	P
3	1980	Agatha	EP	115	09JUN1980	15JUN1980	NW	P
4	1980	Albine	SI	.	27NOV1979	06DEC1979	SE	I
5	1980	Alex	WP	40	09OCT1980	14OCT1980	NE	P
6	1980	Allen	NA	190	31JUL1980	11AUG1980	NW	A
7	1980	Amy	SI	132	04JAN1980	12JAN1980	SE	I
8	1980	Perenice	SI		15DEC1979	21DEC1979	SE	I

End of Demonstration

4.06 Activity

Open **p104a06.sas** from the **activities** folder and perform the following tasks:

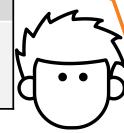
1. Add a WHERE statement that uses the SUBSTR function to include rows where the second letter of **Basin** is **P** (Pacific ocean storms).
2. Run the program and view the log and data. How many storms were in the Pacific basin?

```
data pacific;
  set pg1.storm_summary;
  drop Type Hem_EW Hem_NS MinPressure Lat Lon;
  *Add a WHERE statement that uses the SUBSTR function;
run;
```

Date Functions

Function	What It Does
MONTH (SAS-date)	Returns a number from 1 through 12 that represents the month
YEAR (SAS-date)	Returns the four-digit year
DAY (SAS-date)	Returns a number from 1 through 31 that represents the day of the month
WEEKDAY (SAS-date)	Returns a number from 1 through 7 that represents the day of the week (Sunday=1)
QTR (SAS-date)	Returns a number from 1 through 4 that represents the quarter

These functions extract information from SAS date values.



Date Functions

Function	What It Does
TODAY()	Returns the current date as a numeric SAS date value
MDY (<i>month, day, year</i>)	Returns a SAS date value from month, day, and year values
YRDIF (<i>startdate, enddate, 'AGE'</i>)	Calculates a precise difference in years between two dates

40

Copyright © SAS Institute Inc. All rights reserved.



Note: The optional third argument in the YRDIF function is called the *basis*. The basis value describes how SAS calculates a date difference or a person's age. When calculating the age of a person or event, 'AGE' should be used as the basis. Visit [the SAS documentation for the YRDIF function](#) to learn about other values for the basis.



Using Date Functions

Scenario

Use date functions to manipulate existing date values.

Files

- **p104d04.sas**
- **storm_damage** – a SAS table that contains a description and damage estimates for storms in the US with damages greater than one billion dollars

Syntax

```

YEAR(SAS-date)
MONTH(SAS-date)
DAY(SAS-date)
WEEKDAY(SAS-date)

TODAY()
MDY(month, day, year)
YRDIF(startdate, enddate, 'AGE')

```

Notes

- The YEAR, MONTH, DAY, and WEEKDAY functions return a numeric value. For WEEKDAY, 1 represents Sunday.
- The TODAY function returns the current date based on the system clock as a SAS date value.
- The MDY function creates a SAS date based on numeric month, day, and year values.
- The YRDIF function calculates a precise age between two dates. There are various values for the third argument. However, 'AGE' should be used for accuracy.

Demo

1. Open **p104d04.sas** from the **demos** folder and find the **Demo** section of the program. Create the column **YearsPassed** and use the YRDIF function. The difference in years should be based on each **Date** value and today's date.
2. Create **Anniversary** as the day and month of each storm in the current year.
3. Format **YearsPassed** to round the value to one decimal place, and **Date** and **Anniversary** as MM/DD/YYYY. Highlight the DATA step and run the selected code.

```

data storm_damage2;
  set pg1.storm_damage;
  drop Summary;
  *Add assignment and FORMAT statements;
  YearsPassed=yrdif(Date,today(),'age');
  Anniversary=mdy(month(Date),day(Date),year(today()));
  format YearsPassed 4.1 Date Anniversary mmddyy10.;
run;

```

Note: Values for **YearsPassed** and **Anniversary** will be different based on the current date.

	Event	Date	Cost	YearsPassed	Anniversary
1	Hurricane Katrina	08/25/2005	161300000000	14.3	08/25/2019
2	Hurricane Harvey	08/25/2017	125000000000	2.3	08/25/2019
3	Hurricane Maria	09/19/2017	90000000000	2.2	09/19/2019
4	Hurricane Sandy	10/30/2012	70900000000	7.1	10/30/2019
5	Hurricane Irma	09/06/2017	50000000000	2.2	09/06/2019
6	Hurricane Andrew	08/23/1992	48300000000	27.3	08/23/2019
7	Hurricane Ike	09/12/2008	35100000000	11.2	09/12/2019
8	Hurricane Lili	09/12/2004	27200000000	15.2	09/12/2019

End of Demonstration



Practice

If you restarted your SAS session, open and submit the **libname.sas** program in the course files.

Level 1

4. Creating New Columns

Create a new table named **np_summary_update** from **pg1,np_summary**. Create two new columns: **SqMiles** and **Camping**.

- Open **p104p04.sas** from the **practices** folder. Create a new column named **SqMiles** by multiplying **Acres** by .0015625.
- Create a new column named **Camping** as the sum of **OtherCamping**, **TentCampers**, **RVCampers**, and **BackcountryCampers**.
- Format **SqMiles** and **Camping** to include commas and zero decimal places.
- Modify the **KEEP** statement to include the new columns. Run the program.

	Reg	ParkName	DayVisits	OtherLodging	Acres	SqMiles	Camping
1	A	Cape Krusenstern National Monument	15,000	0	649,096.15	1,014	6,375
2	A	Kenai Fjords National Park	346,534	0	669,650.05	1,046	2,162
3	A	Kobuk Valley National Park	15,500	0	1,750,716.16	2,735	7,050
4	A	Yukon-Charley Rivers National Preserve	1,146	0	2,523,512.44	3,943	3,063
5	A	Bering Land Bridge National Preserve	2,642	0	2,697,391.01	4,215	1,123
6	A	Noatak National Preserve	17,000	0	6,587,071.39	10,292	5,500
7	IM	Alibates Flint Quarries National Monument	8,153	0	1,370.97	2	0

Level 2

5. Creating New Columns with Character and Date Functions

The **pg1.eu_occ** table contains individual columns for nights spent at hotels, short stay accommodations, or camps for each year and month. The **YearMon** column is character.

- Open a new program. Write a DATA step to create a temporary table named **eu_occ_total** based on the **pg1.eu_occ** table. Create the following new columns:
 - **Year** – the four-digit year extracted from **YearMon**.
 - **Month** – the two-digit month extracted from **YearMon**.
 - **ReportDate** – the first day of the reporting month.

Note: Use the MDY function and the new **Year** and **Month** columns.

 - **Total** – the total nights spent at any establishment. Format the new column to display the values with commas.

- b. Format **Hotel**, **ShortStay**, **Camp**, and **Total** with commas. Format **ReportDate** to display the values in the form JAN2018.
- c. Keep **Country**, **Hotel**, **ShortStay**, **Camp**, **ReportDate**, and **Total** in the new table.

	Country	Hotel	ShortStay	Camp	ReportDate	Total
1	Austria	7,768,564	1,453,530	524,121	SEP2017	9,746,215
2	Austria	11,353,432	3,140,217	1,997,801	AUG2017	16,491,450
3	Austria	10,124,106	2,836,425	1,752,605	JUL2017	14,713,136
4	Austria	7,391,827	1,568,683	914,560	JUN2017	9,875,070
5	Austria	5,068,884	1,054,870	359,560	MAY2017	6,483,314
6	Austria	5,647,811	1,360,315	171,094	APR2017	7,179,220
7	Austria	8,666,740	2,534,986	97,576	MAR2017	11,299,302
8	Austria	10,052,766	2,098,249	127,907	FEB2017	12,285,022

Challenge

6. Creating a New Column with the SCAN Function

- a. Access SAS Help to learn about the SCAN function.
- b. Create a new program. Create a new temporary table named **np_summary2** based on the **pg1.np_summary** table. Use the SCAN function to create a new column named **ParkType** that is the last word of the **ParkName** column.

Note: Use a negative number for the second argument to count words from right to left in the character string.

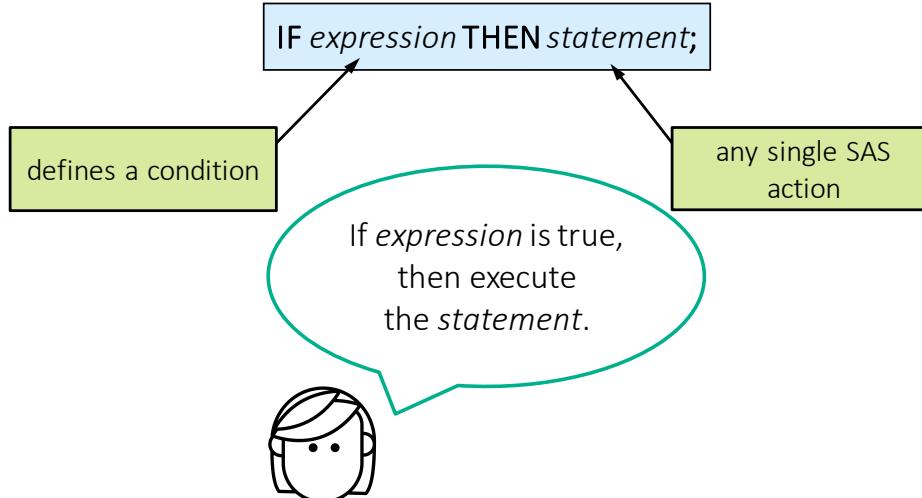
- c. Keep **Reg**, **Type**, **ParkName**, and **ParkType** in the output table.

	Reg	Type	ParkName	ParkType
1	A	NM	Cape Krusenstern National Monument	Monument
2	A	NP	Kenai Fjords National Park	Park
3	A	NP	Kobuk Valley National Park	Park
4	A	PRE	Yukon-Charley Rivers National Preserve	Preserve
5	A	PRE	Bering Land Bridge National Preserve	Preserve
6	A	PRESERVE	Noatak National Preserve	Preserve
7	IM	NM	Alibates Flint Quarries National Monument	Monument

End of Practices

4.3 Conditional Processing

Conditional Processing with IF-THEN



4.4



Copyright © SAS Institute Inc. All rights reserved.

Often in the DATA step, we need to process data conditionally. In other words, if some condition is met, then execute one statement. If a different condition is met, then execute another statement. We can accomplish this using IF-THEN logic.

Conditional Processing with IF-THEN

```
data cars2;
  set sashelp.cars;
  if MSRP<30000 then Cost_Group=1;
  if MSRP>=30000 then Cost_Group=2;
  keep Make Model Type MSRP Cost_Group;
run;
```

Make	Model	Type	MSRP	Cost_Group
Acura	MDX	SUV	\$36,945	2
Acura	RSX Type S 2dr	Sedan	\$23,820	1
Acura	TSX 4dr	Sedan	\$26,990	1
Acura	TL 4dr	Sedan	\$33,195	2
Acura	3.5 RL 4dr	Sedan	\$43,755	2
Acura	3.5 RL w/Navi...	Sedan	\$46,100	2
Acura	NSX coupe 2d...	Sports	\$89,765	2
Audi	A4 1.8T 4dr	Sedan	\$25,940	1

4.5

p104d05





Conditional Processing with IF-THEN

Scenario

Use IF-THEN syntax to assign values conditionally to a new column.

Files

- **p104d05.sas**
- **storm_summary** – a SAS table that contains one row per storm for the 1980 through 2016 storm seasons

Syntax

IF expression THEN statement;

Notes

- The expression following IF defines a condition that is evaluated as true or false for each row.
- If the condition is true, the statement following THEN is executed.
- Only one statement is permitted after THEN.

Demo

1. Open **p104d05.sas** from the **demos** folder and find the **Demo** section of the program. Create a column named **PressureGroup** that is based on the following assignments:

MinPressure≤920 ⇒ 1

MinPressure>920 ⇒ 0

```
data storm_new;
  set pg1.storm_summary;
  keep Season Name Basin MinPressure PressureGroup;
  *Add IF-THEN statements;
  if MinPressure<=920 then PressureGroup=1;
  if MinPressure>920 then PressureGroup=0;
run;
```

2. Highlight the DATA step, run the selected code, and examine the data. What value is assigned to **PressureGroup** when **MinPressure** is missing?
3. Add a new IF-THEN statement **before** the existing IF-THEN statements to assign **PressureGroup=.** if **MinPressure** is missing.

```
data storm_new;
  set pg1.storm_summary;
  keep Season Name Basin MinPressure PressureGroup;
  *Add IF-THEN statements;
  if MinPressure=. then PressureGroup=.;
  if MinPressure<=920 then PressureGroup=1;
  if MinPressure>920 then PressureGroup=0;
run;
```

4. Highlight the DATA step and run the selected code. What value is assigned to PressureGroup?

When MinPressure is missing, the first two IF conditions are true. The last assignment statement determines the value of PressureGroup.

	Season	Name	Basin	MinPressure	PressureGroup
1	1980		na	.	1
2	1980		SP	998	0
3	1980	AGATHA	EP	.	1
4	1980	ALBINE	SI	.	1
5	1980	ALEX	WP	998	0
6	1980	ALLEN	NA	899	1
7	1980	AMY	SI	915	1
8	1980	BERENICE	SI	.	1

End of Demonstration

Conditional Processing with IF-THEN/ELSE

```
IF expression THEN statement;
<ELSE IF expression THEN statement;>
<ELSE IF expression THEN statement;>
ELSE statement;
```

If the others are not true, execute this statement.

If *expression* is true,
then execute
the *statement* and
skip the rest.



47

Copyright © SAS Institute Inc. All rights reserved.



When you have multiple IF-THEN statements, SAS tests all conditions in sequence for every row of the data. The last true condition executes the statement that determines the value in the output table. Suppose you want to treat these conditions as a hierarchy so that when a true condition is found, SAS simply executes the statement following THEN and skips the subsequent IF statements. If you want to enforce this type of sequential testing, be sure to use the keyword ELSE.

Conditional Processing with IF-THEN/ELSE

```
data cars2;
  set sashelp.cars;
  if MSRP<20000 then Cost_Group=1;
  else if MSRP<40000 then Cost_Group=2;
  else if MSRP<60000 then Cost_Group=3;
  else Cost_Group=4;
  keep Make Model Type MSRP Cost_Group;
run;
```

48

Copyright © SAS Institute Inc. All rights reserved.

p104d06



The keyword ELSE is not in the first statement, but it has been added in the three statements that follow. This tells SAS to test the conditions only until a true expression is found.

Conditional Processing with IF-THEN/ELSE

Example: MSRP=35000

```
data cars2;
  set sashelp.cars;
  if MSRP<20000 then Cost_Group=1;
  else if MSRP<40000 then Cost_Group=2;
  else if MSRP<60000 then Cost_Group=3;
  else Cost_Group=4;
  keep Make Model Type MSRP Cost_Group;
run;
```

49

Copyright © SAS Institute Inc. All rights reserved.

p104d06



Let's look at an example where **MSRP** is equal to 35000. The first IF-THEN statement is false, so SAS moves to the next statement.

Conditional Processing with IF-THEN/ELSE

Example: MSRP=35000

```
data cars2;
  set sashelp.cars;
  if MSRP<20000 then Cost_Group=1;
  else if MSRP<40000 then Cost_Group=2;
  else if MSRP<60000 then Cost_Group=3;
  else Cost_Group=4;
  keep Make Model Type MSRP Cost_Group;
run;
```

true

execute

50

Copyright © SAS Institute Inc. All rights reserved.

p104d06



The second condition is true, so SAS assigns the value 2 to **Cost_Group**.

Conditional Processing with IF-THEN/ELSE

Example: MSRP=35000

```
data cars2;
  set sashelp.cars;
  if MSRP<20000 then Cost_Group=1;
  else if MSRP<40000 then Cost_Group=2;
  else if MSRP<60000 then Cost_Group=3;
  else Cost_Group=4;
  keep Make Model Type MSRP Cost_Group;
run;
```

skip

51

Copyright © SAS Institute Inc. All rights reserved.

p104d06



SAS skips the rest of the conditional processing statements.

Conditional Processing with IF-THEN/ELSE

Example: MSRP=75000

```
data cars2;
  set sashelp.cars;
  if MSRP<20000 then Cost_Group=1;
  else if MSRP<40000 then Cost_Group=2;
  else if MSRP<60000 then Cost_Group=3;
  else Cost_Group=4;
  keep Make Model Type MSRP Cost_Group;
run;
```

false

execute

The final ELSE statement
executes if all previous
conditions were false.

52

Copyright © SAS Institute Inc. All rights reserved.

p104d06



For a row with **MSRP** equal to 75000, none of the stated **MSRP** conditions are true, so the last assignment statement is executed. Notice in this final ELSE statement that there is no condition, just an assignment statement. There is no reason to test that final condition because if the preceding conditions are all false, we know **Cost_Group** should be 4.

4.07 Activity

Open **p104a07.sas** from the **activities** folder and perform the following tasks:

1. Add the **ELSE** keyword to test conditions sequentially until a true condition is met.
2. Change the final IF-THEN statement to an ELSE statement.
3. How many storms are in **PressureGroup** 1?

Creating Character Columns with IF-THEN/ELSE

```
data cars2;
  set sashelp.cars;
  if MSRP<60000 then CarType="Basic";
  else CarType="Luxury";
  keep Make Model MSRP CarType;
run;
```

Based on the value of **MSRP**, assign a value to the new character column **CarType**.

Make	Model	MSRP	CarType
Acura	MDX	\$36,945	Basic
Acura	RSX Type S 2dr	\$23,820	Basic
Acura	TSX 4dr	\$26,990	Basic
Acura	TL 4dr	\$33,195	Basic
Acura	3.5 RL 4dr	\$43,755	Basic
Acura	3.5 RL w/Navi...	\$46,100	Basic
Acura	NSX coupe 2d...	\$89,765	Luxur
Audi	A4 1.8T 4dr	\$25,940	Basic

Creating Character Columns with IF-THEN/ELSE

```
data cars2;
  set sashelp.cars;
  if MSRP<60000 then CarType="Basic";
  else CarType="Luxury";
  keep Make Model MSRP CarType;
run;
```

creates a new character column with a length of 5

Make	Model	MSRP	CarType
Acura	MDX	\$36,945	Basic
Acura	RSX Type S 2dr	\$23,820	Basic
Acura	TSX 4dr	\$26,990	Basic
Acura	TL 4dr	\$33,195	Basic
Acura	3.5 RL 4dr	\$43,755	Basic
Acura	3.5 RL w/Navi...	\$46,100	Basic
Acura	NSX coupe 2d...	\$89,765	Luxur
Audi	A4 1.8T 4dr	\$25,940	Basic

The first mention of a column in the DATA step defines the name, type, and length.



p104d06

It is important to know that the first occurrence of a column in the DATA step defines the name, type, and length of the column. So, if you have an assignment statement that defines a character column and assigns the value *Basic*, the column is created with a length of 5, the number of characters in the word *Basic*. You can see from the output that *Luxury* is truncated because it has six characters.

Creating Character Columns with IF-THEN/ELSE

LENGTH char-column \$ length;

number of bytes or characters

column name in the case that you specify

indicator for a character column

Using the LENGTH statement leaves nothing to chance.



p104d06

One way to avoid this problem is to explicitly define a character column in the DATA step with a LENGTH statement. The syntax for this statement is the keyword LENGTH followed by the name of the column, a dollar sign to indicate a character column, and the length that you want to assign.

Creating Character Columns with IF-THEN/ELSE

```
data cars2;
  set sashelp.cars;
  length CarType $ 6;
  if MSRP<60000 then CarType="Basic";
  else CarType="Luxury";
  keep Make Model MSRP CarType;
run;
```

explicitly creates
a new character column
with a length of 6

Make	Model	MSRP	CarType
Acura	MDX	\$36,945	Basic
Acura	RSX Type S 2dr	\$23,820	Basic
Acura	TSX 4dr	\$26,990	Basic
Acura	TL 4dr	\$33,195	Basic
Acura	3.5 RL 4dr	\$43,755	Basic
Acura	3.5 RL w/Navi...	\$46,100	Basic
Acura	NSX coupe 2d...	\$89,765	Luxury
Audi	A4 1.8T 4dr	\$25,940	Basic

58

Copyright © SAS Institute Inc. All rights reserved.

p104d06



4.08 Activity

Open **p104a08.sas** from the **activities** folder and perform the following tasks:

- Run the program and examine the results. Why is **Ocean** truncated? What value is assigned when **Basin='na'**?
- Modify the program to add a LENGTH statement to declare the name, type, and length of **Ocean** before the column is created.

LENGTH char-column \$ length;

- Add an assignment statement after the KEEP statement to convert **Basin** to uppercase. Run the program.
- Move the LENGTH statement to the end of the DATA step. Run the program. Does it matter where the LENGTH statement is in the DATA step?

59

Copyright © SAS Institute Inc. All rights reserved.



Using Compound Conditions with IF-THEN/ELSE

```
data cars2;
  set sashelp.cars;
  if MPG_City>26 and MPG_Highway>30 then Efficiency=1;
  else if MPG_City>20 and MPG_Highway>25 then Efficiency=2;
  else Efficiency=3;
  keep Make Model MPG_City MPG_Highway Efficiency;
run;
```

AND

Both conditions must be true.

OR

One condition must be true.

Make	Model	MPG_City	MPG_Highway	Efficiency
Chevrolet	Tracker	19	22	3
Chevrolet	Aveo 4dr	28	34	1
Chevrolet	Aveo LS 4dr hatch	28	34	1
Chevrolet	Cavalier 2dr	26	37	2
Chevrolet	Cavalier 4dr	26	37	2

62

Copyright © SAS Institute Inc. All rights reserved.



Processing Multiple Statements

```
data cars2;
  set sashelp.cars;
  length Cost_Type $ 4;
  if MSRP<20000 then Cost_Group=1 and Cost_Type="Low";
  else if MSRP<40000 then Cost_Group=2 and Cost_Type="Mid";
  else Cost_Group=3 and Cost_Type="High";
run;
```

Compound statements are not allowed.

This program doesn't work because only one statement is permitted after THEN.



63

Copyright © SAS Institute Inc. All rights reserved.



If you can specify a compound condition to evaluate, can you do the same after the keyword THEN to execute multiple statements? If you attempt to use AND between two statements, the program fails with a syntax error because you are allowed only one executable statement following THEN.

Processing Multiple Statements with IF-THEN/DO

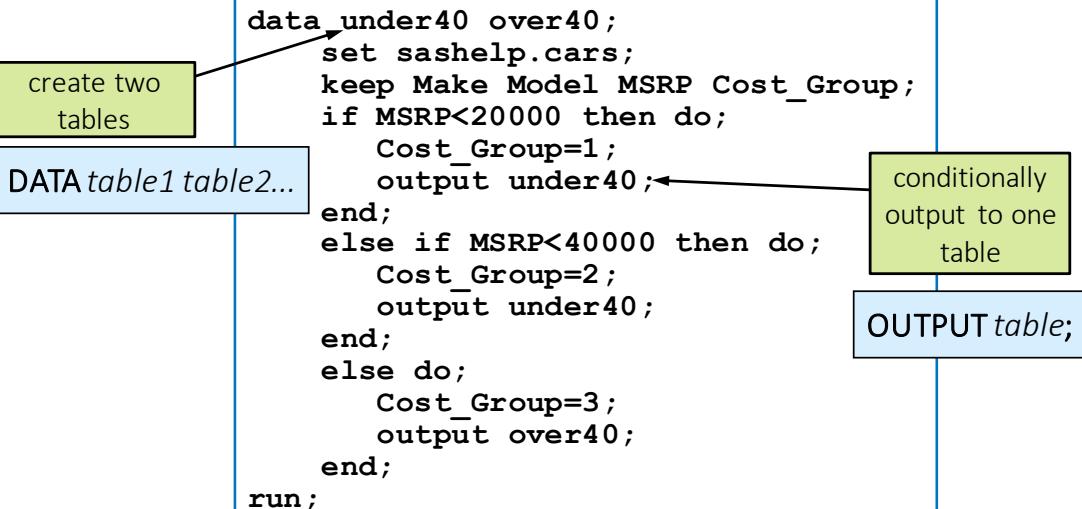
```
IF expression THEN DO;
  <executable statements>
END;
ELSE IF expression THEN DO;
  <executable statements>
END;
ELSE DO;
  <executable statements>
END;
```

If expression is true,
then execute all the
statements between
DO and END.



SAS offers alternate syntax that you can use when you want to execute multiple statements for a given condition. We call this syntax IF-THEN/DO. After a condition, you type **THEN DO** and a semicolon. After that statement, you can list as many statements as you need to process, and then close the block with an END statement. This is repeated for each of the ELSE IF or ELSE DO blocks.

Processing Multiple Statements with IF-THEN/DO



In this example, we use the DATA step to create not one, but two tables. In the DATA statement, we can list more than one output table. In the first condition, if **MSRP** is less than 20000, we assign **Cost_Group** a value of 1, and then use the explicit OUTPUT statement to tell SAS which of the two tables to write that row to. Just remember that because these statements execute in sequence, we must first assign a value to **Cost_Group** and then output the row to a particular table. The remaining conditions also include statements to assign a different value to **Cost_Group** and output to either the **under40** or **over40** table.

4.09 Activity

Open **p104a09.sas** from the **activities** folder. Run the program. Why does the program fail?

```
data front rear;
  set sashelp.cars;
  if DriveTrain="Front" then do;
    DriveTrain="FWD";
    output front;
  else if DriveTrain='Rear' then do;
    DriveTrain="RWD";
    output rear;
run;
```



Processing Multiple Statements with IF-THEN/DO

Scenario

Use IF-THEN/DO syntax to execute multiple statements for each condition.

Files

- **p104d07.sas**
- **storm_summary** – a SAS table that contains one row per storm for the 1980 through 2016 storm seasons

Syntax

```

IF expression THEN DO;
  <executable statements>
END;
ELSE IF expression THEN DO;
  <executable statements>
END;
ELSE DO;
  <executable statements>
END;

```

Notes

- After the IF-THEN/DO statement, list any number of executable statements.
- Close each DO block with an END statement.

Demo

Open **p104d07.sas** from the **demos** folder and find the **Demo** section of the program. Modify the IF-THEN statements to use IF-THEN/DO syntax to write rows to either the **indian**, **atlantic**, or **pacific** table based on the value of **Ocean**. Highlight the DATA step and run the selected code.

```

data indian atlantic pacific;
  set pg1.storm_summary;
  length Ocean $ 8;
  keep Basin Season Name MaxWindMPH Ocean;
  Basin=upcase(Basin);
  OceanCode=substr(Basin,2,1);
  *Modify the program to use IF-THEN-DO syntax;
  if OceanCode="I" then do;
    Ocean="Indian";
    output indian;
  end;
  else if OceanCode="A" then do;
    Ocean="Atlantic";
    output atlantic;
  end;
  else do;
    Ocean="Pacific";
    output pacific;
  end;

```

```
run;
```

indian Table

	② Season	③ Name	④ Basin	② MaxWindMPH	③ Ocean
1	1980	ALBINE	SI	.	Indian
2	1980	AMY	SI	132	Indian
3	1980	BERENICE	SI	.	Indian
4	1980	BRIAN	SI	115	Indian

atlantic Table

	② Season	③ Name	④ Basin	② MaxWindMPH	③ Ocean
1	1980		NA	35	Atlantic
2	1980	ALLEN	NA	190	Atlantic
3	1980	BONNIE	NA	98	Atlantic
4	1980	CHARLEY	NA	81	Atlantic

pacific Table

	② Season	③ Name	④ Basin	② MaxWindMPH	③ Ocean
1	1980		SP	.	Pacific
2	1980	AGATHA	EP	115	Pacific
3	1980	ALEX	WP	40	Pacific
4	1980	BETTY	WP	115	Pacific

End of Demonstration

Beyond SAS Programming 1

What if you want to...

... understand how the DATA step processes behind the scenes to control how data is read and written?

... merge multiple tables with the DATA step?

... simplify repetitive code with DO loops or arrays?

Take the [SAS Programming 2: Data Manipulation Techniques](#) and [SAS Programming 3: Advanced Techniques and Efficiencies](#) courses.

Links

- [SAS Programming 2: Data Manipulation Techniques](#)
- [SAS Programming 3: Advanced Techniques and Efficiencies](#)

Beyond SAS Programming 1

What if you want to...

... manipulate data with PROC SQL?

- Stick around for the last lesson!
- Take the [SAS SQL 1](#) course.

... learn more about PROC DS2 to manipulate data?

- Read this blog post: [Reasons to love PROC DS2](#).
- Take the [DS2 Programming](#) course.

... use the DATA step to read messy raw data files?

- Look for ***Reading Text Files with the DATA Step*** on the Extended Learning page.

70
Copyright © SAS Institute Inc. All rights reserved.



Links

- Take the [SAS SQL 1](#) course.
- Read this blog post: [Reasons to love PROC DS2](#).
- Take the [DS2 Programming](#) course.



Practice

If you restarted your SAS session, open and submit the **libname.sas** program in the course files.

Level 1

7. Processing Statements Conditionally with IF-THEN/ELSE

The **pg1.np_summary** table contains public use statistics from the National Park Service. The values of the **Type** column represent park type as a code. Create a new column, **ParkType**, that contains full descriptive values.

- Open **p104p07.sas** from the **practices** folder. Submit the program and view the generated output.

Type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
NM	63	46.67	63	46.67
NP	51	37.78	114	84.44
NPRE	1	0.74	115	85.19
NS	10	7.41	125	92.59
PRE	3	2.22	128	94.81
PRESERVE	4	2.96	132	97.78
RIVERWAYS	1	0.74	133	98.52
RVR	2	1.48	135	100.00

- In the DATA step, use IF-THEN/ELSE statements to create a new column, **ParkType**, based on the value of **Type**.

Type	ParkType
NM	Monument
NP	Park
NPRE, PRE, or PRESERVE	Preserve
NS	Seashore
RVR or RIVERWAYS	River

- Modify the PROC FREQ step to generate a frequency report for **ParkType**.

ParkType	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Monument	63	46.67	63	46.67
Park	51	37.78	114	84.44
Preserve	8	5.93	122	90.37
River	3	2.22	125	92.59
Seashore	10	7.41	135	100.00

Level 2

8. Processing Statements Conditionally with DO Groups

Use conditional processing to split pg1.np_summary into two tables: **parks** and **monuments**.

- Create a new program. Write a DATA step to create two temporary tables named **parks** and **monuments** based on the **pg1.np_summary** table. Read only national parks or monuments from the input table. (**Type** is either *NP* or *NM*.)
- Create a new column named **Campers** that is the sum of all columns containing counts of campers. Format the column to include commas.
- When **Type** is *NP*, create a new column named **ParkType** that is equal to **Park**, and write the row to the **parks** table. When **Type** is *NM*, assign **ParkType** as *Monument* and write the row to the **monuments** table.
- Keep **Reg**, **ParkName**, **DayVisits**, **OtherLodging**, **Campers**, and **ParkType** in both output tables.

parks Table

	Reg	ParkName	DayVisits	OtherLodging	Campers	ParkType
1	A	Kenai Fjords National Park	346,534	0	2,162	Park
2	A	Kobuk Valley National Park	15,500	0	7,050	Park
3	IM	Arches National Park	1,585,718	0	47,878	Park
4	IM	Big Bend National Park	388,290	48,280	145,425	Park
5	IM	Black Canyon of the Gunnison National Park	238,018	0	32,884	Park

monuments Table

	Reg	ParkName	DayVisits	OtherLodging	Campers	ParkType
1	A	Cape Krusenstern National Monument	15,000	0	6,375	Monument
2	IM	Alibates Flint Quarries National Monument	8,153	0	0	Monument
3	IM	Aztec Ruins National Monument	57,692	0	0	Monument
4	IM	Bandelier National Monument	198,478	0	10,533	Monument
5	IM	Canyon De Chelly National Monument	821,406	23,259	11,918	Monument

Challenge

9. Processing Statements Conditionally with SELECT-WHEN Groups

SELECT and WHEN statements can be used in a DATA step as an alternative to IF-THEN statements to process code conditionally.

- Use SAS Help or online documentation to read about using SELECT and WHEN statements in the DATA step.
- Repeat Practice 8 using SELECT groups and WHEN statements.

End of Practices

4.4 Solutions

Solutions to Practices

1. Creating a SAS Table

```
data eu_occ2016;
  set pg1.eu_occ;
  where YearMon like "2016%";
  format Hotel ShortStay Camp comma17.;
  drop geo;
run;
```

2. Creating a Permanent SAS Table

```
libname out "s:/workshop/output";

data out.fox;
  set pg1.np_species;
  where Category='Mammal' and Common_Names like '%Fox%'
    and Common_Names not like '%Squirrel%';
  drop Category Record_Status Occurrence Nativeness;
run;

proc sort data=out.fox;
  by Common_Names;
run;
```

3. Creating a SAS Table Using Macro Variables

```
*Before macro variable;
data mammal;
  set pg1.np_species;
  where Category="Mammal";
  drop Abundance Seasonality Conservation_Status;
run;

proc freq data=mammal;
  tables Record_Status;
run;

*Using macro variable;
%let cat=Bird;

data &cat;
  set pg1.np_species;
  where Category="&cat";
  drop Abundance Seasonality Conservation_Status;
run;
```

```
proc freq data=&cat;
  tables Record_Status;
run;
```

4. Creating New Columns

```
data np_summary_update;
  set pg1.np_summary;
  keep Reg ParkName DayVisits OtherLodging Acres SqMiles
    Camping;
  SqMiles=Acres*.0015625;
  Camping=sum(OtherCamping,TentCampers,
    RVCampers,BackcountryCampers);
  format SqMiles comma6. Camping comma10.;
run;
```

5. Creating New Columns with Character and Date Functions

```
data eu_occ_total;
  set pg1.eu_occ;
  Year=substr(YearMon,1,4);
  Month=substr(YearMon,6,2);
  ReportDate=MDY(Month,1,Year);
  Total=sum(Hotel,ShortStay,Camp);
  format Hotel ShortStay Camp Total comma17.
    ReportDate monyy7.;
  keep Country Hotel ShortStay Camp ReportDate Total;
run;
```

6. Creating a New Column with the SCAN Function

```
data np_summary2;
  set pg1.np_summary;
  ParkType=scan(parkname,-1);
  keep Reg Type ParkName ParkType;
run;
```

7. Processing Statements Conditionally with IF-THEN/ELSE

```
data park_type;
  set pg1.np_summary;
  length ParkType $ 8;
  if Type='NM' then ParkType='Monument';
  else if Type='NP' then ParkType='Park';
  else if Type in ('NPRE', 'PRE', 'PRESERVE') then
    ParkType='Preserve';
  else if Type='NS' then ParkType='Seashore';
  else if Type in ('RVR', 'RIVERWAYS') then ParkType='River';
run;

proc freq data=park_type;
  tables ParkType;
run;
```

8. Processing Statements Conditionally with DO Groups

```

data parks monuments;
  set pg1.np_summary;
  where type in ('NM', 'NP');
  Campers=sum(OtherCamping, TentCampers, RVCampers,
              BackcountryCampers);
  format Campers comma17.;
  length ParkType $ 8;
  if type='NP' then do;
    ParkType='Park';
    output parks;
  end;
  else do;
    ParkType='Monument';
    output monuments;
  end;
  keep Reg ParkName DayVisits OtherLodging Campers ParkType;
run;

```

9. Processing Statements Conditionally with SELECT-WHEN Groups

```

data parks monuments;
  set pg1.np_summary;
  where type in ('NM', 'NP');
  Campers=sum(OtherCamping, TentCampers, RVCampers,
              BackcountryCampers);
  format Campers comma17.;
  length ParkType $ 8;
  select (type);
    when ('NP') do;
      ParkType='Park';
      output parks;
    end;
    otherwise do;
      ParkType='Monument';
      output monuments;
    end;
  end;
  keep Reg ParkName DayVisits OtherLodging Campers ParkType;
run;

```

End of Solutions

Solutions to Activities and Questions

4.01 Activity – Correct Answer

temporary table

```
data storm_new;
  set pg1.storm_summary;
run;
```

permanent table

```
libname out "s:/workshop/output";
data out.storm_new;
  set pg1.storm_summary;
run;
```

Keep this program open for the next activity.

10



Copyright © SAS Institute Inc. All rights reserved.

4.02 Multiple Answer Question – Correct Answers

The table listed in the SET statement must be read via a library. Which data sources can be used in the SET statement?

- a. SAS tables
- b. Excel spreadsheets
- c. DBMS tables
- d. comma-delimited files

Any data source that can be read via a library can be used as the input table in the SET statement.



12



Copyright © SAS Institute Inc. All rights reserved.

4.03 Activity – Correct Answer

```
data out.storm_cat5;
  set pg1.storm_summary;
  where StartDate>="01jan2000'd and MaxWindMPH>=156;
  keep Season Basin Name Type MaxWindMPH;
run;
```

There were 18 Category 5 storms since January 1, 2000.

NOTE: There were 18 observations read from the data set PG1.STORM_SUMMARY.
WHERE (StartDate>='01JAN2000'D)
and (MaxWindMPH>=156);

NOTE: The data set WORK.STORM_CAT5 has 18 observations and 5 variables.

How is the KEEP statement different from the VAR statement in PROC PRINT?



Sas

18

Copyright © SAS Institute Inc. All rights reserved.

4.04 Activity – Correct Answer

Open p104a04.sas from the activities folder and perform the following tasks:

1. Add an assignment statement to create **StormLength** that represents the number of days between **StartDate** and **EndDate**.
2. Run the program. In 1980, how long did the storm named Agatha last?
6 days

```
data storm_length;
  set pg1.storm_summary;
  drop Hem_EW Hem_NS Lat Lon;
  StormLength = EndDate-StartDate;
run;
```

27

Copyright © SAS Institute Inc. All rights reserved.

Sas

4.05 Activity – Correct Answer

```
data storm_windavg;
  set pg1.storm_range;
  WindAvg=mean(wind1, wind2, wind3, wind4);
  WindRange=range(of wind1-wind4);
run;
```

	Season	Basin	Name	Wind1	Wind2	Wind3	Wind4	WindAvg	WindRange
1	1980	EP	AGATHA	100	95	90	85	92.5	15
2	1980	EP	BLAS	50	50	50	45	48.75	5
3	1980	EP	CELIA	65	65	65	65	65	
4	1980	EP	DARBY	45	45	35	30	38.75	

OF col1 - coln

That's a good shortcut for listing a range of columns!



sas

33

Copyright © SAS Institute Inc. All rights reserved.

4.06 Activity – Correct Answer

```
data pacific;
  set pg1.storm_summary;
  drop type Hem_EW Hem_NS MinPressure Lat Lon;
  where substr(Basin,2,1)="P";
run;
```

NOTE: There were 1958 observations read from the data set PG1.STORM_SUMMARY.
 WHERE SUBSTR(basin, 2, 1)='P';
 NOTE: The data set WORK.PACIFIC has 1958 observations and 6 variables.

	Season	Name	Basin	MaxWindMPH	StartDate	EndDate
	1980	SP	.	.	27MAR1980	30MAR1980
	1980	AGATHA	EP	115	09JUN1980	15JUN1980
	1980	ALEX	WP	40	09OCT1980	14OCT1980
	1980	BETTY	WP	115	28OCT1980	08NOV1980
	1980	BLAS	EP	58	16JUN1980	19JUN1980
	1980	CARMEN	WP	69	05APR1980	07APR1980
	1980	CARY	WP	52	28OCT1980	02NOV1980

38

Copyright © SAS Institute Inc. All rights reserved.

sas

4.07 Activity – Correct Answer

```
data storm_cat;
  set pg1.storm_summary;
  keep Name Basin MinPressure StartDate PressureGroup;
  *add ELSE keyword and remove final condition;
  if MinPressure=. then PressureGroup=.;
  else if MinPressure<=920 then PressureGroup=1;
  else PressureGroup=0;
run;
```

Name	Basin	MinPressure	StartDate	PressureGroup
na		.	17JUL1980	.
SP		998	27MAR1980	0
AGATHA	EP	.	09JUN1980	.
ALBINE	SI	.	27NOV1979	.
ALEX	WP	998	09OCT1980	0
ALLEN	NA	899	31JUL1980	1
AMY	SI	915	04JAN1980	1

The FREQ Procedure

PressureGroup	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	2733	93.53	2733	93.53
1	189	6.47	2922	100.00
Frequency Missing = 196				

54

Copyright © SAS Institute Inc. All rights reserved.



4.08 Activity – Correct Answer

```
data storm_summary2;
  set pg1.storm_summary;
  length Ocean $ 8;
  keep Basin Season Name MaxWindMPH Ocean;
  Basin=uppercase(Basin);
  OceanCode=substr(Basin,2,1);
  if OceanCode="I" then Ocean="Indian";
  else if OceanCode="A" then Ocean="Atlantic";
  else Ocean="Pacific";
run;
```

Without the LENGTH statement, Ocean had a length of 6.

Without converting Basin to uppercase, all lowercase OceanCode values were assigned as Pacific.

60

Copyright © SAS Institute Inc. All rights reserved.



4.08 Activity – Correct Answer

Does it matter where the LENGTH statement is in the DATA step?

Yes, the length of a column is set the first time it occurs in the DATA step.
It cannot be changed by a LENGTH statement that occurs later in the code.

```

56           else Ocean="Pacific";
57           length Ocean $ 8;
WARNING: Length of character variable Ocean has
already been set.
Use the LENGTH statement as the very
first statement in the DATA STEP to
declare the length of a character
variable.
58           run;

```

The order of KEEP, DROP, and WHERE statements does not matter in the DATA step.



61

Copyright © SAS Institute Inc. All rights reserved.

4.09 Activity – Correct Answer

Open p104a09.sas from the activities folder. Run the program. Why does the program fail?

ERROR 117-185: There were 2 unclosed DO blocks.

```

data front rear;
  set sashelp.cars;
  if DriveTrain="Front" then do;
    DriveTrain="FWD";
    output front;
  end;
  else if DriveTrain='Rear' then do;
    DriveTrain="RWD";
    output rear;
  end;
run;

```

Using the Format Code feature in Enterprise Guide and SAS Studio helps you identify the DO blocks.

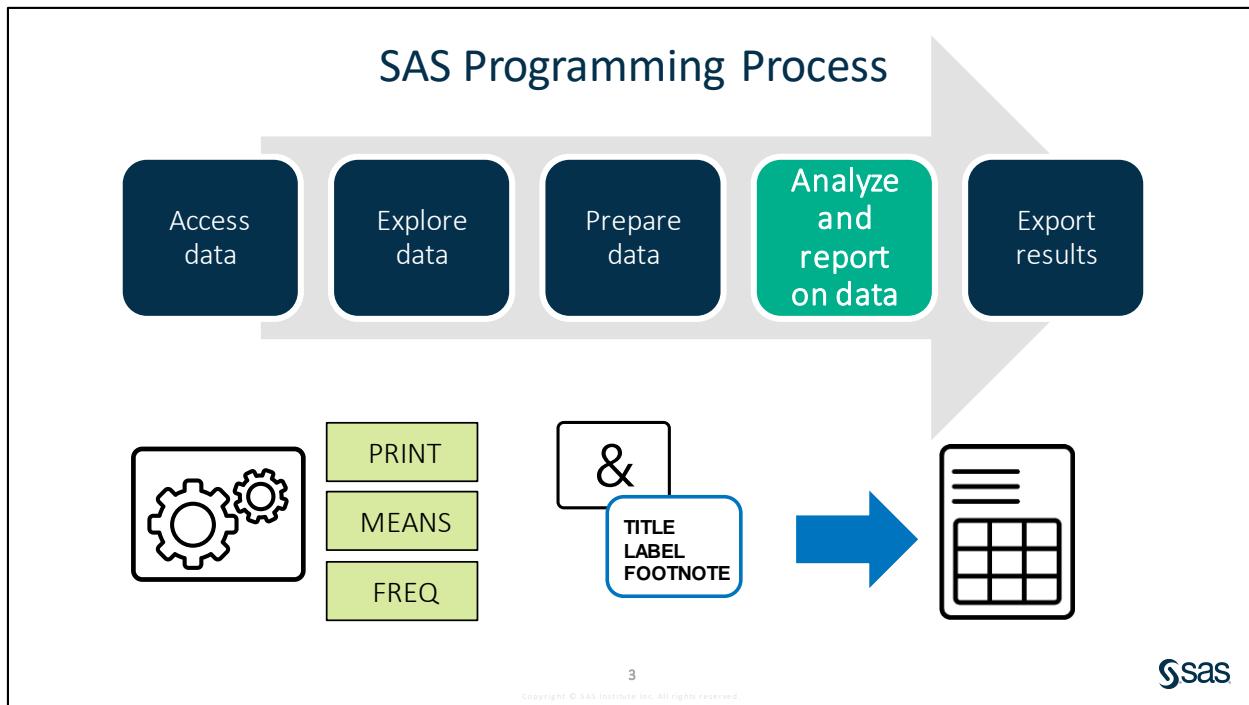


Copyright © SAS Institute Inc. All rights reserved.

Lesson 5 Analyzing and Reporting on Data

5.1 Enhancing Reports with Titles, Footnotes, and Labels.....	5-3
Demonstration: Enhancing Reports	5-9
5.2 Creating Frequency Reports	5-12
Demonstration: Creating Frequency Reports and Graphs	5-14
Demonstration: Creating Two-Way Frequency Reports.....	5-17
Practice.....	5-19
5.3 Creating Summary Statistics Reports	5-23
Demonstration: Creating Summary Statistics Reports	5-24
Practice.....	5-30
5.4 Solutions	5-33
Solutions to Practices	5-33
Solutions to Activities and Questions.....	5-36

5.1 Enhancing Reports with Titles, Footnotes, and Labels



Now that data access, validation, and manipulation are behind you, you are ready to address the peak of the programming process: analyzing and reporting on the data. Analyzing your data can mean a lot of different things. It could be basic summarization to examine what happened in the past, or it could be complex data mining or machine learning algorithms to predict what might happen in the future. In this lesson, you concentrate on summarizing data. Specifically, you explore in more depth the procedures that you can use for exploration: PRINT, MEANS, and FREQ.

Using Titles and Footnotes

TITLE<n>"title-text";

FOOTNOTE<n>"footnote-text";

```
title1 "Class Report";
title2 "All Students";
footnote1 "School Use Only";

proc print data=pg1.class_birthdate;
run;
```

Class Report All Students						
Obs	Name	Sex	Age	Height	Weight	Birthdate
1	Alfred	M	14	69.0	112.5	16370
2	Alice	F	13	56.5	84.0	16756
3	Barbara	F	13	65.3	98.0	16451
4	Carol	F	14	62.8	102.5	16256
5	Henry	M	14	63.5	102.5	16406
6	James	M	12	57.3	83.0	16987
7	Jane	F	12	59.8	84.5	16873
8	Janet	F	15	62.5	112.5	15797
9	Jeffrey	M	13	62.5	84.0	16552
10	John	M	12	59.0	99.5	17036
11	Joyce	F	11	51.3	50.5	17169
12	Judy	F	14	64.3	90.0	16410
13	Louise	F	12	56.3	77.0	17021
14	Mary	F	15	66.5	112.0	15790
15	Philip	M	16	72.0	150.0	15665
16	Robert	M	12	64.8	128.0	16958
17	Ronald	M	15	67.0	133.0	15992
18	Thomas	M	11	57.5	85.0	17243
19	William	M	15	66.5	112.0	16067

School Use Only

4

Copyright © SAS Institute Inc. All rights reserved.

p105d01



TITLE is a global statement that establishes a permanent title for all reports created in your SAS session. The syntax is just the keyword **TITLE** followed by the title text enclosed in quotation marks. You can have up to 10 titles. You specify a number 1 through 10 after the keyword **TITLE** to indicate the line number. **TITLE** and **TITLE1** are equivalent.

You can also add footnotes to any report with the **FOOTNOTE** statement. The same rules for titles apply to footnotes.

5.01 Activity

Open **p105a01.sas** from the **activities** folder and perform the following tasks:

1. In the program, notice that there is a **TITLE** statement followed by two procedures. Run the program. Where does the title appear in the output?
2. Add a **TITLE2** statement above **PROC MEANS** to print a second line:
Summary Statistics for MaxWind and MinPressure
3. Add another **TITLE2** statement above **PROC FREQ** with this title:
Frequency Report for Basin
4. Run the program. Which titles appear above each report?

5

Copyright © SAS Institute Inc. All rights reserved.



5.02 Activity

Open **p105a02.sas** from the **activities** folder. Notice that there are no TITLE statements in the code. Run the program. Does the report have the same titles assigned in the previous activity?

- Yes
- No

7

Copyright © SAS Institute Inc. All rights reserved.



Clearing Titles and Footnotes

TITLE;
FOOTNOTE;

clears titles and footnotes

ODS NOPROCTITLE;

turns off procedure titles

```
title;footnote;
ods noproctitle;
proc means data=sashelp.heart;
  var height weight;
run;
```

It's a good practice
to clear all titles
and footnotes at
the beginning or end
of a program.



9

Copyright © SAS Institute Inc. All rights reserved.



Remember that TITLE and FOOTNOTE are global statements, and they remain active as long as the SAS session is active. If you want to clear the titles and footnotes that you have specified, you can use the keyword TITLE or FOOTNOTE with no text. That is called a null TITLE statement. The null TITLE statement clears all the titles that you have specified on any line. It is a good idea to do this at the end of your program. Client applications such as SAS Studio submit a null TITLE statement for you at the end of your code, but it is a good idea to get in the habit of submitting the statement yourself.

Some procedures include the name of the procedure in a title above the results. You can turn this off by submitting an ODS statement with the NOPROCTITLE option. You do more with ODS in another lesson.

Using Macro Variables in Titles and Footnotes

```
%let age=13;

title1 "Class Report";
title2 "Age=&age";
footnote1 "School Use Only";

proc print data=pg1.class_birthdate;
  where age=&age;
run;

title;
footnote;
```

10 Copyright © SAS Institute Inc. All rights reserved.

Class Report							
Age=13							
Obs	Name	Sex	Age	Height	Weight	Birthdate	
2	Alice	F	13	56.5	84	16756	
3	Barbara	F	13	65.3	98	16451	
9	Jeffrey	M	13	62.5	84	16552	

School Use Only



Applying Temporary Labels to Columns

```
LABEL col-name="label-text";
```

```
proc means data=sashelp.cars;
  where type="Sedan";
  var MSRP MPG_Highway;
  label MSRP="Manufacturer Suggested Retail Price"
        MPG_Highway="Highway Miles per Gallon";
run;
```

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
MSRP	Manufacturer Suggested Retail Price	262	29773.62	15584.59	10280.00	128420.00
MPG_Highway	Highway Miles per Gallon	262	28.6297710	4.4674591	17.0000000	46.0000000



Applying Temporary Labels to Columns

```
proc print data=sashelp.cars label;
  where type="Sedan";
  var Make Model MSRP MPG_Highway MPG_City;
  label MSRP="Manufacturer Suggested Retail Price"
        MPG_Highway="Highway Miles per Gallon";
run;
```

Make	Model	Manufacturer Suggested Retail Price	Highway Miles per Gallon	MPG (City)
Acura	RSX Type S 2dr	\$23,820	31	24
Acura	TSX 4dr	\$26,990	29	22
Acura	TL 4dr	\$33,195	28	20
Acura	3.5 RL 4dr	\$43,755	24	18
Acura	3.5 RL w/Navigation 4dr	\$46,100	24	18
Audi	A4 1.8T 4dr	\$25,940	31	22

Copyright © SAS Institute Inc. All rights reserved.



In PROC PRINT, you must use either the LABEL or SPLIT= option in the PROC PRINT statement to display labels in the report. When you use the LABEL option, SAS determines whether to split the labels to multiple lines, and if so, where to make the split. The SPLIT= option enables you to define a character that forces labels to split in specific locations.

```
proc print data=sashelp.cars split="*";
  var Make Model MSRP MPG_Highway MPG_City;
  label MSRP="Manufacturer Suggested*Retail Price"
        MPG_Highway="Highway Miles*per Gallon";
run;
```

Make	Model	Manufacturer Suggested Retail Price	Highway Miles per Gallon	MPG (City)
Acura	RSX Type S 2dr	\$23,820	31	24
Acura	TSX 4dr	\$26,990	29	22

Segmenting Reports

```

proc sort data=sashelp.cars
           out=cars_sort;
   by Origin;
run;

proc freq data=cars_sort;
   by Origin;
   tables Type;
run;

```

The data must be sorted first before you use the BY statement in a reporting procedure.

The FREQ Procedure				
Origin=Asia				
Type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Hybrid	3	1.90	3	1.90
SUV	25	15.82	28	17.72
Sedan	94	59.49	122	77.22
Sports	17	10.76	139	87.97
Truck	8	5.06	147	93.04
Wagon	11	6.96	158	100.00

The FREQ Procedure				
Origin=Europe				
Type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
SUV	10	8.13	10	8.13
Sedan	78	63.41	88	71.54
Sports	23	18.70	111	90.24
Wagon	12	9.76	123	100.00

The FREQ Procedure				
Origin=USA				
Type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
SUV	25	17.01	25	17.01
Sedan	90	61.22	115	78.23
Sports	9	6.12	124	84.35
Truck	16	10.88	140	95.24
Wagon	7	4.76	147	100.00

13

Copyright © SAS Institute Inc. All rights reserved.

p105d01



You can use the BY statement in a reporting procedure to segment a report based on the unique values of one or more columns. For example, what if you want to generate a separate frequency report for each value of **Origin**? You must sort the table by **Origin** first, and then use the BY statement in PROC FREQ. Then SAS treats the rows for each value of **Origin** as a separate table and runs the frequency report.



Enhancing Reports

Scenario

Use titles, footnotes, labels, and grouping to enhance a report.

Files

- **p105d01.sas**
- **storm_final** – a SAS table that contains one row per storm for the 1980 through 2017 storm seasons. The data was cleaned and prepared previously using the DATA step.

Syntax

```

TITLEn "title-text";
FOOTNOTEn "footnote-text";

LABEL col-name="label-text"
      col-name="label-text";

Grouped Reports (sort first)
PROC procedure-name;
  BY col-name;
RUN;

```

Notes

- TITLE is a global statement that establishes a permanent title for all reports that are created in your SAS session.
- You can have a maximum of 10 titles. You use a number 1 through 10 after the keyword TITLE to indicate the line number. TITLE and TITLE1 are equivalent.
- Titles can be replaced with an additional TITLE statement with the same number. **TITLE;** clears all titles.
- You can also add footnotes to any report with the FOOTNOTE statement. The same rules for titles apply to footnotes.
- Labels can be used to provide more descriptive column headings. A label can include any text at a maximum of 256 characters.
- All procedures automatically display labels except for PROC PRINT. You must add the LABEL option in the PROC PRINT statement.
- To create a grouped report, first use PROC SORT to arrange the data by the grouping column, and then use the BY statement in the reporting procedure.

Demo

1. Open **p105d01.sas** from the **demos** folder and find the **Demo** section of the program. Add a PROC SORT step before PROC PRINT to sort **pg1.storm_final** by **BasinName** and descending **MaxWindMPH**. Create a temporary table named **storm_sort**. Filter the rows to include only **MaxWindMPH>156**.

```

proc sort data=pg1.storm_final out=storm_sort;
  by BasinName descending MaxWindMPH;
  where MaxWindMPH > 156;
run;

```

2. Modify the PROC PRINT step to read the **storm_sort** table and group the report by **BasinName**.
3. Add the following title: **Category 5 Storms**. Clear the title for future results.
4. Add labels for the following columns and ensure that PROC PRINT displays the labels:

MaxWindMPH ⇔ **Max Wind (MPH)**

MinPressure ⇔ **Min Pressure**

StartDate ⇔ **Start Date**

StormLength ⇔ **Length of Storm (days)**

5. Add the **NOOBS** option in the PROC PRINT statement to suppress the OBS column. Highlight the demo program and run the selected code.

```
title "Category 5 Storms";
proc print data=storm_sort label noobs;
  by BasinName;
  var Season Name MaxWindMPH MinPressure StartDate StormLength;
  label MaxWindMPH="Max Wind (MPH)"
        MinPressure="Min Pressure"
        StartDate="Start Date"
        StormLength="Length of Storm (days)";
run;
title;
```

Category 5 Storms					
BasinName=East Pacific					
Season	Name	Max Wind (MPH)	Min Pressure	Start Date	Length of Storm (days)
2015	PATRICIA	213	872	20OCT2015	4
1997	LINDA	184	902	09SEP1997	55
2009	RICK	178	906	15OCT2009	6
1994	JOHN	173	920	11AUG1994	30

End of Demonstration

Applying Permanent Labels to Columns

```

data cars_update;
  set sashelp.cars;
  keep Make Model Type MSRP AvgMPG;
  AvgMPG=mean(MPG_Highway, MPG_City);
  label MSRP="Manufacturer Suggested Retail Price"
        AvgMPG="Average Miles per Gallon";
run;

proc contents data=cars_update;
run;

```

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
5	AvgMPG	Num	8		Average Highway Miles per Gallon
4	MSRP	Num	8	DOLLAR8.	Manufacturer Suggested Retail Price
1	Make	Char	13		
2	Model	Char	40		
3	Type	Char	8		

If labels are assigned in the DATA step, they become permanent attributes in the table.

15

Copyright © SAS Institute Inc. All rights reserved.



When the LABEL statement is used in a DATA step, labels are assigned as permanent attributes in the descriptor portion of the table. When procedures create reports using that data, labels are automatically displayed. Notice that the LABEL option is still required in PROC PRINT.

5.03 Activity

Open **p105a03.sas** from the **activities** folder and perform the following tasks:

1. Modify the LABEL statement in the DATA step to label the **Invoice** column as **Invoice Price**.
2. Run the program. Why do the labels appear in the PROC MEANS report but not in the PROC PRINT report? Fix the program and run it again.

16

Copyright © SAS Institute Inc. All rights reserved.



5.2 Creating Frequency Reports

Creating One-Way Frequency Reports and Graphs

The diagram illustrates the creation of one-way frequency reports and graphs. It shows a SAS log window with PROC FREQ output and a corresponding bar chart.

PROC FREQ Output:

Number of Variable Levels				
Variable	Label	Levels	Missing Levels	Nonmissing Levels
Chol_Status	Cholesterol Status	4	1	3

Cholesterol Status		
Chol_Status	Frequency	Percent
Borderline	1861	36.80
High	1791	35.42
Desirable	1405	27.78
Frequency Missing = 152		

Annotations:

- A green box labeled "number of unique values" points to the "Levels" column in the first table.
- A green box labeled "change statistics" points to the "Percent" column in the second table.
- A green box labeled "order of rows" points to the "Chol_Status" column in the second table.
- A green box labeled "graphs to view distribution" points to the bar chart.

Bar Chart:

Distribution of Chol_Status

Chol_Status	Percent
Desirable	~27%
High	~35%
Borderline	~36%

Callout:

PROC FREQ can do much more than simple frequency counts!

20 Copyright © SAS Institute Inc. All rights reserved.

p105d02

PROC FREQ was used with the TABLES statement for data validation. However, many more statements and options are available in PROC FREQ to customize the output and include additional statistics.

Creating One-Way Frequency Reports and Graphs

PROC FREQ Syntax:

```
PROC FREQ DATA=input-table<options>;
  TABLES col-name(s)</ options>;
  RUN;
```

Annotations:

- A green box below the syntax states: Options can be used to request various statistics, reports, and graphs.
- An orange callout bubble points to the "Options" part of the syntax, stating: Don't forget that you can still use WHERE, FORMAT, LABEL, and BY statements!

21 Copyright © SAS Institute Inc. All rights reserved.

A basic frequency report is based on individual columns. By default, each column listed in the TABLES statement generates a separate frequency table that includes the number and percentage of rows for each value in the data, as well as a cumulative frequency and percent. The numbers included in this report can be customized using options in the PROC FREQ and TABLES statements.



Creating Frequency Reports and Graphs

Scenario

Use statements and options that are available in PROC FREQ to customize frequency reports and graphs.

Files

- **p105d02.sas**
- **storm_final** – a SAS table that contains one row per storm for the 1980 through 2017 storm seasons. The data was cleaned and prepared previously using the DATA step.

Syntax

```
PROC FREQ DATA=input-table <proc-options>;
  TABLES col-name(s) </ options>;
RUN;
```

PROC FREQ statement options:
ORDER=FREQ|FORMATTED|DATA
NLEVELS

TABLES statement options:
NOCUM
NOPERCENT
PLOT=FREQPLOT
(must turn on ODS Graphics)
OUT=output-table

Notes

- One or more TABLES statements can be used to define frequency tables and options.
- ODS Graphics enables graph options to be used in the TABLES statement.
- WHERE, FORMAT, LABEL, and BY statements can be used in PROC FREQ to customize the report.

Demo

Note: Highlight the demo program and run the selected code after each step.

1. Open **p105d02.sas** from the **demos** folder and find the **Demo** section of the program. Highlight the PROC FREQ step and run the selected code. Examine the default results.
2. In the PROC FREQ statement, add the ORDER=FREQ option to sort results by descending frequency. Add the NLEVELS option to include a table with the number of distinct values.

```
proc freq data=pg1.storm_final order=freq nlevels;
```

3. Add the NOCUM option in the TABLES statement to suppress the cumulative columns.

```
tables BasinName Season / nocum;
```

4. Change **Season** to **StartDate** in the TABLES statement. Add a FORMAT statement to display **StartDate** as the month name (MONNAME.).

```
proc freq data=pg1.storm_final order=freq nlevels;
  tables BasinName StartDate / nocum;
  format StartDate monname.;
run;
```

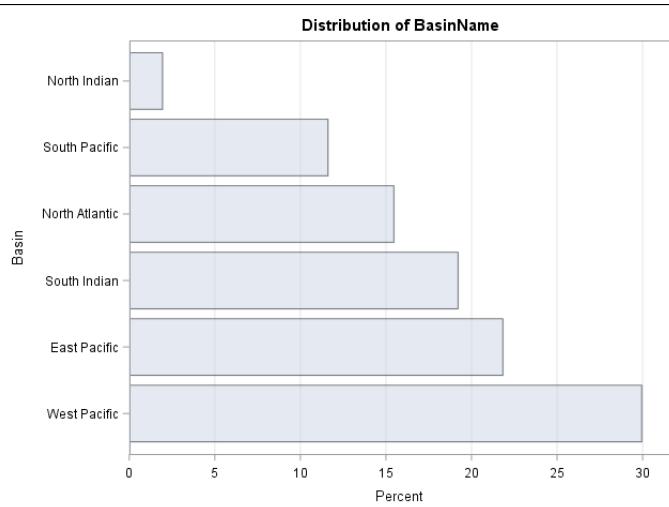
5. Add the ODS GRAPHICS ON statement before PROC FREQ. Use the PLOTS=FREQPLOT option in the TABLES statement to create a bar chart. Add the chart options ORIENT=HORIZONTAL and SCALE=PERCENT.

```
ods graphics on;
proc freq data=pg1.storm_final order=freq nlevels;
  tables BasinName StartDate /
    nocum plots=freqplot(orient=horizontal scale=percent);
  format StartDate monname.;
run;
```

6. Add the title **Frequency Report for Basin and Storm Month**. Turn off the procedure title with the ODS NOPROCTITLE statement. Add a LABEL statement to display **BasinName** as **Basin** and **StartDate** as **Storm Month**. Clear the titles and turn the procedure titles back on.

```
ods graphics on;
ods noproctitle;
title "Frequency Report for Basin and Storm Month";
proc freq data=pg1.storm_final order=freq nlevels;
  tables BasinName StartDate /
    nocum plots=freqplot(orient=horizontal scale=percent);
  format StartDate monname.;
  label BasinName="Basin"
        StartDate="Storm Month";
run;
title;
ods proctitle;
```

Number of Variable Levels		
Variable	Label	Levels
BasinName	Basin	6
StartDate	Storm Month	12
Basin		
BasinName	Frequency	Percent
West Pacific	926	29.95
East Pacific	675	21.83
South Indian	594	19.21
North Atlantic	478	15.46
South Pacific	359	11.61
North Indian	60	1.94



End of Demonstration

5.04 Activity

Open **p105a04.sas** from the **activities** folder and perform the following tasks:

1. Create a temporary output table named **storm_count** by completing the OUT= option in the TABLES statement.
2. Add the NOPRINT option in the PROC FREQ statement to suppress the printed report.
3. Run the program. Which statistics are included in the output table?
Which month has the highest number of storms?

Creating Two-Way Frequency Reports

```
PROC FREQ DATA=input-table< options >;
  TABLES col-name*col-name </ options >;
RUN;
```

rows

columns

```
proc freq data=sashelp.heart;
  tables BP_Status*Chol_Status;
run;
```

Blood Pressure by Cholesterol Status					
Frequency Percent Row Pct Col Pct	Table of BP_Status by Chol_Status				
	BP_Status(Blood Pressure Status)	Borderline	Desirable	High	Total
High	798	456	950	2204	43.58
	15.78	9.02	18.79		
	36.21	20.69	43.10		
	42.88	32.46	53.04		
Normal	793	634	655	2082	41.17
	15.68	12.54	12.95		
	38.09	30.45	31.46		
	42.61	45.12	36.57		
Optimal	270	315	186	771	15.25
	5.34	6.23	3.68		
	35.02	40.86	24.12		
	14.51	22.42	10.39		
Total	1861	1405	1791	5057	100.00
	36.80	27.78	35.42	100.00	
Frequency Missing = 152					

When you place an asterisk between two columns in the TABLES statement, PROC FREQ produces a two-way frequency or crosstabulation report. A two-way frequency report can use some of the same options that we have seen with the one-way frequency report, including NLEVELS to create the number of levels table, ORDER= to control the sequence of rows, and OUT= to create an output table. But there are additional options unique to the two-way frequency report that enable you to apply different layouts to the results or include new statistics or analyses.



Creating Two-Way Frequency Reports

Scenario

Create a two-way frequency report using PROC FREQ to customize the results with options.

Files

- **p105d03.sas**
- **storm_final** – a SAS table that contains one row per storm for the 1980 through 2017 storm seasons. The data was cleaned and prepared previously using the DATA step.

Syntax

```
PROC FREQ DATA=input-table;
  TABLES col-name*col-name
    </ options>;
RUN;
```

PROC FREQ *statement options*:
NOPRINT

TABLES *statement options*:
NOROW
NOCOL
NOPERCENT
CROSSLIST
LIST
OUT=*output-table*

Notes

- When you place an asterisk between two columns in the TABLES statement, PROC FREQ produces a two-way frequency or crosstabulation report. The values of the first listed column are the rows of the report, and the values of the second column are the columns.
- Use options in the TABLES statement to customize the table structure and the statistics that are included in the output.

Demo

Note: Highlight the PROC FREQ step and run the selected code after each step.

1. Open **p105d03.sas** from the **demos** folder and find the **Demo** section of the program. Highlight the PROC FREQ step, run the selected code, and examine the default results.
2. Add the NOPERCENT, NOROW, and NOCOL options in the TABLES statement.

```
tables StartDate*BasinName / norow nocol nopercnt;
```

3. Delete the options in the TABLES statement and add the CROSSLIST option.

```
tables StartDate*BasinName / crosslist;
```

4. Change the CROSSLIST option to the LIST option in the TABLES statement.

```
tables StartDate*BasinName / list;
```

5. Delete the previous options and add OUT=STORMCOUNTS. Add NOPRINT to the PROC FREQ statement to suppress the report.

```
proc freq data=pg1.storm_final noprint;
  tables StartDate*BasinName / out=stormcounts;
  format StartDate monname.;
  label BasinName="Basin"
        StartDate="Storm Month";
run;
```

	StartDate	BasinName	COUNT	PERCENT
1	November	East Pacific	15	0.4851228978
2	November	North Atlantic	26	0.8408796895
3	November	North Indian	19	0.6144890039
4	November	South Indian	41	1.3260025873
5	November	South Pacific	14	0.4527813713
6	November	West Pacific	74	2.3932729625
7	December	East Pacific	3	0.0970245796
8	December	North Atlantic	5	0.1617076226

End of Demonstration



Practice

If you restarted your SAS session, open and submit the **libname.sas** program in the course files.

Level 1

1. Creating One-Way Frequency Reports

The **pg1.np_species** table provides a detailed species list for selected national parks. Use this table to analyze categories of reported species.

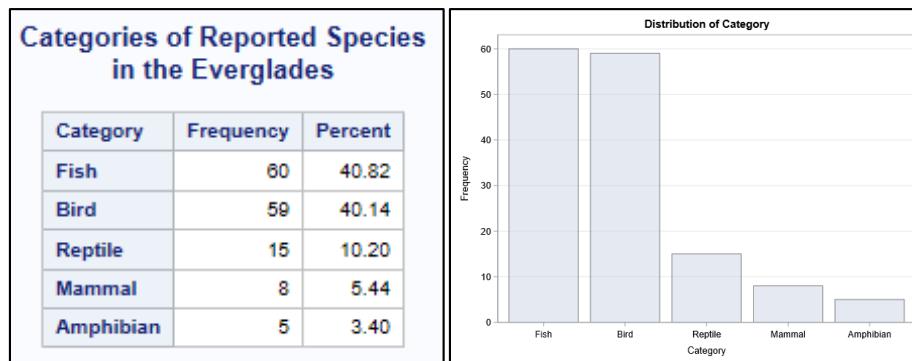
- a. Create a new program. Write a PROC FREQ step to analyze rows from **pg1.np_species**.
 - 1) Use the TABLES statement to generate a frequency table for **Category**.
 - 2) Use the NOCUM options to suppress the cumulative columns.
 - 3) Use the ORDER=FREQ option in the PROC FREQ statement to order the results by descending frequency.
 - 4) Use **Categories of Reported Species** as the report title.
 - 5) Run the program and review the results.

Categories of Reported Species		
The FREQ Procedure		
Category	Frequency	Percent
Vascular Plant	9614	54.30
Bird	2141	12.09
Insect	2104	11.88
Fungi	939	5.30
Nonvascular Plant	672	3.80
Mammal	619	3.50
Fish	576	3.25
Invertebrate	224	1.27
Reptile	216	1.22
Algae	167	0.94
Spider/Scorpion	121	0.68
Amphibian	114	0.64
Slug/Snail	110	0.62
Crab/Lobster/Shrimp	89	0.50

- b. Modify the PROC FREQ step to make the following changes:
 - 1) Include only rows where **Species_ID** starts with *EVER* and **Category** is *not* *Vascular Plant*.

Note: EVER represents Everglades National Park.
 - 2) Turn on ODS Graphics before the PROC FREQ step and turn off the procedure title. Add the PLOTS=REQPLOT option to display frequency plots.

- 3) Add **in the Everglades** as a second title. Run the program and review the results.



Level 2

2. Creating Two-Way Frequency Reports

The **pg1.np_codelookup** table is primarily used to look up a park name or park code. However, the table also includes columns for the park type and park region. Use this table to analyze the frequency of park types by the various regions.

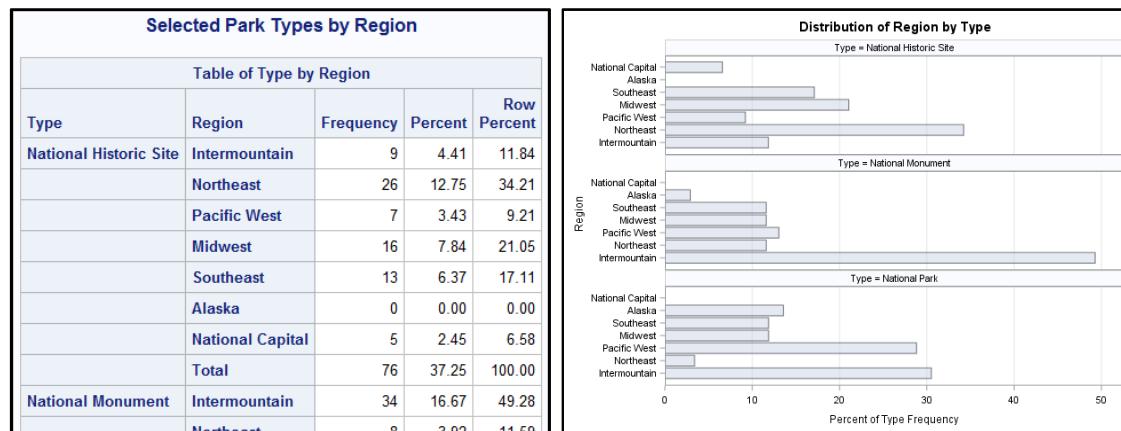
- Create a new program. Write a PROC FREQ step to analyze rows from **pg1.np_codelookup**. Generate a two-way frequency table for **Type** by **Region**. Exclude any park type that contains the word *Other*. The levels with the most rows should come first in the order. Suppress the display of column percentages. Use **Park Types by Region** as the report title.
- Run the program and review the results. Identify the top three park types based on total frequency count.

Note: Statistics labels appear in the main table in Enterprise Guide if SAS Report is the output format.

Park Types by Region									
Frequency Percent Row Pct	Type	Table of Type by Region							
		Region							
		Intermountain	Northeast	Southeast	Pacific West	Midwest	National Capital	Alaska	Total
National Historic Site		9	26	13	7	16	5	0	76
		2.52	7.28	3.64	1.96	4.48	1.40	0.00	21.29
		11.84	34.21	17.11	9.21	21.05	6.58	0.00	
National Monument		34	8	8	9	8	0	2	69
		9.52	2.24	2.24	2.52	2.24	0.00	0.56	19.33
		49.28	11.59	11.59	13.04	11.59	0.00	2.90	
National Park		18	2	7	17	7	0	8	59
		5.04	0.56	1.96	4.76	1.96	0.00	2.24	16.53
		30.51	3.39	11.86	28.81	11.86	0.00	13.56	
National Historical Park		6	14	7	9	3	2	2	43

- c. Modify the PROC FREQ step by limiting the park types to the three that were determined in the previous step. In addition to suppressing the display of column percentages, display the table using the CROSSLIST option. Add a frequency plot that groups the bars by the row variable, displays row percentages, and has a horizontal orientation. Use **Selected Park Types by Region** as the report title. Run the program and review the results.

Note: Use SAS documentation to learn how the GROUPBY=, SCALE=, and ORIENT= options can be used to control the appearance of the plot.

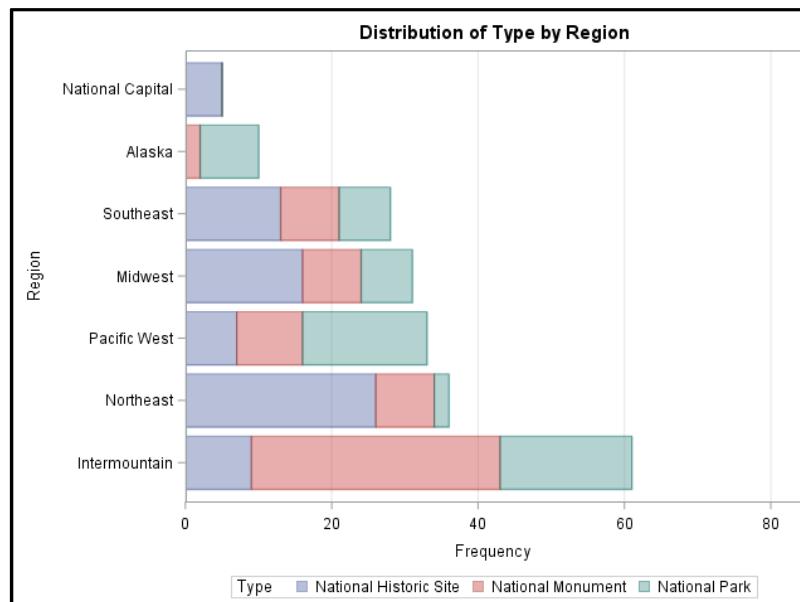


Challenge

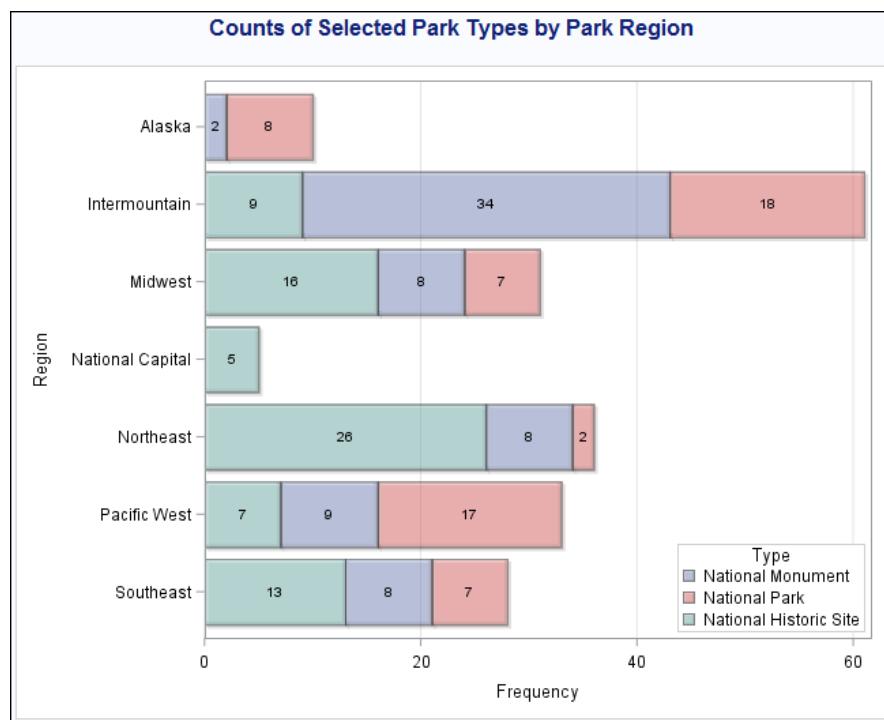
3. Creating a Customized Graph of a Two-Way Frequency Table

The SGLOT procedure can be used to create statistical graphics such as histograms and regression plots, in addition to simple graphics such as bar charts and line plots. Statements and options enable you to control the appearance of your graph and add additional features such as legends and reference lines.

- a. Open **p105p03.sas** from the **practices** folder. Highlight the first TITLE statement and PROC FREQ step, run the selected code, and examine the generated plot. The program subsets the **pg1.np_codelookup** table for three park types: National Historic Site, National Monument, and National Park. The plot uses a stacked layout with a horizontal orientation.



- b. To create a more customized frequency bar chart, the SGLOT procedure can be used with the pg1.np_codelookup table. Examine the PROC SGLOT step in the demo program.
 - 1) The HBAR statement creates a horizontal bar chart with separate bars for each value of **Type**. The GROUP= option segments each bar by the distinct values of **Type**.
 - 2) The KEYLEGEND statement customizes the appearance and position of the legend.
 - 3) The XAXIS statement adds reference lines on the horizontal axis.
- c. Use SAS Help or autocomplete prompts to look for additional options in the HBAR statement to customize the appearance of the chart.
 - 1) Display labels on each segment of the bars.
 - 2) Change the fill attributes for each bar to make the color 50% transparent.
 - 3) Apply different values for the DATASKIN option to change the color effect on the bars.



End of Practices

5.3 Creating Summary Statistics Reports

Creating a Summary Statistics Report

Blood Pressure Status	Cholesterol Status	N Obs	Variable	Mean	Median	Std Dev
High	Borderline	798	Weight	162.9	160.0	29.0
				Cholesterol	219.9	220.0
Desirable		456	Weight	161.0	157.0	32.2
				Cholesterol	178.9	182.0
Normal	High	950	Weight	161.0	159.0	28.7
				Cholesterol	278.1	268.0
Borderline	Normal	793	Weight	150.7	149.0	26.5
				Cholesterol	218.5	219.0
Desirable		634	Weight	145.4	142.0	27.3
				Cholesterol	178.3	182.0
Optimal	Desirable	655	Weight	151.1	148.0	26.5
				Cholesterol	271.6	266.0
Borderline	Optimal	270	Weight	139.5	137.5	24.8
				Cholesterol	218.7	219.0
Desirable		315	Weight	136.3	133.0	23.0
				Cholesterol	174.9	175.0

Sex	Mean	Lower 95% CL for Mean	Upper 95% CL for Mean
Female	229	227	230
Male	226	224	228

group data

specify statistics

create output table

PROC MEANS makes it easy to summarize your data in reports or tables!

Sas

29

Copyright © SAS Institute Inc. All rights reserved.

PROC MEANS is a very useful procedure for calculating basic summary statistics and looking for numeric values that might be outside of an expected range. Now that you are beyond validation, you can use PROC MEANS to generate complex reports that include various statistics and groupings within the data.

Creating a Summary Statistics Report

```
PROC MEANS DATA=input-table <stat-list><options>;
  VAR col-name(s);
  CLASS col-name(s);
  WAYS n;
  RUN;
```

group the statistics based on distinct values of columns in the CLASS statement

specify ways to combine values of columns in the CLASS statement

specify statistics and options to include in the report

30

Copyright © SAS Institute Inc. All rights reserved.

Sas



Creating Summary Statistics Reports

Scenario

Use statements and options that are available in PROC MEANS to create a custom summary statistics report.

Files

- **p105d04.sas**
- **storm_final** – a SAS table that contains one row per storm for the 1980 through 2017 storm seasons. The data was cleaned and prepared previously using the DATA step.

Syntax

```
PROC MEANS DATA=input-table stat-list;  
    VAR col-name(s);  
    CLASS col-name(s);  
    WAYS n;  
RUN;
```

Notes

- Options in the PROC MEANS statement control the statistics that are included in the report.
- The CLASS statement specifies columns to group the data before calculating statistics.
- The WAYS statement specifies the number of ways to make unique combinations of class columns.

Demo

Note: Highlight the PROC MEANS step and run the selected code after each step.

1. Open **p105d04.sas** from the **demos** folder and find the **Demo** section of the program. Run the step and examine the starting report.
2. List the following statistics in the PROC MEANS statement: MEAN, MEDIAN, MIN, and MAX. Add the MAXDEC=0 option to round statistics to the nearest integer.

```
proc means data=pg1.storm_final mean median min max maxdec=0;
```

3. The CLASS statement can be used to calculate statistics for groups. Add a CLASS statement and list the **BasinName** column.

Note: The CLASS statement does not require the data to be sorted.

```
proc means data=pg1.storm_final mean median min max maxdec=0;  
    var MaxWindMPH;  
    class BasinName;  
run;
```

4. Add **StormType** as an additional column in the CLASS statement. Run the program and notice that one report is created with statistics that are calculated for the combination of **BasinName** and **StormType** values.

```
class BasinName StormType;
```

5. The WAYS statement can be used to indicate the combinations of class columns to use for creating the report. Add the WAYS statement and provide a value of 1.

```
proc means data=pg1.storm_final mean median min max maxdec=0;
  var MaxWindMPH;
  class BasinName StormType;
  ways 1;
run;
```

6. Change the WAYS statement to list 0, 1, and 2.

```
proc means data=pg1.storm_final mean median min max maxdec=0;
  var MaxWindMPH;
  class BasinName StormType;
  ways 0 1 2;
run;
```

Analysis Variable : MaxWindMPH					
N Obs	Mean	Median	Minimum	Maximum	
3038	80	75	6	213	

Analysis Variable : MaxWindMPH					
StormType	N Obs	Mean	Median	Minimum	Maximum
Disturbance	289	75	63	17	178
Extratropical	758	90	86	35	184
Not Reported	647	78	69	6	173
Subtropical	4	60	52	43	92
Tropical	1340	76	69	23	213

Analysis Variable : MaxWindMPH					
BasinName	N Obs	Mean	Median	Minimum	Maximum
East Pacific	655	83	75	17	213

End of Demonstration

5.05 Activity

Open **p105a05.sas** from the **activities** folder and perform the following tasks:

1. Add options to include N (count), MEAN, and MIN statistics. Round each statistic to the nearest integer.
2. Add a CLASS statement to group the data by **Season** and **Ocean**. Run the program.
3. Modify the program to add the WAYS statement so that separate reports are created for **Season** and **Ocean** statistics. Run the program.

Which ocean had the lowest mean for minimum pressure?

Which season had the lowest mean for minimum pressure?

Creating an Output Summary Table

```
OUTPUT OUT=output-table<statistic=col-name>;
```

```
proc means data=sashelp.heart noprint;
  var Weight;
  class Chol_Status;
  ways 1;
  output out=heart_stats mean=AvgWeight;
run;
```

Chol_Status	_TYPE_	_FREQ_	AvgWeight
Borderline	1	1861	154.31827957
Desirable	1	1405	148.43121882
High	1	1791	155.4082774

When you analyze detailed data, you might want to create a SAS table that summarizes the data for further analysis. PROC MEANS is a great way to create summary tables. The OUTPUT statement offers several options to customize the table that is generated. You use the OUT= option to name the output table. The OUTPUT statement also enables you to generate output statistics and name a column to store them in.

5.06 Activity

Open **p105a06.sas** from the **activities** folder and perform the following tasks:

1. Run the PROC MEANS step and compare the report and the **wind_stats** table. Are the same statistics in the report and table? What do the first five rows in the table represent?
2. Uncomment the WAYS statement. Delete the statistics listed in the PROC MEANS statement and add the NOPRINT option. Run the program. Notice that a report is not generated and the first five rows from the previous table are excluded.
3. Add the following options in the OUTPUT statement and run the program again. How many rows are in the output table?

```
output out=wind_stats mean=AvgWind max=MaxWind;
```

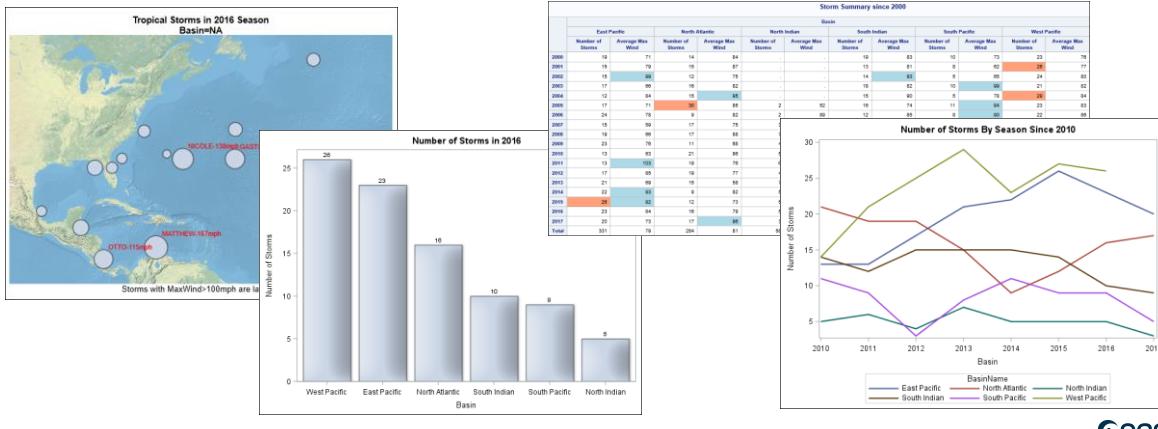
35

Copyright © SAS Institute Inc. All rights reserved.



5.07 Activity

Open **p105a07.sas** from the **activities** folder. Run the program and examine the results to see examples of other procedures that analyze and report on the data.



38

Copyright © SAS Institute Inc. All rights reserved.



The map is created using the SGMAP procedure, which requires SAS 9.4M5 or later.

Beyond SAS Programming 1

What if you want to ...

... create high-quality, customized graphs?

- Review the [SAS 9.4 ODS Graphics documentation](#).
- Take the [ODS Graphics: Essentials](#) course.
- Use this [ODS Graphics tip sheet](#) as a reference.

... learn about basic statistical procedures to analyze your data?

- Take the free e-learning [Statistics 1: Introduction to ANOVA, Regression, and Logistic Regression](#) course.
- Check out other training options for [advanced analytics](#).

... generate detail and summary tabular reports?

- Learn to use PROC REPORT and PROC TABULATE in the [SAS Report Writing 1: Essentials](#) course.
- Read [PROC REPORT by Example: Techniques for Building Professional Reports Using SAS](#).

39

Copyright © SAS Institute Inc. All rights reserved.



Links

- Review the [SAS 9.4 ODS Graphics documentation](#).
- Take the [ODS Graphics: Essentials](#) course.
- Use this [ODS Graphics tip sheet](#) as a reference.
- Take the free e-learning [Statistics 1: Introduction to ANOVA, Regression, and Logistic Regression](#) course.
- Check out other training options for [advanced analytics](#).
- Learn to use PROC REPORT and PROC TABULATE in the [SAS Report Writing 1: Essentials](#) course.
- Read [PROC REPORT by Example: Techniques for Building Professional Reports Using SAS](#).



Practice

If you restarted your SAS session, open and submit the **libname.sas** program in the course files.

Level 1

4. Producing a Descriptive Statistic Report

The **pg1.np_westweather** table contains weather-related information for four national parks: Death Valley National Park, Grand Canyon National Park, Yellowstone National Park, and Zion National Park. Use the MEANS procedure to analyze the data in this table.

- a. Create a new program. Write a PROC MEANS step to analyze rows from **pg1.np_westweather** with the following specifications:
 - 1) Generate the mean, minimum, and maximum statistics for the **Precip**, **Snow**, **TempMin**, and **TempMax** columns.
 - 2) Use the MAXDEC= option to display the values with a maximum of two decimal positions.
 - 3) Use the CLASS statement to group the data by **Year** and **Name**.
 - 4) Use **Weather Statistics by Year and Park** as the report title. Run the program and review the results.

Weather Statistics by Year and Park						
Year	NAME	N Obs	Variable	Mean	Minimum	Maximum
2015	DEATH VALLEY, CA US	365	PRECIP	0.01	0.00	0.55
			SNOW	0.00	0.00	0.00
			TEMPMIN	64.44	26.00	97.00
			TEMPMAX	93.29	53.00	125.00
	GRAND CANYON VISITOR CENTER, AZ US	365	PRECIP	0.07	0.00	2.20
			SNOW	0.13	0.00	5.70
			TEMPMIN	40.99	8.00	69.00
			TEMPMAX	61.60	17.00	93.00
	YELLOWSTONE NATIONAL PARK EAST ENTRANCE, WY US	363	PRECIP	0.06	0.00	0.85
			SNOW	0.33	0.00	10.00
			TEMPMIN	23.05	-22.00	48.00
			TEMPMAX	53.11	9.00	89.00
	ZION NATIONAL PARK, UT US	362	PRECIP	0.05	0.00	1.23
			SNOW	0.01	0.00	4.00
			TEMPMIN	49.63	7.00	78.00
			TEMPMAX	75.60	35.00	110.00

Level 2

5. Creating an Output Table with Custom Columns

The **pg1.np_westweather** table contains weather-related information for four national parks: Death Valley National Park, Grand Canyon National Park, Yellowstone National Park, and Zion National Park. Use the MEANS procedure to analyze the data in this table.

- a. Create a new program. Write a PROC MEANS step to analyze rows from **pg1.np_westweather** where values for **Precip** are **not** equal to zero. Analyze precipitation amounts grouped by **Name** and **Year**. Create only an output table, named **rainstats**, with columns for the N and SUM statistics. Name the columns **RainDays** and **TotalRain** respectively. Keep only those rows that are the combination of **Year** and **Name**.
- b. Write a PROC PRINT step to print the **rainstats** table. Suppress the printing of observation numbers, and display column labels. Display the columns in the following order: **Name**, **Year**, **RainDays**, and **TotalRain**. Label **Name** as **Park Name**, **RainDays** as **Number of Days Raining**, and **TotalRain** as **Total Rain Amount (inches)**. Use **Rain Statistics by Year and Park** as the report title.
- c. Run the program and review the results.

Rain Statistics by Year and Park			
Park Name	Year	Number of Days Raining	Total Rain Amount (inches)
DEATH VALLEY, CA US	2015	15	2.45
DEATH VALLEY, CA US	2016	16	1.42
DEATH VALLEY, CA US	2017	11	1.46
GRAND CANYON VISITOR CENTER, AZ US	2015	97	25.9
GRAND CANYON VISITOR CENTER, AZ US	2016	82	21.1
GRAND CANYON VISITOR CENTER, AZ US	2017	65	11
YELLOWSTONE NATIONAL PARK EAST ENTRANCE, WY US	2015	150	22.2
YELLOWSTONE NATIONAL PARK EAST ENTRANCE, WY US	2016	149	23.4
YELLOWSTONE NATIONAL PARK EAST ENTRANCE, WY US	2017	143	25.7
ZION NATIONAL PARK, UT US	2015	77	16.9
ZION NATIONAL PARK, UT US	2016	68	21.7
ZION NATIONAL PARK, UT US	2017	56	14.5

Challenge

6. Identifying the Top Three Extreme Values with the Output Statistics

- a. Create a new program. Write a PROC MEANS step to analyze rows from **pg1.np_multiyr** and create a table named **top3parks** with the following attributes:
 - 1) Suppress the display of the PROC MEANS report.
 - 2) Analyze **Visitors** grouped by **Region** and **Year**.
 - 3) Drop the **_FREQ_** and **_TYPE_** columns from **top3parks** and keep only rows that are a result of a combination of **Region** and **Year**.
 - 4) Create a column for **TotalVisitors** in the output table.
 - 5) Include in the output table the top three parks in terms of the number of visitors. Automatically resolve conflicts in the column names when names are assigned to the new columns in the output table.

Note: Use SAS Help to learn about the IDGROUP option in the OUTPUT statement.

- b. Run the program and review the output.

	Region	Year	TotalVisitors	Visitors_1	Visitors_2	Visitors_3	ParkName_1	ParkName_2	ParkName_3
1	Alaska	2010	2,274,843	193,116	191,495	188,594	Klondike Go...	Klondike Gol...	Klondike Gol...
2	Alaska	2011	2,333,919	208,958	189,427	187,383	Klondike Go...	Klondike Gol...	Klondike Gol...
3	Alaska	2012	2,412,524	201,814	187,285	183,204	Klondike Go...	Klondike Gol...	Klondike Gol...
4	Alaska	2013	2,585,980	260,494	235,738	229,747	Klondike Go...	Klondike Gol...	Klondike Gol...
5	Alaska	2014	2,684,693	278,870	259,349	237,976	Klondike Go...	Klondike Gol...	Klondike Gol...
6	Alaska	2015	2,664,293	239,023	213,899	209,604	Klondike Go...	Klondike Gol...	Klondike Gol...
7	Alaska	2016	2,783,011	224,793	221,231	219,057	Klondike Go...	Klondike Gol...	Klondike Gol...
8	Intermountain	2010	42,652,924	957,785	854,837	694,841	Yellowstone...	Yellowstone...	Yellowstone...
9	Intermountain	2011	40,543,746	906,935	805,173	743,741	Yellowstone...	Yellowstone...	Rocky Mount...
10	Intermountain	2012	41,274,295	929,225	790,706	671,025	Yellowstone	Yellowstone	Yellowstone

End of Practices

5.4 Solutions

Solutions to Practices

1. Creating One-Way Frequency Reports

```

/*part a*/
title1 "Categories of Reported Species";
proc freq data=pg1.np_species order=freq;
  tables Category / nocum;
run;

/*part b*/
ods graphics on;
ods noproctitle;
title1 "Categories of Reported Species";
title2 "in the Everglades";
proc freq data=pg1.np_species order=freq;
  tables Category / nocum plots=freqplot;
  where Species_ID like "EVER%" and
        Category ne "Vascular Plant";
run;
title;

```

2. Creating Two-Way Frequency Reports

What are the top three park types based on total frequency?

National Historic Site, National Monument, and National Park

```

/*part a, b*/
title1 'Park Types by Region';
proc freq data=pg1.np_codelookup order=freq;
  tables Type*Region / nocol;
  where Type not like '%Other%';
run;

/*part c*/
title1 'Selected Park Types by Region';
ods graphics on;
proc freq data=pg1.np_codelookup order=freq;
  tables Type*Region / nocol crosslist
    plots=freqplot(groupby=row scale=grouppercent
                  orient=horizontal);
  where Type in ('National Historic Site', 'National Monument',
                 'National Park');
run;
title;

```

3. Creating a Customized Graph of a Two-Way Frequency Table

```

/*part a*/
title1 'Counts of Selected Park Types by Park Region';
ods graphics on;
proc freq data=pg1.np_codelookup order=freq;
    tables Type*Region / crosslist plots=freqplot(twoway=stacked
                                                orient=horizontal);
    where Type in ('National Historic Site', 'National Monument',
                   'National Park');
run;

/*part b */
title1 'Counts of Selected Park Types by Park Region';
ods graphics on;
proc freq data=pg1.np_codelookup order=freq noprint;
    tables Type*Region / out=park_freq;
    where Type in ('National Historic Site', 'National Monument',
                   'National Park');
run;

/*part c*/
proc sgplot data=pg1.np_codelookup;
    where Type in ('National Historic Site', 'National Monument',
                   'National Park');
    hbar region / group=type;
    keylegend / opaque across=1 position=bottomright
                location=inside;
    xaxis grid;
run;

/*part d*/
proc sgplot data=pg1.np_codelookup;
    where Type in ('National Historic Site', 'National Monument',
                   'National Park');
    hbar region / group=type seglabel
                  fillattrs=(transparency=0.5) dataskin=crisp;
    keylegend / opaque across=1 position=bottomright
                location=inside;
    xaxis grid;
run;
title;

```

4. Producing a Descriptive Statistic Report

```

title1 'Weather Statistics by Year and Park';
proc means data=pg1.np_westweather mean min max maxdec=2;
    var Precip Snow TempMin TempMax;
    class Year Name;
run;

```

5. Creating an Output Table with Custom Columns

```

proc means data=pg1.np_westweather noprint;
  where Precip ne 0;
  var Precip;
  class Name Year;
  ways 2;
  output out=rainstats n=RainDays sum=TotalRain;
run;

title1 'Rain Statistics by Year and Park';
proc print data=rainstats label noobs;
  var Name Year RainDays TotalRain;
  label Name='Park Name'
    RainDays='Number of Days Raining'
    TotalRain='Total Rain Amount (inches)';
run;
title;

```

6. Identifying the Top Three Extreme Values with the Output Statistics

```

proc means data=pg1.np_multiyr noprint;
  var Visitors;
  class Region Year;
  ways 2;
  output out=top3list(drop=_freq__type_)
    sum=TotalVisitors /*sum total visitors*/
    idgroup(max(Visitors) /*find the max of visitors*/
    out[3] /*top 3*/
    (Visitors ParkName)=) /*output columns for top 3 parks*/
run;

```

End of Solutions

Solutions to Activities and Questions

5.01 Activity – Correct Answer

```

title "Storm Analysis";
title2 "Summary Statistics for MaxWind and MinPressure";
proc means data=pg1.storm_final;
  var MaxWindMPH MinPressure;
run;
title2 "Frequency Report for Basin";
proc freq data=pg1.storm_final;
  tables BasinName;
run;

```

The first title appears above both reports.
The second title is replaced for the PROC FREQ output.


Copyright © SAS Institute Inc. All rights reserved.
6

5.02 Activity – Correct Answer

Does the report have titles?

- Yes (SAS Enterprise Guide)
- No (SAS Studio)

Some procedures automatically add a procedure title.

The MEANS Procedure

SAS Studio

Variable	N	Mean	Std Dev	Minimum	Maximum
Height	5203	64.8131847	3.5827074	51.5000000	76.5000000
Weight	5203	153.0866808	28.9154261	67.0000000	300.0000000

Storm Analysis
Frequency Report for Basin

Enterprise Guide

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Height	5203	64.8131847	3.5827074	51.5000000	76.5000000
Weight	5203	153.0866808	28.9154261	67.0000000	300.0000000

It depends. SAS Studio automatically clears existing titles before a new program is submitted.

Enterprise Guide does not.


Copyright © SAS Institute Inc. All rights reserved.
8

continued...

5.03 Activity – Correct Answer

1. Modify the LABEL statement in the DATA step to label the Invoice column as Invoice Price.

```
data cars_update;
  set sashelp.cars;
  keep Make Model MSRP Invoice AvgMPG;
  AvgMPG=mean(MPG_Highway, MPG_City);
  label MSRP="Manufacturer Suggested Retail Price"
        AvgMPG="Average Miles per Gallon"
        Invoice="Invoice Price";
run;
```

17

Copyright © SAS Institute Inc. All rights reserved.



5.03 Activity – Correct Answer

2. Why do the labels appear in the PROC MEANS report but not in the PROC PRINT report? Fix the program and run it again.

```
proc means data=cars_update min mean max;
  var MSRP Invoice;
run;

proc print data=cars_update label;
  var Make Model MSRP Invoice AvgMPG;
run;
```

Most procedures automatically display permanent labels, but PROC PRINT still needs the LABEL option.



18

Copyright © SAS Institute Inc. All rights reserved.



5.04 Activity – Correct Answer

Which statistics are included? Count and Percent

Which month has the highest number of storms?

September (With ORDER=FREQ, the highest count is listed first.)

```
title "Frequency Report for Storm Month";
proc freq data=pg1.storm_final order=freq noprint;
  tables StartDate / out=storm_count;
  format StartDate monname.;
run;
```

	StartDate	COUNT	PERCENT
1	September	486	15.71981889
2	August	485	15.685640362
3	July	346	11.190168176
4	October	326	10.543337646
5	January	246	7.9560155239
6	February	224	7.2445019405
7	November	100	3.1125405122

24

Copyright © SAS Institute Inc. All rights reserved.



5.05 Activity – Correct Answer

Which ocean had the lowest mean for minimum pressure? Pacific

Which season had the lowest mean for minimum pressure? 2015

```
proc means data=pg1.storm_final maxdec=0 n mean min;
  var MinPressure;
  where Season >=2010;
  class Season Ocean;
  ways 1;
run;
```

Analysis Variable : MinPressure				
Ocean	N Obs	N	Mean	Minimum
Atlantic	128	128	983	908
Indian	144	144	972	910
Pacific	385	382	969	872

Analysis Variable : MinPressure				
Season	N Obs	N	Mean	Minimum
2010	78	78	973	885
2011	80	80	975	920
2012	83	83	973	900
2013	95	95	977	895
2014	85	85	968	900
2015	93	93	965	872
2016	89	86	972	884
2017	54	54	979	908

33

Copyright © SAS Institute Inc. All rights reserved.



continued...

5.06 Activity – Correct Answer

- Run the PROC MEANS step and compare the report and the `wind_stats` table. Are the same statistics in the report and table? What do the first five rows in the table represent?

The statistics are different. The first five rows in the table summarize the entire input table.

Analysis Variable : MaxWindMPH				
BasinName	N Obs	Mean	Median	Maximum
East Pacific	675	82.7629630	75.0000000	213.0000000
North Atlantic	478	82.8953975	75.0000000	190.0000000
North Indian	60	63.6000000	52.0000000	146.0000000
South Indian	594	77.3593750	69.0000000	155.0000000
South Pacific	359	78.0421348	69.0000000	173.0000000
West Pacific	926	79.6511879	81.0000000	144.0000000

BasinName	_TYPE_	_FREQ_	_STAT_	MaxWindMPH
	0	3092	N	3071
	0	3092	MIN	6
	0	3092	MAX	213
	0	3092	MEAN	79.910126994
	0	3092	STD	31.602947926
East Pacific	1	675	N	675
East Pacific	1	675	MIN	17
East Pacific	1	675	MAX	213
East Pacific	1	675	MEAN	79.762962963

36



5.06 Activity – Correct Answer

- Add the following options in the OUTPUT statement and run the program again. How many rows are in the output table?
Six rows, one for each value of BasinName.

```
proc means data=pg1.storm_final noint;
  var MaxWindMPH;
  class BasinName;
  ways 1;
  output out=wind_stats mean=AvgWind max=MaxWind;
run;
```

View SAS documentation for more options to customize the output table.

BasinName	_TYPE_	_FREQ_	AvgWind	MaxWind
1 East Pacific	1	675	82.762962963	213
2 North Atlantic	1	478	82.89539749	190
3 North Indian	1	60	63.6	146
4 South Indian	1	594	77.359375	155
5 South Pacific	1	359	78.042134831	173
6 West Pacific	1	926	79.651187905	144

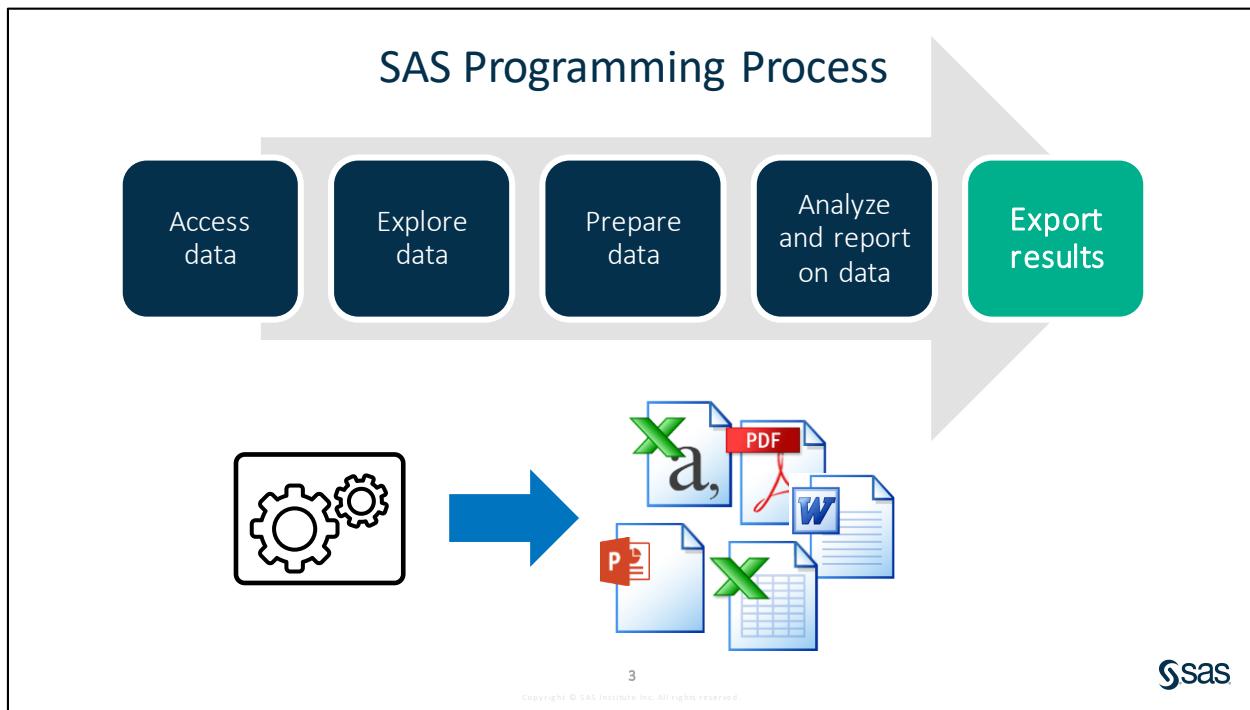
37



Lesson 6 Exporting Results

6.1 Exporting Data	6-3
Demonstration: Exporting Data to an Excel Workbook.....	6-8
6.2 Exporting Reports.....	6-11
Demonstration: Exporting Results to Excel.....	6-15
Demonstration: Exporting Results to PDF.....	6-20
Practice.....	6-23
6.3 Solutions	6-27
Solutions to Practices	6-27
Solutions to Activities and Questions.....	6-29

6.1 Exporting Data



You have clean data and accurate, interesting reports. Now you need to share what you created with others. You realize that not everyone who needs access to your results uses SAS, so you need methods to save the data and reports in formats that are easy to view.

Exporting Data Using Point-and-Click Tools

The screenshot shows two overlapping windows. The top window is the "Save" dialog, which lists several files in a table format. The bottom window is the "Export Table" dialog, which allows selecting a folder and file format for exporting data.

Save Dialog (Top Left):

Name	Date modified	Type	Size
class_birthdate	12/17/2019 11:30 ...	Microsoft Excel C...	1 K
np_traffic	12/17/2019 11:30 ...	Microsoft Excel C...	282 K
pg1m6	12/16/2019 2:40 PM	Compressed (zip)...	19,125 K
storm_damage	12/17/2019 11:30 ...	Microsoft Excel C...	13 K
TSAClaims2002_2017	12/17/2019 11:30 ...	Microsoft Excel C...	34,334 K

Export Table Dialog (Bottom Right):

Folders:

- S:\WINDOWS
- workshop
 - activities
 - data
 - class_birthdate.csv
 - np_traffic.csv
 - storm_damage.csv
 - TSAClaims2002_2017.csv
 - demos

File name: CLASS_BIRTHDATE

File format: CSV (Comma delimited file)(*.csv)

Other options in the "File format" dropdown include: CSV (Comma delimited file)(*.csv), DLM (Space delimited file)(*.dlm), TAB (Tab delimited file)(*.tab), XML (Extensible markup language file)(*.xml), JMP (JMP files)(*.jmp), DBF (dBASE 5.0, IV, III+, and III files)(*.dbf), DTA (Stata file)(*.dta), and XLSX (Microsoft Excel 2010 or 2013 workbook)(*.xlsx).

If you want to export data using a manual process, each of the SAS programming environments includes point-and-click tools for exporting data to various delimited text formats, such as comma-separated values (CSV), tab-delimited values (TAB) and space-delimited (DLM) files.

- In Enterprise Guide, you can start this process by selecting **Share** ⇒ **Output Data** from the toolbar.
- In SAS Studio, you can right-click a table in the Library panel and select **Export**.

Exporting Data Using Code

```

PROC EXPORT DATA=input-table OUTFILE="output-file"  

  <DBMS=identifier><REPLACE>;  

RUN;

```

tells SAS how to
format the output



 Column names are
automatically
written as the first
row of the output
file.

5
Copyright © SAS Institute Inc. All rights reserved.

There are several methods to programmatically export data too. By writing a program to export data, you can easily integrate the export into your overall program to automate the final export step. PROC EXPORT can export a SAS table to a variety of external formats.

The DATA= option specifies the data source. The OUTFILE= option specifies the fully qualified path and file name of the exported data file. The DBMS= option tells SAS how to format the output.

Here are common DBMS identifiers that are included with Base SAS:

- CSV – comma-separated values.
- JMP – JMP files, JMP 7 or later.
- TAB – tab-delimited values.
- DLM – delimited files. The default delimiter is a space. To use a different delimiter, use the DELIMITER= statement.

Here are additional DBMS identifiers that are included with SAS/ACCESS Interface to PC Files:

- XLSX – Microsoft Excel 2007, 2010, and later
- ACCESS – Microsoft Access 2000 and later

Exporting Data Using Code

```
proc export data=sashelp.cars  
    outfile="s:/workshop/output/cars.txt"  
    dbms=tab replace;  
run;
```

Remember that
the path is relative
to the location
of SAS.



6

Copyright © SAS Institute Inc. All rights reserved.

In this code example, PROC EXPORT creates a tab-delimited text file that has column names in the first row of the file. Remember that the path in the OUTFILE= option must be relative to the location of SAS. In other words, if SAS is running on a server, the path must be accessible from the server location.

If SAS Studio or Enterprise Guide were configured to connect to SAS on a remote server, both interfaces provide a method to download files from the remote server to your local machine.

SAS Studio – Select the file in the Files and Folders section of the navigation pane and click **Download** .

Enterprise Guide – Click **Open a task** and select **Browse** ⇒ **Data** ⇒ **Copy Files**.

6.01 Activity

1. Open the **libname.sas** program in the course files folder.
2. Create a macro variable named **outpath** that stores the location of the **output** folder in your course files location.
3. Run the code and save the program.

6.02 Activity

Open **p106a02.sas** from the **activities** folder and perform the following tasks:

1. Complete the PROC EXPORT step to read the **pg1.storm_final** SAS table and create a comma-delimited file named **storm_final.csv**. Use **&outpath** to substitute the path of the **output** folder.
2. Run the program and view the text file:

SAS Studio – Navigate to the **output** folder in the navigation pane, right-click **storm_final.csv**, and select **View File as Text**.

Enterprise Guide – Navigate to the **output** folder in the Servers pane, right-click **storm_final.csv**, and select **Open**.

Exporting Data with a LIBNAME Engine

```

libname myxl xlsx "&outpath/cars.xlsx";
data myxl.asiacars;
  set sashelp.cars;
  where origin='Asia';
run;

libname myxl clear;

```

defines a library to the Microsoft Excel workbook that you are creating

This code extracts data and writes it to the **cars** workbook on a tab named **asiacars**.



11
Copyright © SAS Institute Inc. All rights reserved.

p106d01 

Another easy way to export data is to use a SAS/ACCESS Interface LIBNAME engine. We simply create the data in the desired format right from a SAS process. For example, a DATA step or procedure OUTPUT statement can write results directly to the target data source. I do not have to create a SAS table first and then export the SAS table in a separate step. Of course, you need Write permission to the target destination.

For example, this program uses the SAS/ACCESS Interface to PC File Formats XLSX engine to define a library to an Excel workbook named **cars**. The DATA step references the library and output worksheet named **asiacars**. The code extracts data about cars manufactured in Asia from **sashelp.cars** and writes the result directly into the worksheet **asiacars**.



Exporting Data to an Excel Workbook

Scenario

Use the XLSX LIBNAME engine to export SAS tables to multiple worksheets in an Excel workbook.

Files

- **p106d01.sas**
- **storm_final** – a SAS table that contains one row per storm for the 1980 through 2017 storm seasons. The data was cleaned and prepared previously using the DATA step.

Syntax

```
LIBNAME libref XLSX "path/file.xlsx";  
use libref for output table(s)  
LIBNAME libref CLEAR;
```

Notes

- The XLSX engine requires a license for SAS/ACCESS Interface to PC Files.
- The XLSX engine can read and write data in Excel files.
- To write data to a new or existing Excel workbook, use the LIBNAME statement to assign a libref that points to the Excel file. Use the libref when you name output tables. The table name is the worksheet label in the Excel file.

Demo

1. Open **p106d01.sas** from the **demos** folder and find the **Demo** section of the program. Examine the DATA and PROC MEANS steps and identify the temporary SAS tables that will be created. Highlight the demo program and run the selected code.
2. Add a LIBNAME statement to create a library named **xlout** that points to an Excel file named **southpacific.xlsx** in the **output** folder of the course data.

Note: Use the **outpath** macro variable to substitute the path of the output folder. If you did not define the **outpath** macro variable, run the **libname.sas** program that was completed in Activity 6.01.

```
libname xlout xlsx "&outpath/southpacific.xlsx";
```

3. Modify the DATA and PROC steps to write output tables to the **xlout** library.

```
libname xlout xlsx "&outpath/southpacific.xlsx";
data xlout.South_Pacific;
  set pg1.storm_final;
  where Basin="SP";
run;

proc means data=pg1.storm_final noprint maxdec=1;
  where Basin="SP";
  var MaxWindKM;
  class Season;
  ways 1;
  output out=xlout.Season_Stats n=Count mean=AvgMaxWindKM
        max=StrongestWindKM;
run;
```

4. Add a LIBNAME statement to clear the **xlout** libref. Highlight the demo program and run the selected code.

```
libname xlout clear;
```

5. Open Excel if it is available. Open the **southpacific.xlsx** workbook and confirm that the data is contained in the worksheets that are named **South_Pacific** and **Season_Stats**.

	A	B	C	D	E	F	G
1	Season	Name	Basin	BasinName	OceanCode	Ocean	StormType
2	2017	BART	SP	South Pacific	P	Pacific	
3	2017	DEBBIE	SP	South Pacific	P	Pacific	
4	2017	COOK	SP	South Pacific	P	Pacific	
5	2017	DONNA	SP	South Pacific	P	Pacific	
6	2017	ELLA	SP	South Pacific	P	Pacific	
7	2016	TUNI	SP	South Pacific	P	Pacific	Not Reported
8	2016	ULA	SP	South Pacific	P	Pacific	Not Reported
9	2016	VICTOR	SP	South Pacific	P	Pacific	Not Reported
10	2016	TATIANA	SP	South Pacific	P	Pacific	Not Reported
11	2016	WINSTON	SP	South Pacific	P	Pacific	Not Reported
12	2016	VALE	SP	South Pacific	P	Pacific	Not Reported

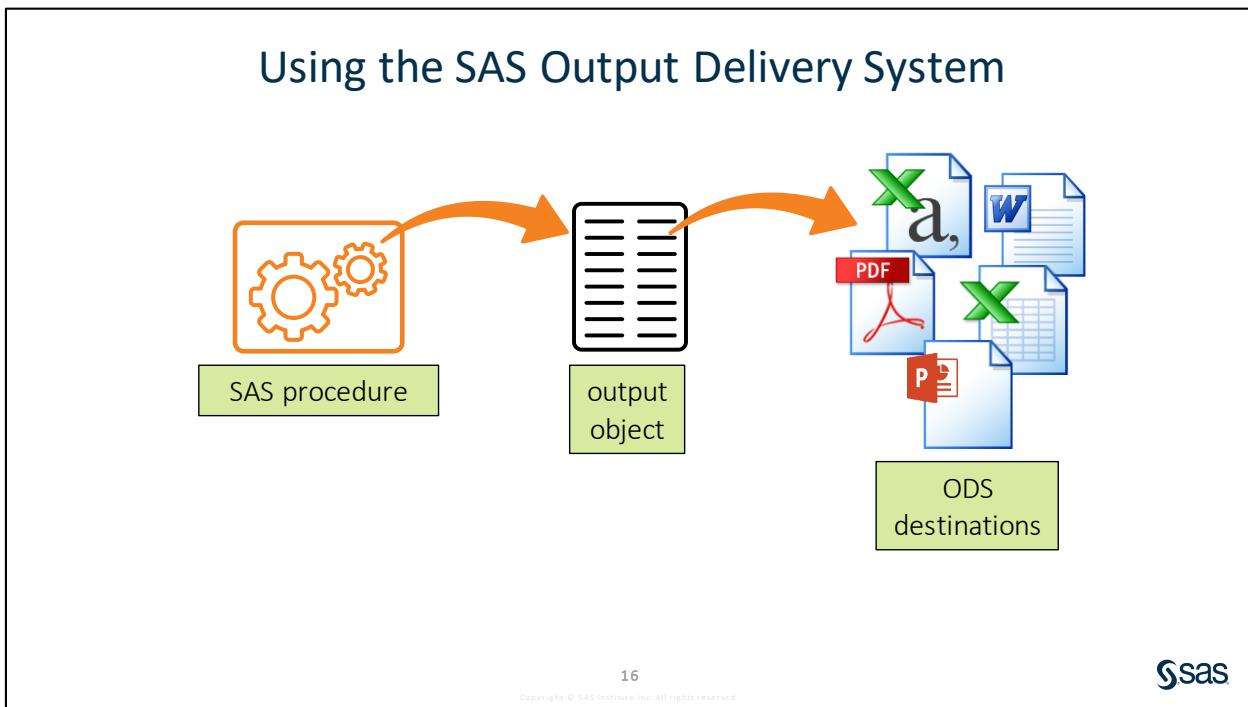
End of Demonstration

6.03 Activity

Open **p106a03.sas** from the **activities** folder and perform the following tasks:

1. Complete the LIBNAME statement using the XLSX engine to create an Excel workbook named **storm.xlsx** in the **output** folder.
2. Modify the DATA step to write the **storm_final** table to the **storm.xlsx** file.
3. After the DATA step, write a statement to clear the library.
4. Run the program and view the log to confirm that **storm.xlsx** was exported with 3092 rows.
5. If possible, open the **storm.xlsx** file. How do dates appear in the **storm_final** workbook?

6.2 Exporting Reports



SAS provides the Output Delivery System (ODS) to create customized output in a variety of formats. In SAS, procedures that generate reports actually generate output objects. These can easily be rendered in one or more output formats that are designed to be viewed in SAS or in other software applications. In ODS terminology, each of these formats is called a *destination*. Some ODS destinations produce very simple output files, such as text files that conform to comma-separated values' standards. Others produce complex output files that are designed to be viewed and manipulated using external software applications. Common destinations of this type include Excel (XLSX), Microsoft Word (RTF), Microsoft PowerPoint (PPTX), and Adobe (PDF). Many other destinations are available in SAS.

Using the SAS Output Delivery System

```
ODS <destination><destination-specifications>;
```

```
/* SAS code that produces output */
```

```
ODS <destination> CLOSE;
```

You can create
different file types
by changing
the destination
in the ODS statement.



Directing output to these destinations is like making a sandwich. The SAS procedure code that creates the output is the “filling” for our sandwich, and the ODS statements preceding and following the output code is the “bread” that makes the output easy to consume outside of SAS. Here are some common destinations:

- EXCEL
- CSVALL (comma-delimited text file)
- RTF (Rich Text Format for viewing in word processors such as Microsoft Word)
- POWERPOINT
- HTML
- PDF

Exporting Output to a CSV File

CSVALL
destination

```
ODS CSVALL FILE="filename.csv";
/* SAS code that produces output */
ODS CSVALL CLOSE;
```

```
ods csvall file=&outpath/cars.csv";
proc print data=sashelp.cars noobs;
  var Make Model Type MSRP MPG_City MPG_Highway;
  format MSRP dollar8.;
run;
ods csvall close;
```

18


Copyright © SAS Institute Inc. All rights reserved.

Exporting Output to a CSV File

```
ods csvall file=&outpath/cars.csv";
proc print data=sashelp.cars noobs;
  var Make Model Type MSRP MPG_City MPG_Highway;
  format MSRP dollar8.;
run;
ods csvall close;
```

```
"Make","Model","Type","MSRP","MPG_City","MPG_Highway"
"Acura","MDX","SUV","$36,945",17,23
"Acura","RSX Type S 2dr","Sedan","$23,820",24,31
"Acura","TSX 4dr","Sedan","$26,990",22,29
"Acura","TL 4dr","Sedan","$33,195",20,28
"Acura","3.5 RL 4dr","Sedan","$43,755",18,24
"Acura","3.5 RL w/Navigation 4dr","Sedan","$46,100",18,
```

Using ODS CSVALL with PROC PRINT enables you to specify the order and format of columns in the CSV file.



19


Copyright © SAS Institute Inc. All rights reserved.

Exporting Results to Excel

```
ODS EXCEL FILE="filename.xlsx" STYLE=style  
OPTIONS(SHEET_NAME='label');  
  
/* SAS code that produces output */  
  
ODS EXCEL CLOSE;
```

By default, the results from each procedure are on separate worksheets in the Excel file.



Sas

20

Copyright © SAS Institute Inc. All rights reserved.

The ODS EXCEL destination provides an enormous amount of flexibility. You can specify a style for the output by using the STYLE= option. There are many different styles that are built in to SAS. You can list additional options in the ODS statement by using the OPTIONS keyword and enclosing option-value pairs in parentheses. The SHEET_NAME= option customizes the tab names in the workbook.

Note: ODS Excel was experimental in SAS 9.4M1 and M2. It is fully supported in SAS 9.4M3 and later releases.



Exporting Results to Excel

Scenario

Use the ODS EXCEL destination to export reports to multiple worksheets in an Excel workbook.

Files

- **p106d02.sas**
- **storm_final** – a SAS table that contains one row per storm for the 1980 through 2017 storm seasons. The data was cleaned and prepared previously using the DATA step.

Syntax

```
ODS EXCEL FILE="filename.xlsx" <STYLE=style>
<OPTIONS(SHEET_NAME='label')>;

/* SAS code that produces output */

<ODS EXCEL OPTIONS(SHEET_NAME='label')>

/* SAS code that produces output */

ODS EXCEL CLOSE;
```

Notes

- The ODS EXCEL destination creates an XLSX file.
- By default, each procedure output is written to a separate worksheet with a default worksheet name. The default style is also applied.
- Use the STYLE= option in the ODS EXCEL statement to apply a different style.
- Use the OPTIONS(SHEET_NAME='label') option in the ODS EXCEL statement to provide a custom label for each worksheet.

Demo

1. Open **p106d02.sas** from the **demos** folder and find the **Demo** section in the program. Add an ODS statement to create an Excel file named **wind.xlsx** in the output folder of the course files. Close the Excel destination at the end of the program. Highlight the demo program and run the selected code.

Note: Use the **outpath** macro variable to substitute the path of the **output** folder. If you did not define the **outpath** macro variable, run the **libname.sas** program that was completed in Activity 6.01.

Note: If you are using Enterprise Guide 8.1 or later, you receive a warning in the log. By default, it uses the graph format **Default**. This allows the Output Delivery System (ODS) to decide on the best graph format. To adjust the default settings, go to **Tools** ⇒ **Results** ⇒ **Graphs** and change the graph format. You can also use the statement GOPTIONS DEV=PNG before the ODS statement.

```
ods excel file="&outpath/wind.xlsx";
title "Wind Statistics by Basin";
...
title;
ods proctitle;
ods excel close;
```

2. Open the Excel file.
 - SAS Studio: Navigate to the **output** folder in the Files and Folders section of the navigation pane. Select **wind.xlsx** and click **Download** .
 - Enterprise Guide: Click **Results** and select the Excel file. Right-click and select **Open**.
3. Examine the Excel workbook. Notice the light blue background in the results that are generated by the default style. Also notice the default spreadsheet names. Close the Excel file.
4. Examine the available style options.
 - SAS Studio or Enterprise Guide: Submit the following program.

```
proc template;
  list styles;
run;
```

- Enterprise Guide only: Select **Tools** \Rightarrow **Style Manager**.
5. Change the style by adding the **STYLE=SASDOCPRINTER** option in the first ODS statement.
 6. Use the **SHEET_NAME=** option in the first ODS EXCEL statement to name the first worksheet **Wind Stats**. Add another ODS EXCEL statement with the **SHEET_NAME=** option before the second TITLE statement and the PROC SGPLOT step. Name the second worksheet **Wind Distribution**. Highlight the demo program and run the selected code. Open the Excel file to view the results.

```
ods excel file="&outpath/wind.xlsx" style=sasdocprinter
  options(sheet_name='Wind Stats');
title "Wind Statistics by Basin";
ods noproctitle;
proc means data=pg1.storm_final min mean median max maxdec=0;
  class BasinName;
  var MaxWindMPH;
run;

ods excel options(sheet_name='Wind Distribution');
title "Distribution of Maximum Wind";
proc sgplot data=pg1.storm_final;
  histogram MaxWindMPH;
  density MaxWindMPH;
run;
title;
ods proctitle;
ods excel close;
```

	A	B	C	D	E	F
1	Analysis Variable : MaxWindMPH					
2	BasinName	N Obs	Minimum	Mean	Median	Maximum
3	East Pacific	675	17	83	75	213
4	North Atlantic	478	23	83	75	190
5	North Indian	60	6	64	52	146
6	South Indian	594	23	77	69	155
7	South Pacific	359	35	78	69	173
8	West Pacific	926	40	80	81	144
9						
10						
11						

End of Demonstration

6.04 Activity

Open **p106a04.sas** from the **activities** folder and perform the following tasks:

1. Add ODS statements to create an Excel file named **pressure.xlsx** in the **output** folder. Be sure to close the ODS location at the end of the program. Run the program and open the Excel file.

SAS Studio: Navigate to the **output** folder in the Files and Folders section of the navigation pane. Select **pressure.xlsx** and click **Download** .

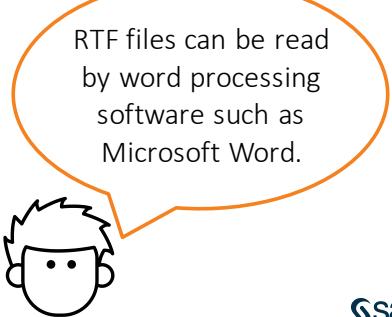
Enterprise Guide: Click the **Results** tab. Then, under **Open with Default Application**, double-click the Excel icon.

2. Add the **STYLE=ANALYSIS** option in the first ODS EXCEL statement. Run the program again and open the Excel file.

Exporting Output to PowerPoint and Microsoft Word

```
ODS POWERPOINT FILE="filename.pptx" STYLE=style;  
/* SAS code that produces output */  
ODS POWERPOINT CLOSE;
```

```
ODS RTF FILE="filename.rtf" STARTPAGE=NO;  
/* SAS code that produces output */  
ODS RTF CLOSE;
```



RTF files can be read by word processing software such as Microsoft Word.

The Output Delivery System also enables you to export reports to common formats that you use in everyday business, such as PowerPoint by using the PowerPoint destination, and Microsoft Word by using the RTF destination. The Rich Text Format (RTF) destination is a software-neutral file type that is made for word processing programs such as Microsoft Word. There are particular options that apply to each of these destinations so that you can customize your output.

6.05 Activity

Open **p106a05.sas** from the **activities** folder and perform the following tasks:

1. Run the program and open the **pressure.pptx** file.
2. Modify the ODS statements to change the output destination to RTF. Change the style to **sapphire**.
3. Add the STARTPAGE=NO option in the first ODS RTF statement to eliminate a page break between the procedure results.
4. Rerun the program and open the **pressure.rtf** file.

Exporting Results to PDF

```
ODS PDF FILE="filename.pdf" STYLE=style
      STARTPAGE=NO PDFTOC=n;
ODS PROCLABEL "label";
/* SAS code that produces output */
ODS PDF CLOSE;
```

The PDF destination has many options for specifying the layout and appearance of your output file.



Finally, let's look at the Portable Document Format (PDF) destination. PDF files are used extensively for reporting because the layout can be precisely controlled, and you can guarantee that the document will look just as you intended it to when the receiver opens it.

In SAS ODS, PDF is one of the PRINTER destinations, meaning that you have a lot of programmatic control over the document's appearance. You can use the PDFTOC= option to control the level of bookmarks that are open. You can use the ODS PROCLABEL statement to label the bookmark for the procedure.



Exporting Results to PDF

Scenario

Use the ODS PDF destination to export reports to a PDF file.

Files

- **p106d03.sas**
- **storm_final** – a SAS table that contains one row per storm for the 1980 through 2017 storm seasons. The data was cleaned and prepared previously using the DATA step.

Syntax

```
ODS PDF FILE="filename.xlsx" STYLE=style STARTPAGE=NO PDFTOC=1;
ODS PROCLABEL "label";
/* SAS code that produces output */
ODS PDF CLOSE;
```

Notes

- The ODS PDF destination creates a PDF file.
- The PDFTOC=*n* option controls the level of the expansion of the table of contents in PDF documents.
- The ODS PROCLABEL statement enables you to change a procedure label.

Demo

1. Open **p106d03.sas** from the **demos** folder and find the **Demo** section of the program. Run the program and open the PDF file to examine the results. Notice that bookmarks are created, and they are linked to each procedure's output.

Note: Use the **outpath** macro variable to substitute the path of the output folder. If you did not define the **outpath** macro variable, run the **libname.sas** program that was completed in Activity 6.01.

2. Add the STARTPAGE=NO option to eliminate page breaks between procedures. Add the STYLE=JOURNAL option.

```
ods pdf file="&outpath/wind.pdf" startpage=no style=journal;
```

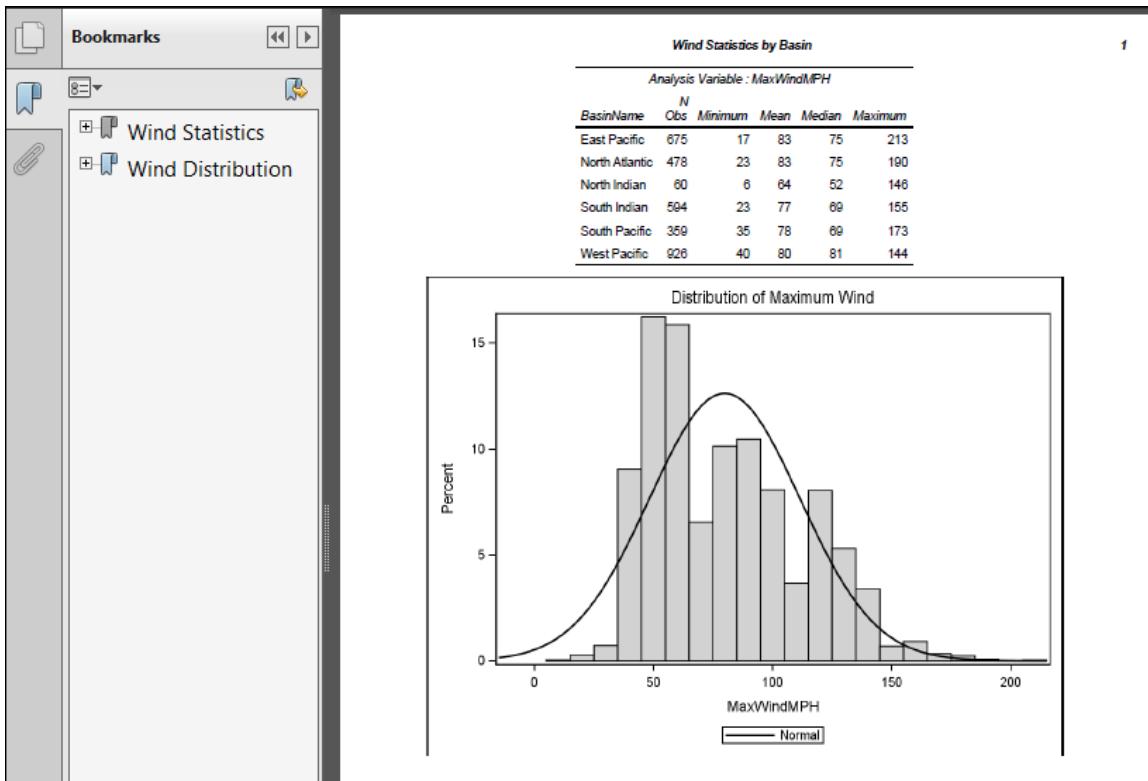
3. To customize the PDF bookmarks, add the PDFTOC=1 option to ensure that bookmarks are expanded only one level when the PDF is opened. To customize the bookmark labels, add the ODS PROCLABEL statement before each PROC with custom text. Run the program and open the PDF file.

```
ods pdf file="&outpath/wind.pdf" startpage=no style=journal
      pdftoc=1;
ods noproctitle;

ods proclabel "Wind Statistics";
title "Wind Statistics by Basin";
proc means data=pg1.storm_final min mean median max maxdec=0;
  class BasinName;
  var MaxWindMPH;
run;

ods proclabel "Wind Distribution";
title "Distribution of Maximum Wind";
proc sgplot data=pg1.storm_final;
  histogram MaxWindMPH;
  density MaxWindMPH;
run;
title;

ods proctitle;
ods pdf close;
```



End of Demonstration

Beyond SAS Programming 1

What if you want to ...

... learn more about working with SAS and Excel?

- Take the [Exporting SAS Data Sets and Creating ODS Files for Microsoft Excel](#) course.

... look up additional options for exporting data or results?

- View the following Help pages:
 - [Base SAS EXPORT procedure](#)
 - [SAS Output Delivery System: User's Guide](#)
 - [SAS/ACCESS Interface to PC Files: Reference](#)

... learn to create and customize reports with ODS?

- Take the [SAS Report Writing 1: Essentials](#) course.
- Explore the [SAS Output Delivery System resource page](#).

29
Copyright © SAS Institute Inc. All rights reserved.



Links

- Take the [Exporting SAS Data Sets and Creating ODS Files for Microsoft Excel](#) course.
- View the following Help pages:
 - [Base SAS EXPORT Procedure](#)
 - [SAS Output Delivery System: User's Guide](#)
 - [SAS/ACCESS Interface to PC Files: Reference](#)
- Take the [SAS Report Writing 1: Essentials](#) course.
- Explore the [SAS Output Delivery System resource page](#).



Practice

If you restarted your SAS session, open and submit the **libname.sas** program in the course files.

Level 1

1. Creating an Excel File Using ODS EXCEL

Create an Excel workbook named **StormStats.xlsx** that includes the results of SAS procedures. Customize the names of the Excel worksheets.

- a. Open **p106p01.sas** from the **practices** folder. Before the PROC MEANS step, add an ODS EXCEL statement to do the following:
 - 1) Write the output file to “**&outpath/StormStats.xlsx**”.

Note: If you did not define the **outpath** macro variable, run the **libname.sas** program that was completed in Activity 6.01.

 - 2) Set the style for the Excel file to **snow**.
 - 3) Set the sheet name for the first tab to **South Pacific Summary**.
- b. Turn off the procedure titles and report titles at the start of the program. Turn the procedure titles on at the end of the program.
- c. Immediately before the PROC PRINT step, add an ODS EXCEL statement to set the sheet name to **Detail**.
- d. At the end of the program, add an ODS EXCEL statement to close the Excel destination.
- e. Submit the program. If possible, open the **StormStats.xlsx** workbook in Excel.

	A	B	C	D	
1	Analysis Variable : Wind Wind(MPH)				
2	Season	N Obs	Median	Maximum	
3	2014	504	30	120	
4	2015	257	50	135	
5	2016	371	50	150	
6					
7					
8					
9					
	◀ ▶	South Pacific Summary	Detail	Detail 2	Detail 3

Level 2

2. Creating a Word Document with ODS RTF

Generate an RTF file that can be opened in Microsoft Word. The file should include the results of three procedures and use different styles to change the appearance.

- Open **p106p02.sas** from the **practices** folder. Modify the program to write the output file to **&outpath/ParkReport.rtf**. Set the style for the output file to **Journal** and remove page breaks between procedure results. Suppress the printing of procedure titles.

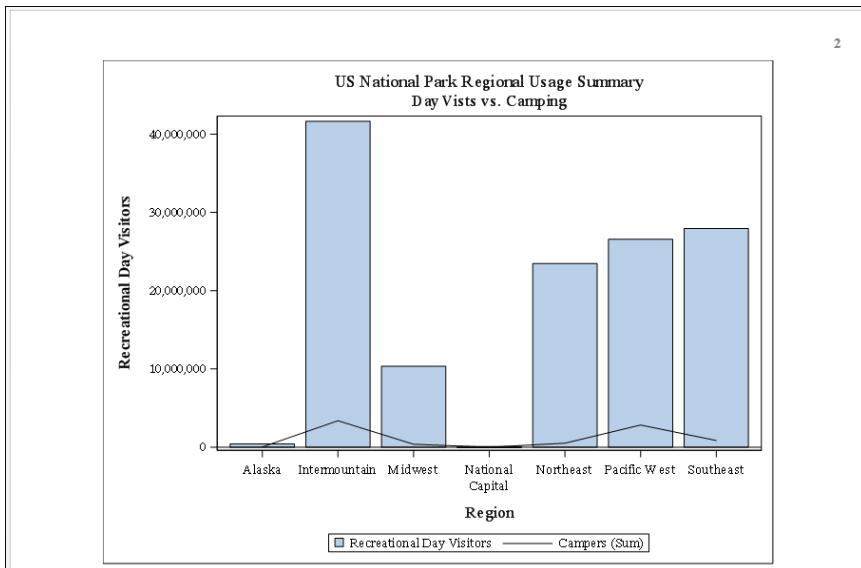
Note: If you did not define the **outpath** macro variable, run the **libname.sas** program that was completed in Activity 6.01.

- Run the program. Open the output file in Microsoft Word. Notice that the Journal style is applied to the results, but the graph is now gray scale instead of color. Also notice that the date and time the program ran is printed in the upper right corner of the page. Close Microsoft Word.
- Modify your SAS program so that both tables are created using the Journal style, but the graph is created using the SASDOCPRINTER style.

Note: An ODS destination statement enables you to specify a style without requiring you to redefine the output file location.

- Add an OPTIONS statement with the NODATE option at the beginning of the program to suppress the date and time in the RTF file. Restore the option for future submissions by adding an OPTIONS statement with the DATE option at the end of the program.
- Run the program. Open the new output file using Microsoft Word. Ensure that the style for both tables is the same, but that the graph is now displayed in color. Close the report.

US National Park Regional Usage Summary					
Region	Frequency	Percent			
Alaska	6	4.44			
Intermountain	52	38.52			
Midwest	18	13.33			
National Capital	1	0.74			
Northeast	13	9.63			
Pacific West	23	17.04			
Southeast	22	16.30			
Region	Variable	Label	Mean	Median	Maximum
Alaska	DayVisits	Recreational Day Visitors	66304	15250	346534
	Campers		4212	4282	7050
Intermountain	DayVisits	Recreational Day Visitors	801061	228679	5969811
	Campers		64890	3358	798361
Midwest	DayVisits	Recreational Day Visitors	573976	133680	2423390
	Campers		20471	18	87152
National Capital	DayVisits	Recreational Day Visitors	67489	67489	67489
	Campers		0	0	0
Northeast	DayVisits	Recreational Day Visitors	1804742	1197931	4812930
	Campers		38730	0	229674
Pacific West	DayVisits	Recreational Day Visitors	1154931	756344	5028868
	Campers		123113	25516	1084164
Southeast	DayVisits	Recreational Day Visitors	1269815	488705	11312786
	Campers		38662	2579	411603



Challenge

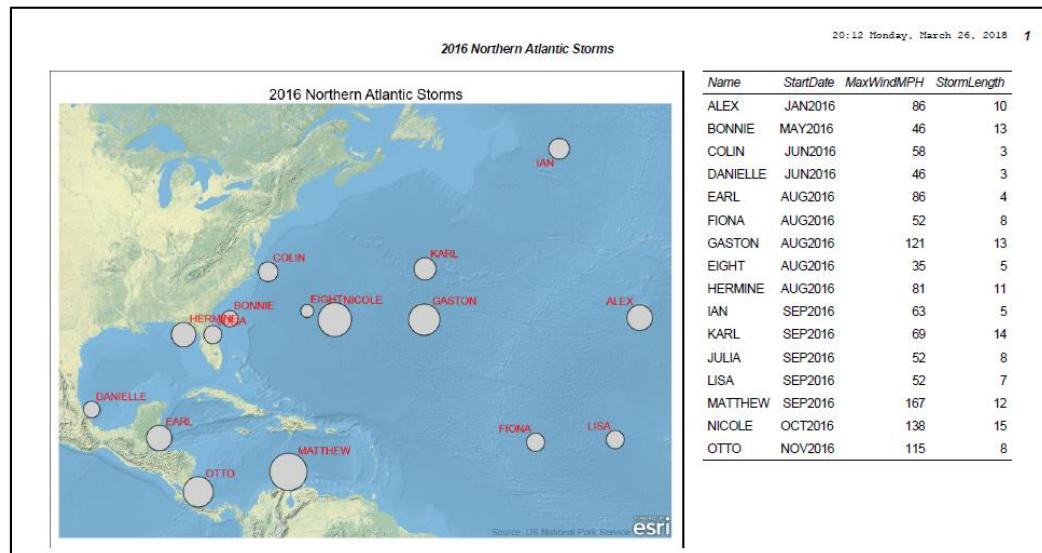
3. Creating a Landscape Report with ODS PDF

Generate a PDF document in landscape orientation. Print a report and map side by side.

- a. Open **p106p03.sas** from the **practices** folder. Run the program and examine the output. The program produces a table and map for North Atlantic region storms in the 2016 season.
- b. Modify the program to produce a PDF file named **StormSummary.pdf** in the **output** folder in the course files. Set the output style to **Journal**.
- c. Use SAS Help to find a SAS system option that changes the page layout to landscape.
- d. Use SAS Help to learn about the ODS LAYOUT GRIDDED statement as a way that you can control the layout of multiple result objects. Force the results to be arranged in one row and two columns.
- e. Reset the system option at the end of the program so that future results have a portrait layout.

- f. Run the program and open the **StormSummary.pdf** file to confirm the results.

Note: SAS Studio generates a warning in the log because the wrapper code is creating an RTF file behind the scenes. LAYOUT is not supported in RTF. The warning can be ignored because it does not impact the PDF results.



End of Practices

6.3 Solutions

Solutions to Practices

1. Creating an Excel File Using ODS EXCEL

```
ods excel file="&outpath/StormStats.xlsx"
  style=snow
  options(sheet_name='South Pacific Summary');
ods noproctitle;
title;

proc means data=pg1.storm_detail maxdec=0 median max;
  class Season;
  var Wind;
  where Basin='SP' and Season in (2014,2015,2016);
run;

ods excel options(sheet_name='Detail');

proc print data=pg1.storm_detail noobs;
  where Basin='SP' and Season in (2014,2015,2016);
  by Season;
run;

ods excel close;
ods proctitle;
```

2. Creating a Word Document with ODS RTF

```
ods rtf file="&outpath/ParkReport.rtf" style=Journal startpage=no;

ods noproctitle;
options nodate;
title "US National Park Regional Usage Summary";

proc freq data=pg1.np_final;
  tables Region / nocum;
run;

proc means data=pg1.np_final mean median max nonobs maxdec=0;
  class Region;
  var DayVisits Campers;
run;
```

```

ods rtf style=SASDocPrinter;
title2 'Day Visits vs. Camping';
proc sgplot data=pg1.np_final;
    vbar Region / response=DayVisits;
    vline Region / response=Campers;
run;
title; ods proctitle;
ods rtf close;
options date;

```

3. Creating a Landscape Report with ODS PDF

```

options orientation=landscape;
ods pdf file="&outpath/StormSummary.PDF" style=Journal
nobookmarkgen;
title1 "2016 Northern Atlantic Storms";

ods layout gridded columns=2 rows=1;
ods region;
proc sgmap plotdata=pg1.storm_final;
    *openstreetmap;
    esrimap
        url='http://services.arcgisonline.com/arcgis/rest/services/
             World_Physical_Map';
    bubble x=lon y=lat size=maxwindmph / datalabel=name
        datalabelattrs=(color=red size=8);
    where Basin='NA' and Season=2016;
    keylegend 'wind';
run;

ods region;
proc print data=pg1.storm_final noobs;
    var name StartDate MaxWindMPH StormLength;
    where Basin="NA" and Season=2016;
    format StartDate monyy7. ;
run;

ods layout end;
ods pdf close;
options orientation=portrait;

```

End of Solutions

Solutions to Activities and Questions

6.01 Activity – Correct Answer

1. Open the `libname.sas` program in the course files folder.
2. Create a macro variable named `outpath` that stores the location of the `output` folder in your course files location.

```
%let outpath=s:/workshop/output;
```

3. Run the code and save the program.

8



6.02 Activity – Correct Answer

```
proc export data=pg1.storm_final
            outfile="&outpath/storm_final.csv"
            dbms=csv;
run;
```

storm_final.csv - Notepad

File Edit Format View Help

Season	Name	Basin	BasinName	OceanCode	Ocean	StormType	MaxWindMPH	MaxWindKM	MinPressure	StartDate	EndDate	StormLength	Lat	Lon
2017	ALFRED	SI	South Indian	I	Indian	,50,80,994,16FEB2017,22FEB2017,6,,								
2017	BART	SP	South Pacific	P	Pacific	,45,72,994,19FEB2017,22FEB2017,3,,								
2017	BLANCHE	SI	South Indian	I	Indian	,65,105,984,02MAR2017,07MAR2017,5,,								
2017	CALEB	SI	South Indian	I	Indian	,50,80,989,23MAR2017,27MAR2017,4,,								
2017	DEBBIE	SP	South Pacific	P	Pacific	,,120,193,943,23MAR2017,30MAR2017,7,,								
2017	ERNIE	SI	South Indian	I	Indian	,,140,225,922,05APR2017,18APR2017,5,,								
2017	COOK	SP	South Pacific	P	Pacific	,,100,161,961,06APR2017,11APR2017,5,,								
2017	MAARUTHA	NT	North Indian	T	Indian	,45,72,996,15APR2017,17APR2017,?								

10



6.03 Activity – Correct Answer

```
libname xl_lib xlsx "&outpath/storm.xlsx";
data xl_lib.storm_final;
  set pg1.storm_final;
  drop Lat Lon Basin OceanCode;
run;

libname xl_lib clear;
```

Dates are automatically formatted.

StartDate	EndDate
16-Feb-17	22-Feb-17
19-Feb-17	22-Feb-17
2-Mar-17	7-Mar-17
23-Mar-17	27-Mar-17
23-Mar-17	30-Mar-17
5-Apr-17	10-Apr-17

14

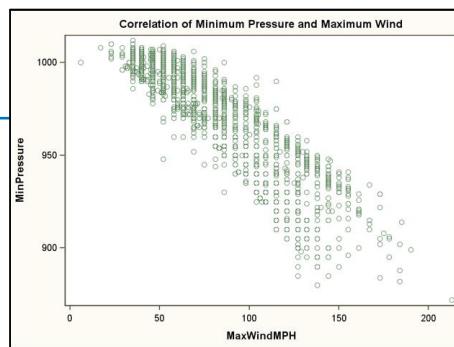
Copyright © SAS Institute Inc. All rights reserved.



6.04 Activity – Correct Answer

```
ods excel file="&outpath/pressure.xlsx" style=analysis;
title "Minimum Pressure Statistics by Basin";
ods noproctitle;
...
ods excel close;
```

A	B	C	D	E	
Analysis Variable : MinPressure					
2	BasinName	N Obs	Mean	Median	Minimum
3	East Pacific	675	979	987	872
4	North Atlantic	478	980	988	882
5	North Indian	60	983	989	920
6	South Indian	594	965	974	895
7	South Pacific	359	967	975	884
8	West Pacific	926	962	965	880



23

Copyright © SAS Institute Inc. All rights reserved.

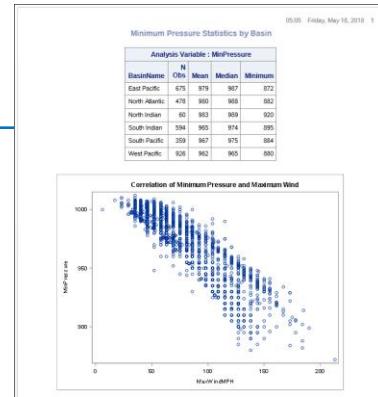


6.05 Activity – Correct Answer

```
ods rtf file="&outpath/pressure.rtf" style=sapphire  
startpage=no;  
  
title "Minimum Pressure Statistics by Basin";  
ods noproctitle;  
...  
  
ods rtf close;
```



The STARTPAGE= option controls page breaks in the file.



26

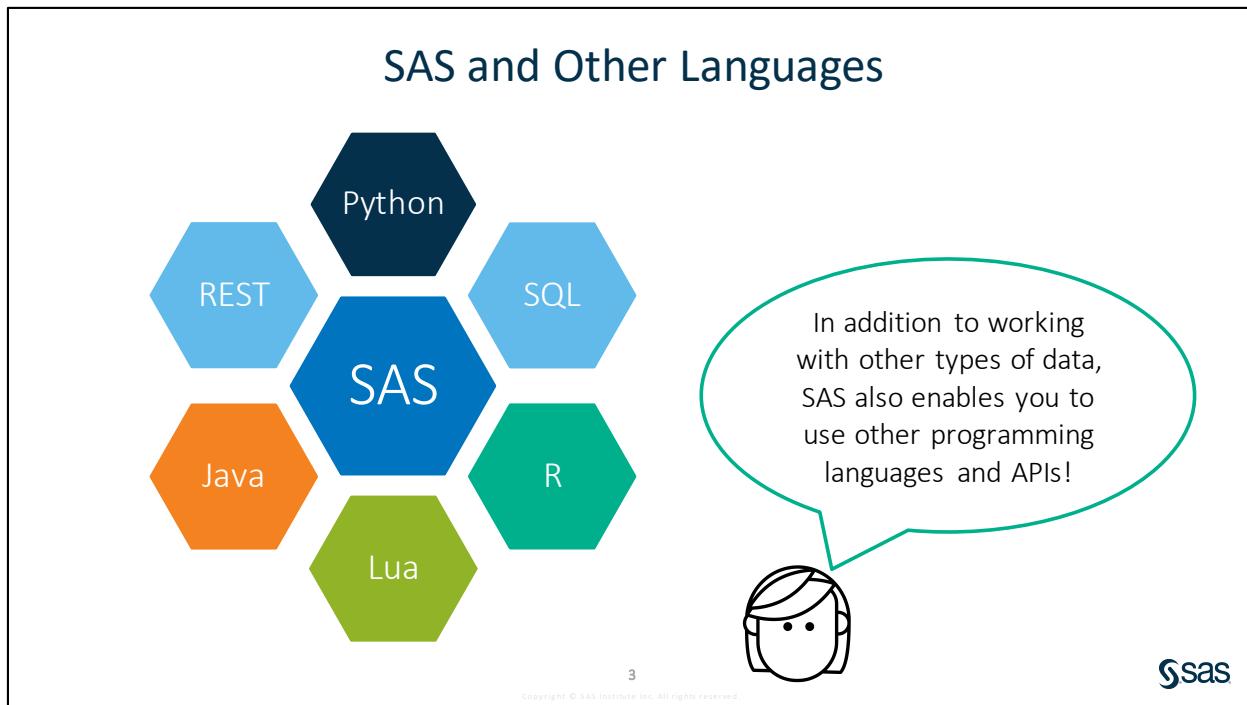
Copyright © SAS Institute Inc. All rights reserved.



Lesson 7 Using SQL in SAS®

7.1	Using Structured Query Language (SQL) in SAS	7-3
	Demonstration: Reading and Filtering Data with SQL.....	7-9
7.2	Joining Tables Using SQL in SAS	7-13
	Demonstration: Joining Tables with PROC SQL	7-16
7.3	Solutions	7-23
	Solutions to Activities and Questions.....	7-23

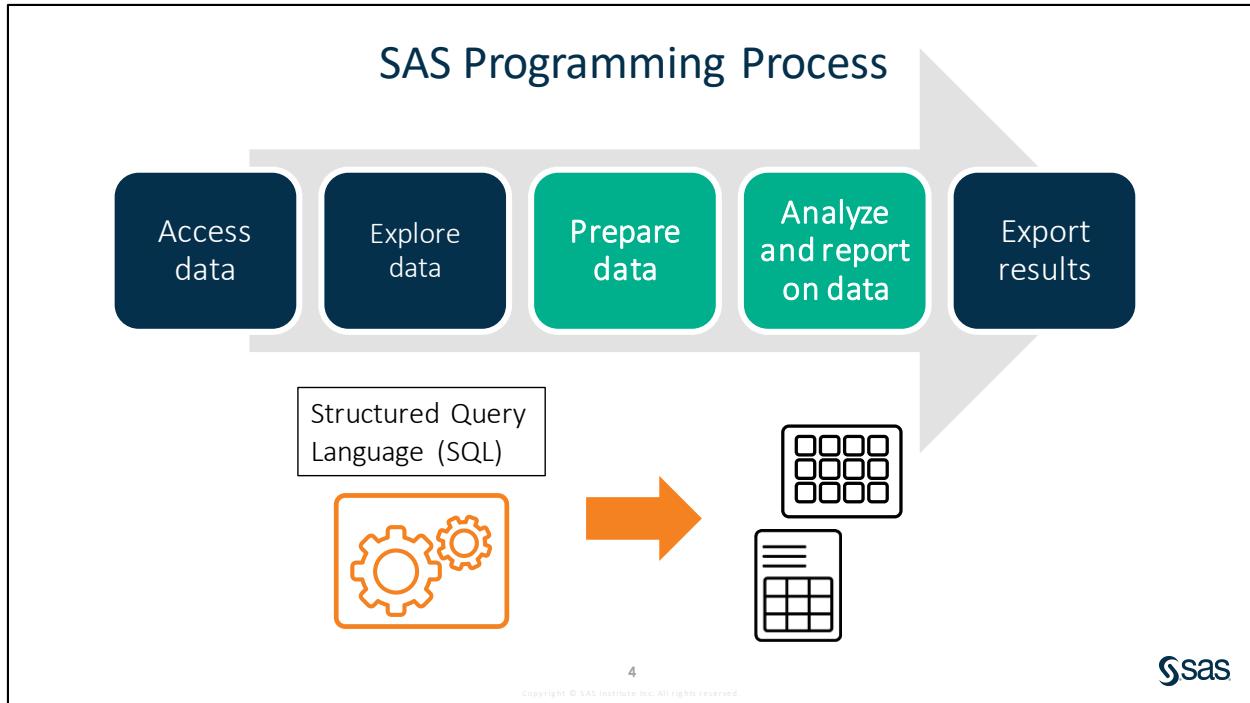
7.1 Using Structured Query Language (SQL) in SAS



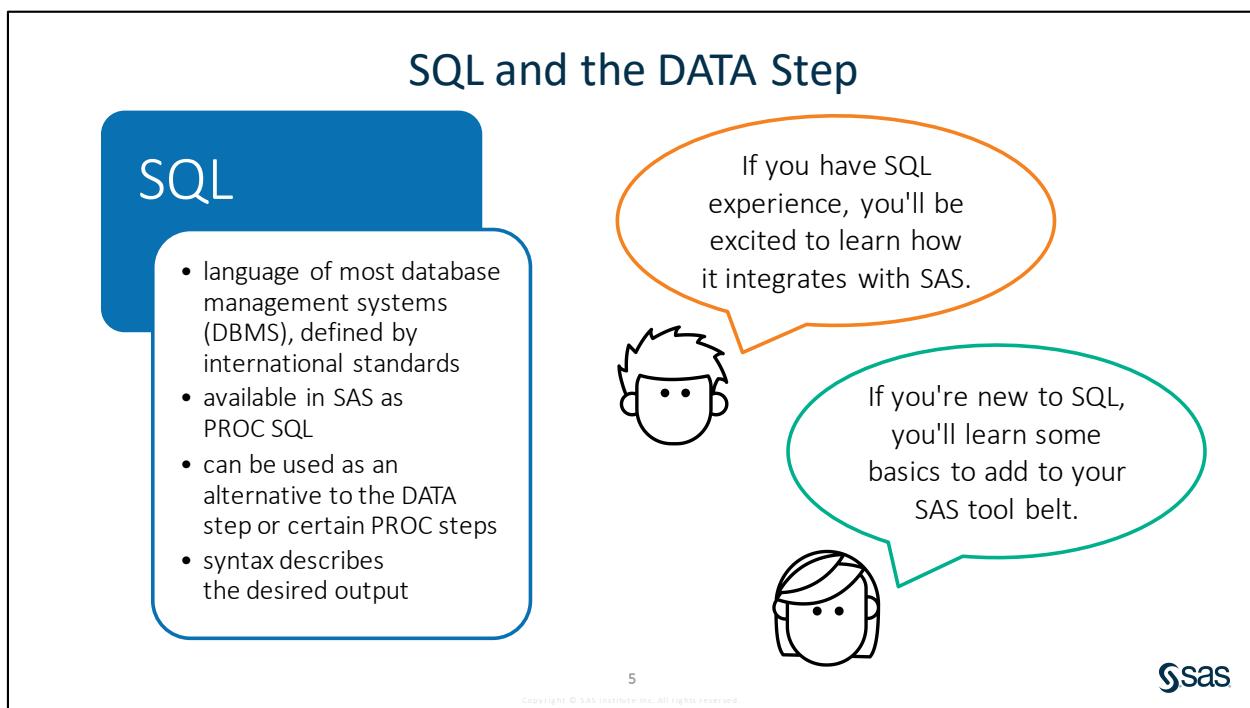
One of the great strengths of SAS has always been the ability to integrate with other types of data. We have seen in this course how SAS integrates with Excel and other Microsoft Office products. You can also read and write data from many other databases that are not part of SAS, including Oracle and Hadoop.

In addition to enabling you to use data from other sources, SAS also supports other common programming languages and APIs. You can take advantage of your knowledge and the strengths of these other languages in the code that you submit in the SAS Platform.

To learn more about how these languages and APIs can be integrated on the SAS Platform, visit <http://developer.sas.com>.

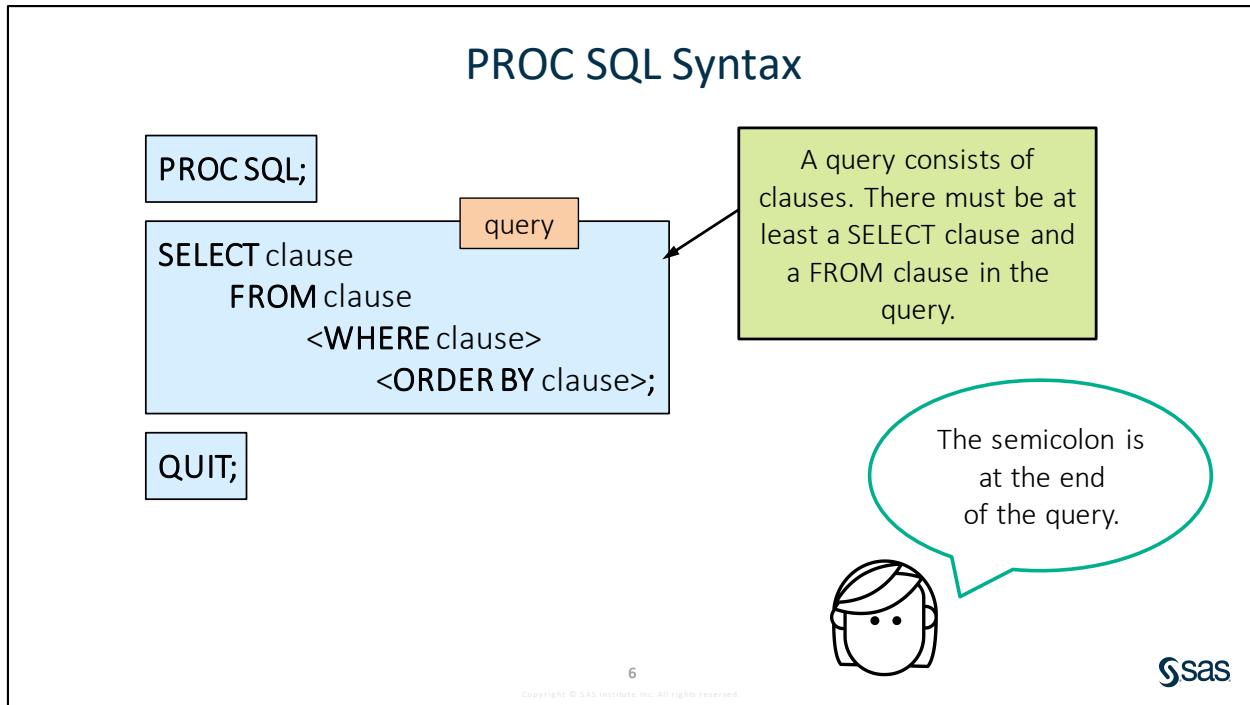


Structured Query Language (SQL) is a common language that is used by many programmers in a wide variety of software. SAS enables you to write SQL code as part of a SAS program. It is likely that you will encounter SQL as you progress as a SAS programmer, so it is important to understand how SQL can be a beneficial tool, and how it compares to the SAS code that was written.



The SQL language is available to use in Base SAS. Because SQL is a separate language, it is implemented in SAS as a procedure. Many programmers who are new to SAS will have prior experience with SQL. This provides an easy, familiar entry point for programming on the SAS Platform.

There are two procedures to choose from for executing SQL in Base SAS: PROC SQL and PROC FedSQL. Each has different extensions and strengths. PROC SQL is more tightly integrated with the SAS System and has several unique extensions that are useful when processing on the SAS Platform. PROC FedSQL is written to a more modern SQL ANSI standard, and it is more ANSI compliant, which means that it has fewer SAS extensions. Because PROC SQL has been available longer, it is more commonly encountered in existing SAS code, so PROC SQL was chosen for executing SQL programs in this class.



The PROC SQL statement invokes the SQL language processor, and subsequent statements are interpreted and executed as SQL until a QUIT statement is encountered.

SELECT is the most commonly used SQL statement and is usually referred to as a *query*. A query consists of clauses that describe the desired result. At a minimum, a query must specify a list of column names to retrieve in the SELECT clause and the name of the table that contains the columns in the FROM clause. By default, an SQL query creates a report.

Note: Each SQL statement executes immediately and independently.

Using PROC SQL to Read Data

```
PROC SQL;
SELECT col-name, col-name
  FROM input-table;
QUIT;
```

```
proc sql;
select Name, Age, Height, Birthdate format=date9.
      from pg1.class_birthdate;
quit;
```

columns that you want to select

table that contains the columns

Name	Age	Height	Birthdate
Alfred	14	69	26OCT2004
Alice	13	56.5	16NOV2005
Barbara	13	65.3	15JAN2005
Carol	14	62.8	04JUL2004
Henry	14	63.5	01DEC2004

7

Copyright © SAS Institute Inc. All rights reserved.

p107d01



This simple query selects columns from the **class_birthdate** table and generates a report. The SELECT clause specifies the columns that you want to appear in the result, and the FROM clause specifies the table containing the source data. Notice that lists, such as column names, are always separated with commas. Also note the syntax applying a format to the **Birthdate** column. Although this is not standard SQL syntax, this SAS extension to the SQL language makes it easier to create more useful and polished reports.

Using PROC SQL to Read Data

expression AS col-name

Computed columns can be included in the SELECT clause.

```
proc sql;
select Name, Age, Height*2.54 as HeightCM format=5.1,
      Birthdate format=date9.
      from pg1.class_birthdate;
quit;
```

Name	Age	HeightCM	Birthdate
Alfred	14	175.3	26OCT2004
Alice	13	143.5	16NOV2005
Barbara	13	165.9	15JAN2005
Carol	14	159.5	04JUL2004
Henry	14	161.3	01DEC2004

8

Copyright © SAS Institute Inc. All rights reserved.

p107d01



7.01 Activity

Open p107a01.sas from the activities folder.

1. What are the similarities and differences in the syntax of the two steps?
2. Run the program. What are the similarities and differences in the results?

```
proc print data=pg1.class_birthdate;
  var Name Age Height Birthdate;
  format Birthdate date9.;
run;
```

```
proc sql;
select Name, Age, Height*2.54 as HeightCM format=5.1,
       Birthdate format=date9.
  from pg1.class_birthdate;
quit;
```

9



Filtering Rows Using the WHERE Clause

WHERE expression

```
proc sql;
select Name, Age, Height, Birthdate format=date9.
  from pg1.class_birthdate
  where age > 14;
quit;
```

Name	Age	Height	Birthdate
Janet	15	62.5	02APR2003
Mary	15	66.5	26MAR2003
Philip	16	72	21NOV2002
Ronald	15	67	14OCT2003
William	15	66.5	28DEC2003

11

p107d01



The WHERE clause is used to subset rows in the query. The same WHERE syntax that worked in other SAS procedures and the DATA step works in SQL too. However, remember that the WHERE expression is not a separate statement in SQL, but instead it is a clause added to the SELECT statement. Only those rows from the input table that meet the criterion provided are included in the result.

Sorting the Output with the ORDER BY Clause

ORDER BY *col-name* <DESC>

```
proc sql;
  select Name, Age, Height, Birthdate format=date9.
    from pg1.class_birthdate
    where age > 14
    order by Height desc;
quit;
```

The default sort
order is ascending.

Name	Age	Height	Birthdate
Philip	16	72	21NOV2002
Ronald	15	67	14OCT2003
Mary	15	66.5	26MAR2003
William	15	66.5	28DEC2003
Janet	15	62.5	02APR2003

12

Copyright © SAS Institute Inc. All rights reserved.

p107d01



In traditional SAS syntax, if you want a report produced in a particular order, you must perform two separate steps. First sort the data, and then execute a reporting procedure. In SQL, we can do it all in one query. We can add an ORDER BY clause to describe the order in which we want the results arranged. If you want the rows ordered with the tallest person listed first (descending order), you would add the DESC keyword after the column name in the ORDER BY clause.



Reading and Filtering Data with SQL

Scenario

Use PROC SQL to select columns and filter rows from an existing SAS table and create a report.

Files

- **p107d01.sas**
- **storm_final** - a SAS table that contains one row per storm for the 1980 through 2017 storm seasons. The data was cleaned and prepared previously using the DATA step.

Syntax

```
PROC SQL;
  SELECT col-name, col-name FORMAT=fmt
    FROM input-table
    WHERE expression
    ORDER BY col-name <DESC>;
QUIT;

New column in SELECT list:
expression AS col-name
```

Notes

- PROC SQL creates a report by default.
- The SELECT statement describes the query. After the SELECT keyword, list columns to include in the results, separated by commas.
- Computed columns can be included in the SELECT clause.
- The FROM clause lists one or more input tables.
- The ORDER BY clause arranges rows based on the listed columns. The default order is ascending. Use DESC after a column name to reverse the sort sequence.
- PROC SQL ends with a QUIT statement.

Demo

1. Open **p107d01.sas** from the **demos** folder and find the **Demo** section of the program. Add a SELECT statement to retrieve all columns from **pg1.storm_final**. Highlight the step and run the selected code. Examine the log and results.

```
proc sql;
  select *
    from pg1.storm_final;
quit;
```

2. Modify the query to retrieve only the **Season**, **Name**, **StartDate**, and **MaxWindMPH** columns. Format **StartDate** with MMDDYY10. Highlight the step and run the selected code.

```
proc sql;
  select Season, Name, StartDate format=mmddyy10., MaxWindMPH
    from pg1.storm_final;
quit;
```

3. Modify **Name** in the SELECT clause to convert the values to proper case.

```
proc sql;
  select Season, propcase(Name) as Name,
         StartDate format=mmddyy10., MaxWindMPH
    from pg1.storm_final;
quit;
```

4. Add a WHERE clause to include storms during or after the 2000 season with **MaxWindMPH** greater than 156.
5. Add an ORDER BY clause to arrange rows by descending **MaxWindMPH**, and then by **Name**.
6. Add TITLE statements to describe the report. Highlight the step and run the selected code.

```
title "International Storms since 2000";
title2 "Category 5 (Wind>156)";
proc sql;
  select Season, propcase(Name) as Name,
         StartDate format=mmddyy10., MaxWindMPH
    from pg1.storm_final
   where MaxWindMPH > 156 and Season >= 2000
     order by MaxWindMPH desc, Name;
quit;
title;
```

International Storms since 2000 Category 5 (Wind>156)

Season	Name	StartDate	MaxWindMPH
2015	Patricia	10/20/2015	213
2017	Irma	08/30/2017	185
2005	Wilma	10/15/2005	184
2009	Rick	10/15/2009	178
2005	Rita	09/18/2005	178
2017	Maria	09/16/2017	175

End of Demonstration

7.02 Activity

Open **p107a02.sas** from the **activities** folder and perform the following tasks:

1. Complete the SQL query to display **Event** and **Cost** from **pg1.storm_damage**. Format the values of **Cost**.
2. Add a new column named **Season** that extracts the year from **Date**.
3. Add a WHERE clause to return rows where **Cost** is greater than 25 billion.
4. Add an ORDER BY clause to arrange rows by descending **Cost**.

Which storm had the highest cost?

```
PROC SQL;
  SELECT col-name, col-name <FORMAT=fmt.>, expression AS col-name
    FROM input-table
      WHERE expression
        ORDER BY col-name<DESC>;
QUIT;
```

14

Copyright © SAS Institute Inc. All rights reserved.



Creating Tables in SQL

CREATE TABLE *table-name* **AS**

```
proc sql;
  create table work.myclass as
    select Name, Age, Height
      from pg1.class_birthdate
        where age > 14
          order by Height desc;
quit;
```

Adding CREATE TABLE at the beginning of the query turns a report into a table.



16

Copyright © SAS Institute Inc. All rights reserved.



Deleting Tables in SQL

```
DROP TABLE table-name;
```

```
proc sql;  
  drop table work.myclass;  
quit;
```

This is helpful if you
are working with
DBMS tables that don't
allow you to overwrite
existing tables.



sas

17

Copyright © SAS Institute Inc. All rights reserved.

For those writing SQL code for SAS to process in other database environments, you might need to drop or delete a table before updating it. If you have appropriate permission to make such changes within the database, you can use the DROP TABLE statement.

7.2 Joining Tables Using SQL in SAS

Creating Inner Joins in SQL

class_update

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
David	M	11	55.3	73
Henry	M	14	63.5	102.5

class_teachers

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith
Carol	8	Thomas
Henry	8	Thomas

class_combine

Only students in both input tables are included.

Name	Sex	Age	Height	Weight	Grade	Teacher
Alfred	M	14	69	112.5	8	Thomas
Alice	F	13	56.5	84	7	Evans
Barbara	F	13	65.3	98	6	Smith
Henry	M	14	63.5	102.5	8	Thomas

19

Copyright © SAS Institute Inc. All rights reserved.



Joining tables is a very common requirement when working with data. There are multiple methods available in SAS to join tables. The most common are SQL and the DATA step. In this course, we introduce the SQL inner join. The SAS Programming 2: Data Manipulation Techniques course addresses the DATA step merge.

In this example, we have information about students in the **class_update** table, and each student's assigned grade and teacher in the **class_teachers** table. Notice that the **Name** column is common in both tables. We would like to join the tables so that all information for each student is contained in a single result. An inner join will create a new report or table that includes students found in both tables. Notice that *David* is in only **class_update**, and *Carol* is in only **class_teachers**, so they are not included in the inner join result.

Creating Inner Joins in SQL

```
FROM table1 INNERJOIN table2
ON table1.column = table2.column
```

```
proc sql;
select Grade, Age, Teacher
  from pg1.class_update inner join pg1.class_teachers
    on class_update.Name = class_teachers.Name;
quit;
```

20

Copyright © SAS Institute Inc. All rights reserved.

p107d02



What is the syntax required to combine the matching rows from two tables? We can modify the FROM clause to add INNER JOIN, followed by the second table.

Creating Inner Joins in SQL

```
FROM table1 INNERJOIN table2
ON table1.column = table2.column
```

```
proc sql;
select Grade, Age, Teacher
  from pg1.class_update inner join pg1.class_teachers
    on class_update.Name = class_teachers.Name;
quit;
```

matching
criteria

21

Copyright © SAS Institute Inc. All rights reserved.

p107d02



Following the table names, this join syntax requires an ON clause to describe the criteria for matching rows in the tables. Omitting the ON clause produces a syntax error.

The join in this example is an example of a specific type of inner join, referred to as an **equijoin**, where only rows with identical values in the **Name** column produce a match. The ON condition could also use other comparison operators, such as greater than or less than.

Although not illustrated in this course, outer joins enable you to include nonmatching rows in the results. This is accomplished simply by changing the keyword INNER to OUTER (all nonmatching rows) or RIGHT or LEFT (all rows from one table).

Qualifying Table Names

```
proc sql;
  select class_update.Name, Grade, Age, Teacher
    from pg1.class_update inner join pg1.class_teachers
      on class_update.Name = class_teachers.Name;
quit;
```

Because **Name** occurs in both tables, you must use the table prefix to indicate which column you want to select.



p107d02 

Note that the **Name** column is prefixed by one of the table names. This is known as *qualifying* the column names, and it is necessary when you have columns with the same name from more than one table. Qualifying the column name avoids creating an ambiguous column reference, where SAS does not know which **Name** column to read.



Joining Tables with PROC SQL

Scenario

Use PROC SQL to perform an inner join between two tables.

Files

- **p107d02.sas**
- **storm_summary** – a SAS table that contains one row per storm for the 1980 through 2016 storm seasons
- **storm_basincodes** – a SAS table that includes each two-letter basin code and the corresponding full basin name

Syntax

```
PROC SQL;
  SELECT col-name, col-name
    FROM input-table1 INNER JOIN input-table2
      ON table1.col-name=table2.col-name;
QUIT;
```

Notes

- An SQL inner join combines matching rows between two tables.
- The two tables to be joined are listed in the FROM clause separated by INNER JOIN.
- The ON expression indicates how rows should be matched. The column names must be qualified as *table-name.col-name*.

Demo

1. Open **pg1.storm_summary** and **pg1.storm_basincodes** and compare the columns. Identify the matching column.
2. Open the **p107d02.sas** program in the **demos** folder and find the **Demo** section of the program. Add **pg1.storm_basincodes** to the FROM clause to perform an inner join on **Basin**. Qualify the **Basin** columns as *table-name.col-name* in the ON expression only.
3. Add the **BasinName** column to the query after **Basin**. Highlight the step, run the selected code, and examine the log. Why does the program fail?

```
proc sql;
  select Season, Name, Basin, BasinName, MaxWindMPH
    from pg1.storm_summary inner join pg1.storm_basincodes
      on storm_summary.basin=storm_basincodes.basin
    order by Season desc, Name;
quit;
```

4. Modify the query to qualify the **Basin** column in the SELECT clause. Highlight the step and run the selected code.

```
proc sql;
select Season, Name, storm_summary.Basin, BasinName, MaxWindMPH
  from pg1.storm_summary inner join pg1.storm_basincodes
    on storm_summary.basin=storm_basinodes.basin
  order by Season desc, Name;
quit;
```

Season	Name	Basin	BasinName	MaxWindMPH
2016		NI	North Indian	35
2016	AERE	WP	West Pacific	69
2016	AGATHA	EP	East Pacific	52
2016	AMOS	SP	South Pacific	92
2016	ANNABELLE	SI	South Indian	63
2016	BLAS	EP	East Pacific	138
2016	BOHALE	SI	South Indian	40
2016	CELIA	EP	East Pacific	69

End of Demonstration

Combining Tables with SQL

`FROM table1 AS alias1 INNERJOIN table2 AS alias2`

```
proc sql;
  select u.Name, Grade, Age, Teacher
    from pg1.class_update as u
      inner join pg1.class_teachers as t
        on u.Name=t.Name;
quit;
```

24

Copyright © SAS Institute Inc. All rights reserved.

Typing the full table names to qualify columns can be tedious. SQL enables you to assign an alias (or nickname) to a table in the FROM clause by adding the keyword AS and the alias of your choice. Then you can use the alias in place of the full table name to qualify columns in the other clauses of a query. In this example, the aliases for the two tables are the letters **U** and **T**.

7.03 Activity

Open **p107a03.sas** from the **activities** folder and perform the following tasks:

1. Define aliases for **storm_summary** and **storm_basinCodes** in the FROM clause.
2. Use one table alias to qualify **Basin** in the SELECT clause.
3. Complete the ON expression to match rows when **Basin** is equal in the two tables. Use the table aliases to qualify **Basin** in the expression. Run the step.

`FROM table1 AS alias1 INNERJOIN table2 AS alias2
ON alias1.column = alias2.column`

25

Copyright © SAS Institute Inc. All rights reserved.

Comparing SQL and the DATA Step

DATA step



PROC SQL



- provides more control of reading, writing, and manipulating data
- can create multiple tables in one step
- includes looping and array processing

- ANSI-standard language used by most databases
- code can be more streamlined
- can manipulate, summarize, and sort data in one step

The DATA step and PROC SQL each have unique strengths.



sas

28

Copyright © SAS Institute Inc. All rights reserved.

The DATA step and SQL each provide rich syntax designed to solve our data processing requirements. But each has its own strengths, and therefore it is helpful to know both as well as the situations in which one might be easier or more efficient than the other.

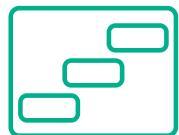
The DATA step provides very detailed and customizable control over how data is read, processed, and written. It includes the ability to create multiple tables simultaneously in a single DATA step, which requires reading the input table only once. It also includes syntax for creating loops and processing data in arrays.

SQL has the distinct advantage of being a standardized language that is used in most databases. Some SQL syntax can be more streamlined than the equivalent statements in a DATA or PROC step. And as we have seen, SQL can sometimes do in one query what can require multiple steps in SAS, such as creating a report in sorted order.

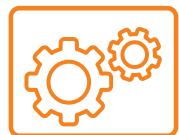
Ultimately, it is a great benefit to know both native SAS syntax and SQL and use them when appropriate in your SAS programs.

Comparing SQL and the DATA Step

DATA step



PROC SQL



SAS Programming 2:
Data Manipulation
Techniques course

SAS SQL 1: Essentials
course

Learn about how
each step
processes data
in these courses.



sas

To learn more about the DATA step, take the SAS Programming 2: Data Manipulation Techniques course. To learn more about SQL, take the SAS SQL 1: Essentials course. In both courses, we teach how the DATA step or PROC SQL runs behind the scenes so that you can control the processing of your data with appropriate syntax. This enables you to take advantage of the best features in each approach.

Beyond SAS Programming 1

What if you want to ...

... gain a deeper understanding of the SQL language and its implementation in SAS?

- Take the [SAS SQL 1 course](#).
- Read [PROC SQL by Example](#).

... get a power tour of techniques to improve SQL efficiency in SAS?

- Take the [SAS SQL Methods and More course](#).
- Read [Practical and Efficient SAS Programming](#).

... combine SQL and DATA step processing in a single programming language?

- Take the [DS2 Programming Essentials course](#).
- Read [Mastering the SAS DS2 Procedure](#).

30
Copyright © SAS Institute Inc. All rights reserved.



Links

- Take the [SAS SQL 1 course](#).
- Read [PROC SQL by Example](#).
- Take the [SAS SQL Methods and More course](#).
- Read [Practical and Efficient SAS Programming](#).
- Take the [DS2 Programming Essentials course](#).
- Read [Mastering the SAS DS2 Procedure](#).

Put It All Together!

Extended Learning - SAS® Programming 1: Essentials

Dashboard / Courses / Extended Learning - SAS® Programming 1: Essentials

Practice all your new SAS skills by completing a comprehensive case study. Visit the Extended Learning page to access the case study materials.



31

Copyright © SAS Institute Inc. All rights reserved.



Join the Discussion

Discuss SAS Courses and Test Your SAS Skills

<https://communities.sas.com/sas-training>

Visit the SAS Training Community to ask questions of our experts and exchange ideas with other SAS users.



SAS Training

Title

Programming 1 and 2

For Questions Related to SAS Programming Courses
Latest Topic - [This is a board post headline](#)
Latest Post - [This is a board post headline](#)

Course Case Studies and Challenges

Practice Your SAS Programming Knowledge

32

Copyright © SAS Institute Inc. All rights reserved.



7.3 Solutions

Solutions to Activities and Questions

7.01 Activity – Correct Answer

1. What are the similarities and differences in the syntax of the two steps?
 - Both the PRINT procedure and the SQL procedure provide the input table name, column list, and format.
 - PROC PRINT uses multiple statements to specify input data and report contents. PROC SQL uses one statement.
 - PROC PRINT separates columns with spaces. PROC SQL separates columns with commas.
 - PROC PRINT ends with a RUN statement. PROC SQL ends with a QUIT statement.
 - PROC SQL allows computed columns in the SELECT clause.
2. What are the similarities and differences in the results?
 - All rows and selected columns are included in both reports.
 - PROC PRINT adds the OBS column by default.

10

Copyright © SAS Institute Inc. All rights reserved.



7.02 Activity – Correct Answer

What storm had the highest cost? Hurricane Katrina

```
title "Most Costly Storms";
proc sql;
select Event, Cost format=dollar16., year(Date) as Season
  from pg1.storm_damage
  where Cost > 25000000000
  order by Cost desc;
quit;
```

Most Costly Storms		
Event	Cost	Season
Hurricane Katrina	\$161,300,000,000	2005
Hurricane Harvey	\$125,000,000,000	2017
Hurricane Maria	\$90,000,000,000	2017
Hurricane Sandy	\$70,900,000,000	2012
Hurricane Irma	\$50,000,000,000	2017
Hurricane Andrew	\$48,300,000,000	1992
Hurricane Ike	\$35,100,000,000	2008
Hurricane Ivan	\$27,300,000,000	2004

15

Copyright © SAS Institute Inc. All rights reserved.



continued...

7.03 Activity – Correct Answer

```
proc sql;
select Season, Name, s.Basin, BasinName, MaxWindMPH
  from pg1.storm_summary as s
    inner join pg1.storm_basincodes as b
      on s.basin=b.basin
  order by Season desc, Name;
quit;
```

Season	Name	Basin	BasinName	MaxWindMPH
2016		NI	North Indian	35
2016	AERE	WP	West Pacific	69
2016	AGATHA	EP	East Pacific	52
2016	AMOS	SP	South Pacific	92
2016	ANNABELLE	SI	South Indian	63
2016	BLAS	EP	East Pacific	138

The `storm_summary` table includes some lowercase `Basin` values. Are they in the results?



26

Copyright © SAS Institute Inc. All rights reserved.

7.03 Activity – Correct Answer

```
proc sql;
select Season, Name, s.Basin, BasinName, MaxWindMPH
  from pg1.storm_summary as s
    inner join pg1.storm_basincodes as b
      on upcase(s.basin)=b.basin
  order by Season desc, Name;
quit;
```

Season	Name	Basin	BasinName	MaxWindMPH
2016		NI	North Indian	35
2016	AERE	WP	West Pacific	69
2016	AGATHA	EP	East Pacific	52
2016	ALEX	na	North Atlantic	86
2016	AMOS	SP	South Pacific	92
2016	ANNABELLE	SI	South Indian	63
2016	BLAS	EP	East Pacific	138

Use the `UPCASE` function in the `ON` expression!



27

Copyright © SAS Institute Inc. All rights reserved.