# Base SAS® Enhancements in SAS®9

Course Notes

*Base SAS® Enhancements in SAS®9 Course Notes* was developed by Kathy Kiraly. Additional contributions were made by Theresa Lautato and Jason Secosky. Editing and production support was provided by the Curriculum Development and Support Department.



**Base SAS® Enhancements in SAS®9 Course Notes**



Book code 59208, course code LWBASE, prepared date 01Sep04.

# Table of Contents

# Course Description

This Live Web class focuses on the Cross-Environment Data Access (CEDA) and the new features of Base SAS procedures, including the SORT, CONTENTS, IMPORT, and EXPORT procedures. You also learn SAS language components (for example, Perl regular expressions for character searches and pattern matching), new system options, and longer format and informat names.

During this Live Web class, students talk directly with a SAS software expert, learn and apply new software skills, and receive feedback on their learning in real time at their desktops. Students are sent a copy of the course data prior to class. Using this predefined data, students practice in their own SAS software session with guidance from the instructor. Lectures and demonstrations are combined with hands-on exercises and question-and-answer sessions for a highly interactive learning experience.

## To learn more…

SAS Education

A full curriculum of general and statistical instructor-based training is available at any of the Institute's training facilities. Institute instructors can also provide on-site training.

For information on 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 support.sas.com/training as well as in the Training Course Catalog.

SAS Publishing

For a list of other SAS books that relate to the topics covered in these Course Notes, USA customers can contact our SAS Publishing Department at 1-800-727-3228 or send e-mail to sasbook@sas.com. Customers outside the USA, please contact your local SAS office.

Also, see the Publications Catalog on the Web at www.sas.com/pubs for a complete list of books and a convenient order form.

# Prerequisites

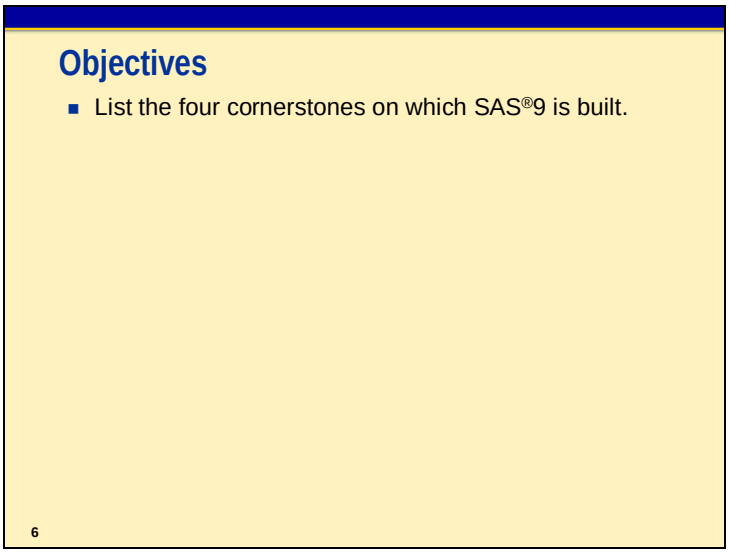
Before selecting this class, you should be able to

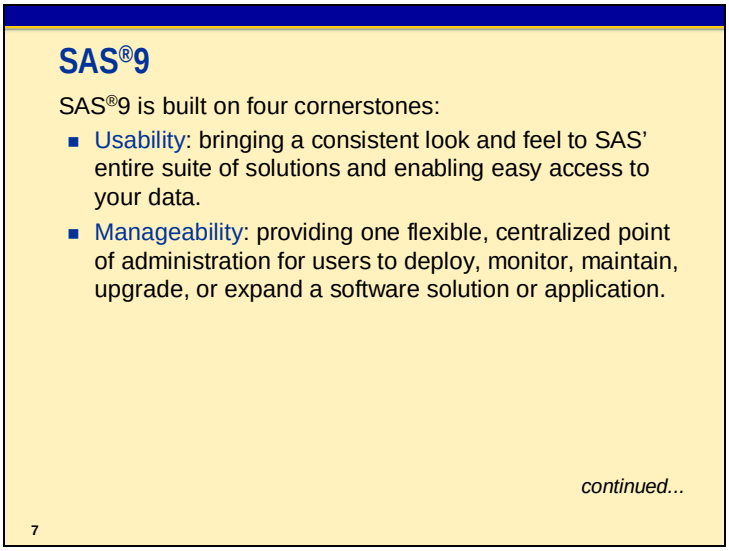
- submit SAS programs on your operating system
- understand the structure of a SAS data set
- create SAS output
- use the basic features of Base SAS procedures such as PROC PRINT and PROC CONTENTS.

You can gain this experience from the *SAS Programming I: Essentials* course.

# Module 1 Introduction to SAS®9

# 1.1 Introduction to SAS®9



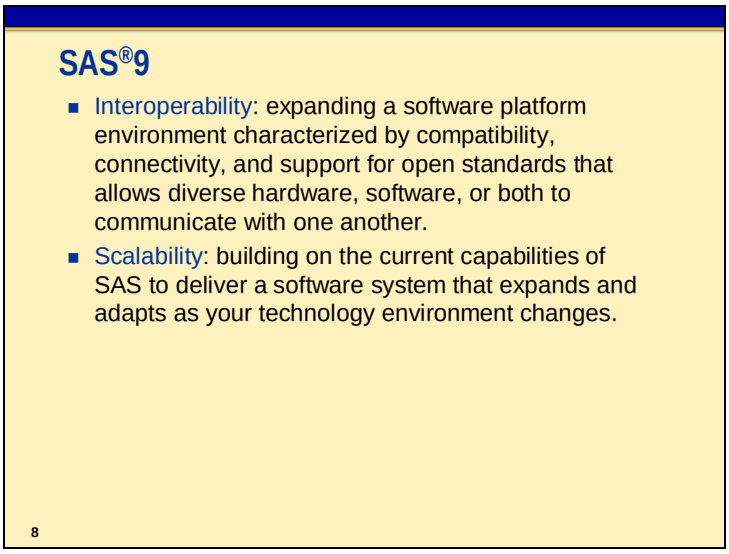

**Usability**

- The SAS Business Intelligence Server delivers an open and integrated enterprise architecture, with targeted interfaces that support your specific needs and skills. It enables you to access, analyze, and share your results by using a common metadata infrastructure and centralized administration.
- The MIGRATE procedure simplifies the process of migrating your libraries.
- New SAS/GRAPH styles provide a consistent look for output created by the Output Delivery System (ODS), which enhances readability and usability.
- The metadata LIBNAME engine enables you to read and create metadata in a SAS Metadata Repository. By incorporating metadata, this engine makes it easier to control access to the data.
- The Output Delivery System writes to more destinations and provides a greater variety of formatting selections, which enables you to select an output destination and format that best meets your reporting needs.

- User-created formats and informats can have names that are longer than eight characters, enabling you to provide names that are more descriptive.
- New SAS functions improve how you can search for character strings and regular expressions, which makes it easier for you to search your data for specific results.
- SAS Data Quality Server has been re-engineered to add increased accuracy and usability to your data.
- All help and reference documentation is accessible within a SAS session. Selecting **SAS Help and Documentation** from the Help menu enables you to view both online Help information and the full SAS reference library. Before SAS®9, the reference library was available only on the SAS OnlineDoc CD-ROM or in hard-copy format.
- SAS has been enhanced with accessibility features for SAS users who have disabilities.
- National Language Support (NLS) enhancements enable customers in regions around the world to process data successfully in their native languages and environments.

**Manageability**

- SAS Management Console provides a single point-of-control for SAS administrative tasks.
- SAS ETL Studio is a thin-client system, developed by using Java technology, which enables you to manage the Extraction, Transformation, and Loading (ETL) of data.
- Application Response Measurement (ARM) enables you to check the availability and the transaction rates of SAS applications.
- Secure Sockets Layer (SSL) provides network security and privacy. SSL is used in Base SAS, SAS/CONNECT, and SAS/SHARE software and in Integration Technologies.
- The new Integration Technologies Windows Object Manager and Java Connection Service create and manage workspace objects that support new types of IOM servers.
- New IOM options are provided to support load balancing.




**Interoperability**

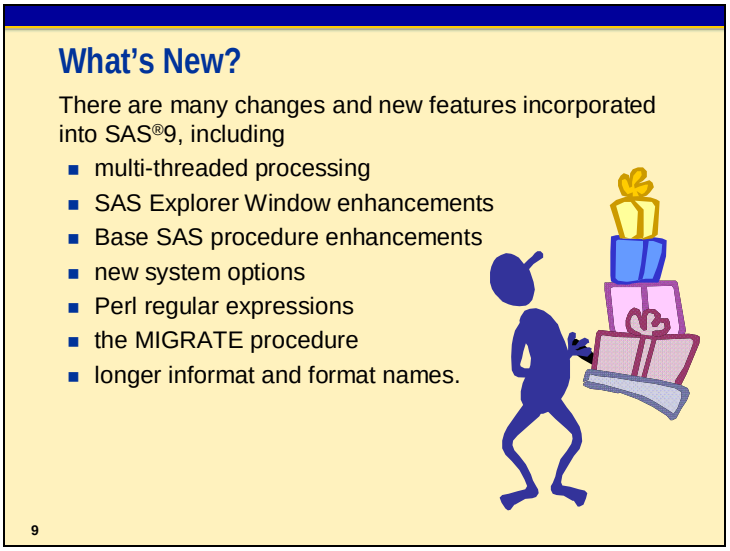
- SAS Open Metadata Architecture provides common metadata services to all SAS applications, which improves communication among applications.
- The SAS Add-In for Microsoft Office is an integral part of a complete, end-to-end Business Intelligence solution that enables you to access relational data sources, access and perform analytics, and create reports within Microsoft Office.
- SAS/CONNECT libref inheritance eliminates the need to duplicate data for use in multiple SAS sessions. Server sessions can inherit client-defined librefs, which allows multiple sessions to read and write data in a single library.
- SAS Integration Technologies now includes support for creating Web services that enable cross-platform integration, an enhanced publishing framework that supports the generation and publication of explicit and implicit events, and a new set of core infrastructure services that Java programmers can use to write applications that are integrated with the SAS platform.
- The XML LIBNAME engine imports and exports a broader variety of XML documents.

**Scalability**

- SAS now runs in many 64-bit operating environments, which allows SAS to scale in-memory processes.
- Parallel processing takes advantage of multiple CPUs by dividing processing among the available CPUs, which provides performance gains for two types of SAS processes: threaded I/O and threaded application processing. Following are some areas that use parallel processing:
  - When creating an index that requires sorting, SAS attempts to sort the data using the thread-enabled sort. By dividing the sorting task into separately executable processes, the time needed to sort the data can be reduced.
  - The following selected analytic procedures are thread enabled:
    - Base SAS procedures: MEANS, REPORT, SORT, SQL, SUMMARY, TABULATE
    - SAS/STAT procedures: GLM, LOESS, REG, ROBUSTREG
    - SAS/SHARE procedure: SERVER (with the experimental THREADEDTCP option)
    - SAS Enterprise Miner procedures: DMINE, DMREG.

- The following SAS/ACCESS engines use multiple threads to access data from the DBMS server:
  - Oracle
  - Sybase
  - DB2 (UNIX and PC)
  - ODBC
  - SQL Server
  - Teradata.
- The Scalable Performance Data (SPD) Engine provides parallel I/O by using multiple CPUs to read SAS data and deliver it rapidly to applications.
- MP CONNECT gives you the ability to exploit SMP (Symmetric Multi-Processing) hardware as well as network resources to perform parallel processing and easily coordinate all the results into a single client SAS session.
- The SAS Metadata Server uses threads to enable the best response time for delivering metadata as requested by any number of clients.
- The SAS OLAP Server provides a new multi-threaded data storage and server functionality that provides faster cube performance. The data can be stored in a multidimensional form (MOLAP) or in a form that includes existing aggregations from presummarized data sources.

- SAS/CONNECT supports pipeline parallelism, which allows multiple DATA steps or procedures to execute in parallel and to pipe the output from one process as the input to the next process in a pipeline. Piping improves performance and reduces the demand for disk space.
- The time required for transferring large amounts of data when using SAS/CONNECT has been significantly reduced as a result of improvements to the file compression algorithm.



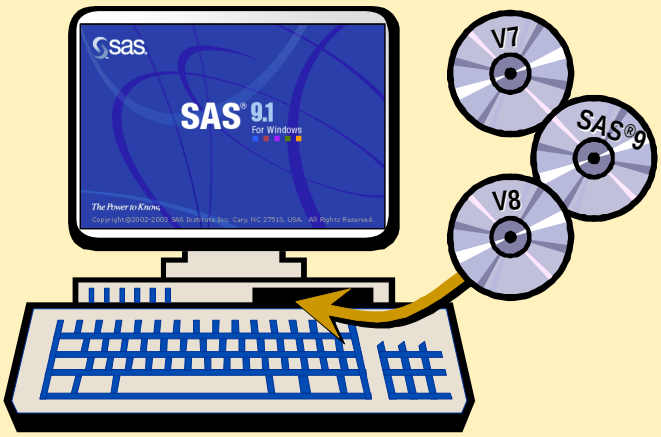
# 1.2 SAS Version 8 to SAS®9: A Peaceful Coexistence

## Objectives

- List the type of access SAS®9 has to Version 7 and Version 8 files.

12

## SAS Programs




13

## SAS®9 Access to Version 7 and 8 Files

| Version 7 and 8 File Type | Supported Processing, Native File | Supported Processing, Foreign File |
|---|---|---|
| Data Files | Read/Write/Update | Read/Write |
| DATA Step Views | Read/Write/Update | Not Supported |
| SQL Views | Read/Write/Update | Read Only |
| SAS/ACCESS Views (Oracle and Sybase) | Read/Write/Update | Read Only |
| SAS/ACCESS Views (other than Oracle and Sybase) | Read/Write/Update | Not Supported |
| SAS Catalogs | Read/Write/Update | Not Supported |
| Stored Programs | Read/Write/Update | Not Supported |

14

Native files are created under an operating environment that is compatible with the current operating system, for example, Windows 2000 and Windows XP. To automatically process Version 7 and 8 native files, SAS uses the default base engine. All SAS®9 processes are supported, except the longer format and informat names.

Foreign, or non-native, files are not compatible with the current environment. This noncompatibility occurs when you migrate from a 32-bit platform to a 64-bit platform. This is also true if you change operating system families, for example, from z/OS to UNIX.

# 1.3 Overview of Enhancements

## Objectives

- List selected enhancements in SAS®9.

18

## SAS Explorer Window Enhancements

- Favorite Folders
- My Computer
- Enhancements to the Libraries window


19

## Selected Procedure Enhancements

Enhanced Procedures

- FORMAT
- SORT
- FREQ

New Procedure

- MIGRATE

20

## Selected SAS Language Enhancements

New Functions

- PROPCASE
- CAT, CATT, CATS, CATX
- FIND
- COUNT

Perl Regular Expressions

- PRXCHANGE
- PRXMATCH

System Options

- DTRESET
- VALIDFMTNAME

21

# 1.4 SAS Explorer Window Enhancements

## Objectives

- Open a file in My Computer.
- View metadata in the enhanced Properties window.
- View a SAS data set in Microsoft Excel.
- Add a folder to Favorite Folders.

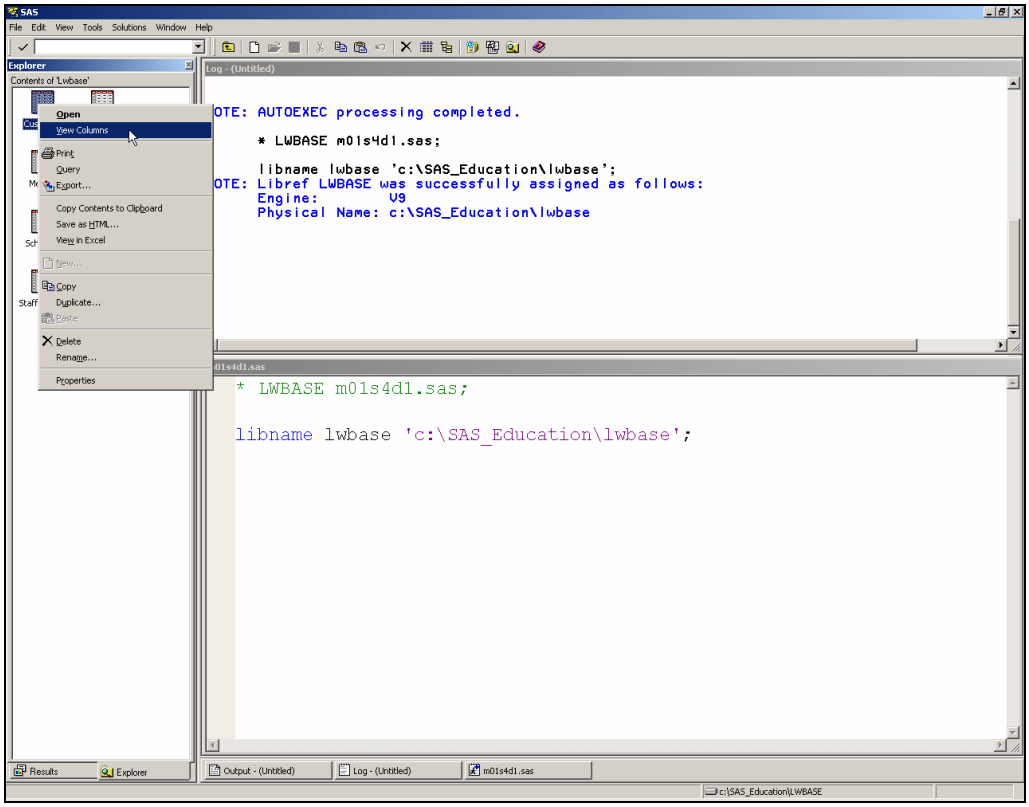
24

## SAS Explorer Window Enhancements

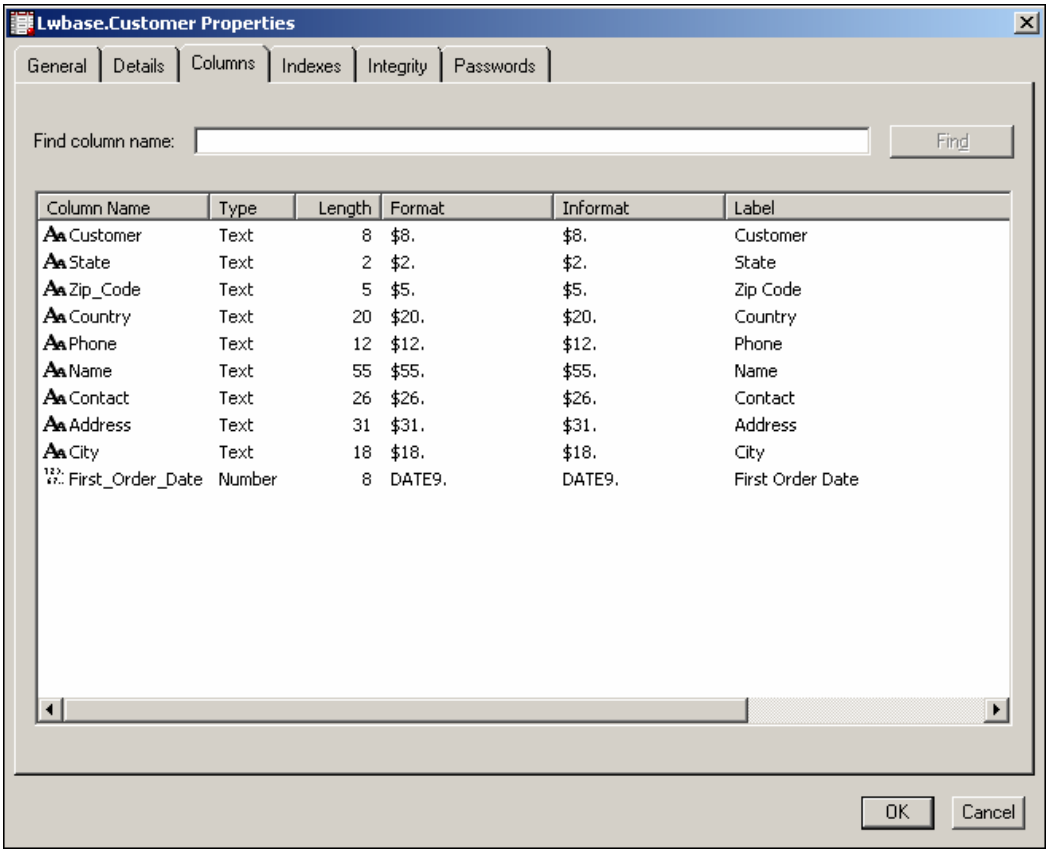
m01s4d1.sas

```
libname lwbase 'c:\SAS_Education\lwbase';
```

1. Submit the LIBNAME statement.
2. In the SAS Explorer window, select **Libraries** ⇨ **Lwbase**.
3. Right-click **Customer** in the Explorer window and then select **View Columns** from the menu.

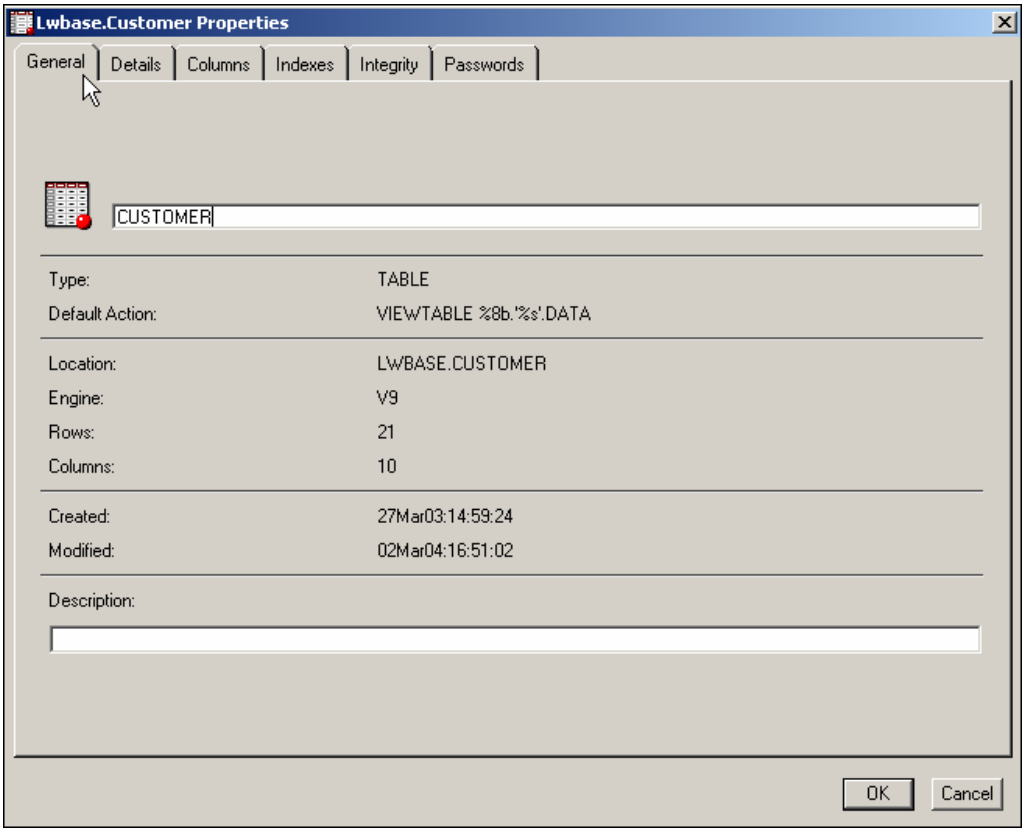


4. Review the metadata for **`Lwbase.Customer`**.

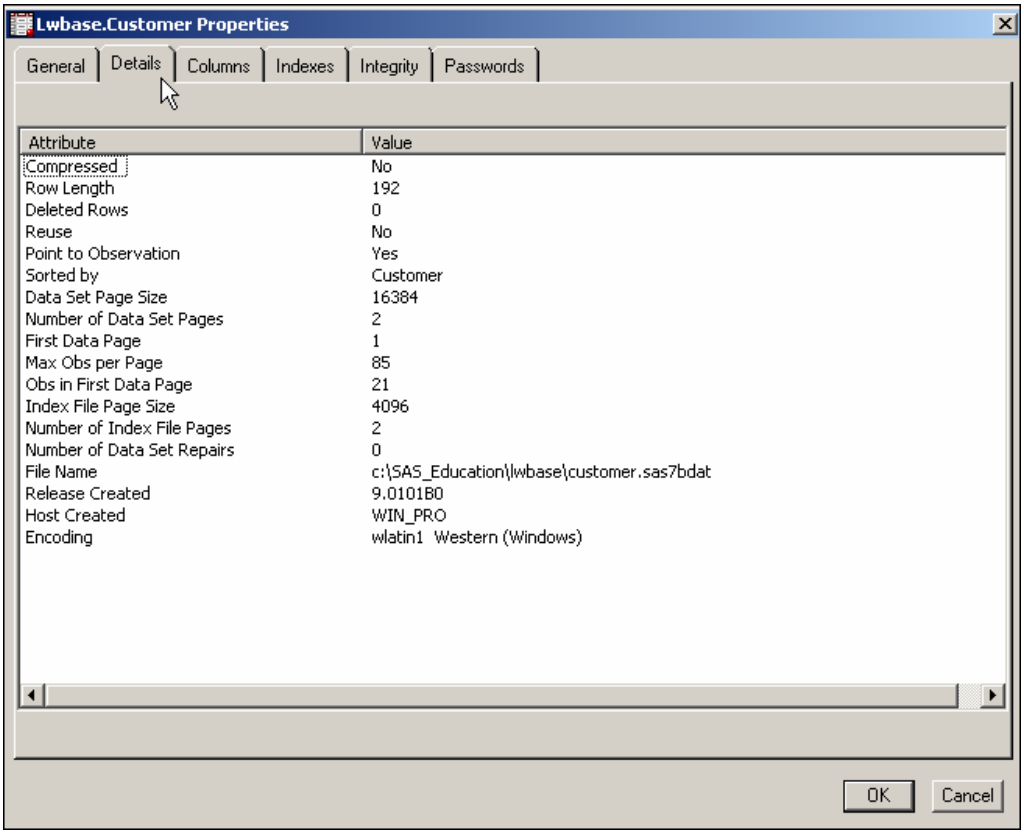



| Column Name | Type | Length | Format | Informat | Label |
|---|---|---|---|---|---|
| Customer | Text | 8 | $8. | $8. | Customer |
| State | Text | 2 | $2. | $2. | State |
| Zip_Code | Text | 5 | $5. | $5. | Zip Code |
| Country | Text | 20 | $20. | $20. | Country |
| Phone | Text | 12 | $12. | $12. | Phone |
| Name | Text | 55 | $55. | $55. | Name |
| Contact | Text | 26 | $26. | $26. | Contact |
| Address | Text | 31 | $31. | $31. | Address |
| City | Text | 18 | $18. | $18. | City |
| First_Order_Date | Number | 8 | DATE9. | DATE9. | First Order Date |

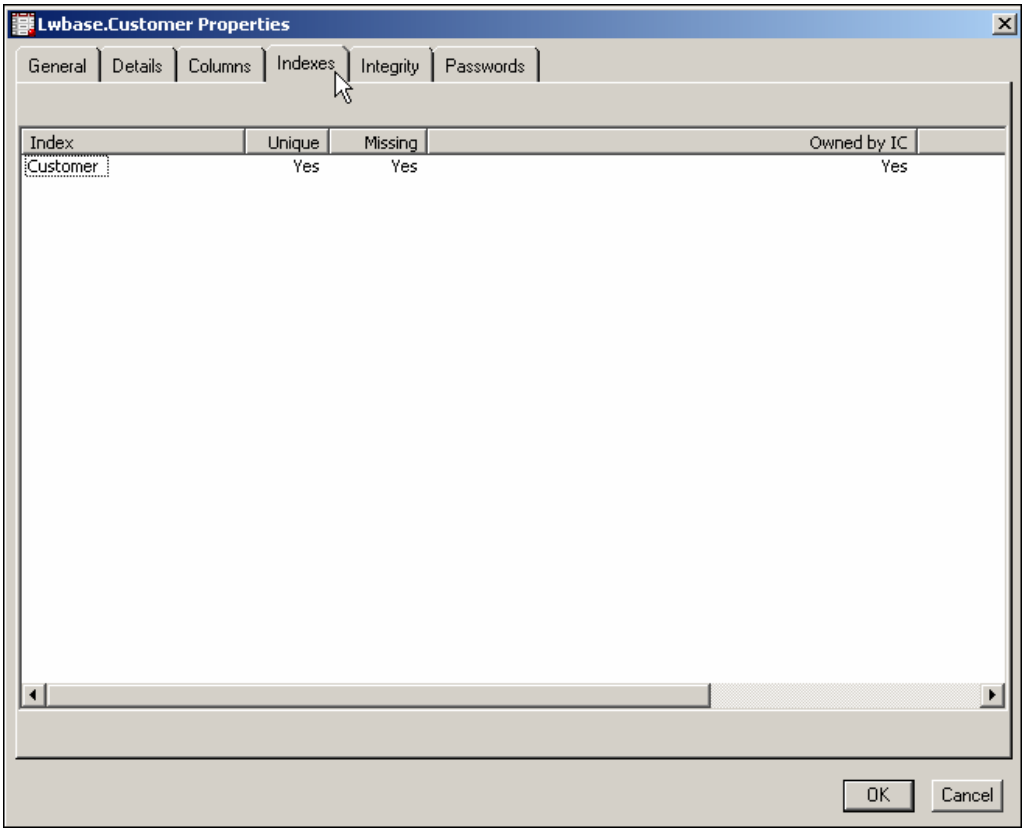
5. Select the General tab to see the number of observations and variables in the data set.



6. Select the Details tab to view Engine/Host information for the data set.

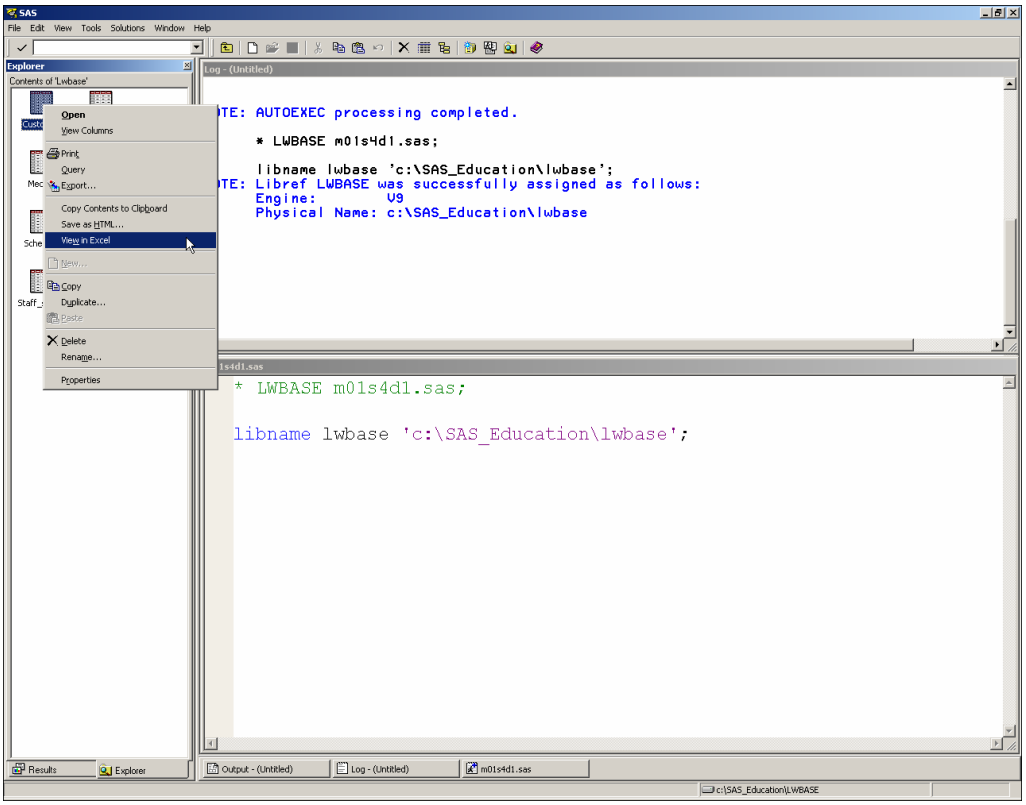


7. Select the Indexes tab to see index information.

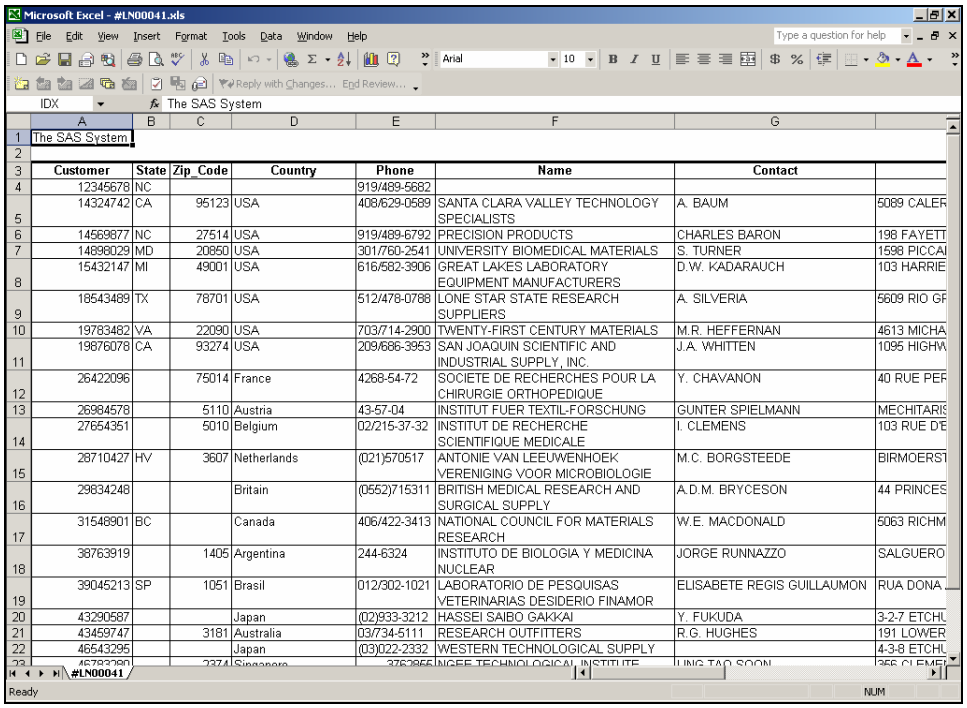



8. Select **OK** to close the Properties window.

9. Return to the Explorer window. Right-click **Customer** and then select **View in Excel** from the menu.



10. View your data in a Microsoft Excel worksheet.




| Customer | State | Zip_Code | Country | Phone | Name | Contact | |
|---|---|---|---|---|---|---|---|
| 12345678 | NC | | | 919/489-5682 | | | |
| 14324742 | CA | 95123 | USA | 408/629-0589 | SANTA CLARA VALLEY TECHNOLOGY SPECIALISTS | A. BAUM | 5089 CALER |
| 14569877 | NC | 27514 | USA | 919/489-6792 | PRECISION PRODUCTS | CHARLES BARON | 198 FAYETT |
| 14898029 | MD | 20850 | USA | 301/760-2541 | UNIVERSITY BIOMEDICAL MATERIALS | S. TURNER | 1598 PICCAI |
| 15432147 | MI | 49001 | USA | 616/582-3906 | GREAT LAKES LABORATORY EQUIPMENT MANUFACTURERS | D.W. KADARAUCH | 103 HARRIE |
| 18543489 | TX | 78701 | USA | 512/478-0788 | LONE STAR STATE RESEARCH SUPPLIERS | A. SILVERIA | 5609 RIO GF |
| 19783482 | VA | 22090 | USA | 703/714-2900 | TWENTY-FIRST CENTURY MATERIALS | M.R. HEFFERNAN | 4613 MICHA |
| 19876078 | CA | 93274 | USA | 209/686-3953 | SAN JOAQUIN SCIENTIFIC AND INDUSTRIAL SUPPLY, INC. | J.A. WHITTEN | 1095 HIGHV |
| 26422096 | | 75014 | France | 4268-54-72 | SOCIETE DE RECHERCHES POUR LA CHIRURGIE ORTHOPEDIQUE | Y. CHAVANON | 40 RUE PEF |
| 26984578 | | 5110 | Austria | 43-57-04 | INSTITUT FUER TEXTIL-FORSCHUNG | GUNTER SPIELMANN | MECHITARIS |
| 27654351 | | 5010 | Belgium | 02/215-37-32 | INSTITUT DE RECHERCHE SCIENTIFIQUE MEDICALE | I. CLEMENS | 103 RUE D'E |
| 28710427 | HV | 3607 | Netherlands | (021)570517 | ANTONIE VAN LEEUWENHOEK VERENIGING VOOR MICROBIOLOGIE | M.C. BORGSTEEDE | BIRMOERST |
| 29834248 | | | Britain | (0552)715311 | BRITISH MEDICAL RESEARCH AND SURGICAL SUPPLY | A.D.M. BRYCESON | 44 PRINCES |
| 31548901 | BC | | Canada | 406/422-3413 | NATIONAL COUNCIL FOR MATERIALS RESEARCH | W.E. MACDONALD | 5063 RICHM |
| 38763919 | | 1405 | Argentina | 244-6324 | INSTITUTO DE BIOLOGIA Y MEDICINA NUCLEAR | JORGE RUNNAZZO | SALGUERO |
| 39045213 | SP | 1051 | Brasil | 012/302-1021 | LABORATORIO DE PESQUISAS VETERINARIAS DESIDERIO FINAMOR | ELISABETE REGIS GUILLAUMON | RUA DONA |
| 43290587 | | | Japan | (02)933-3212 | HASSEI SAIBO GAKKAI | Y. FUKUDA | 3-2-7 ETCHU |
| 43459747 | | 3181 | Australia | 03/734-5111 | RESEARCH OUTFITTERS | R.G. HUGHES | 191 LOWER |
| 46543295 | | | Japan | (03)022-2332 | WESTERN TECHNOLOGICAL SUPPLY | | 4-3-8 ETCHU |



You can issue the null title statement (`title;`) in your SAS program before you view output in Microsoft Excel. This removes the SAS title from your output.

11. Save your data to an Excel worksheet by selecting **File** ⇨ **Save As...**. Enter **`saved.xls`** as the name of the worksheet.

12. Return to the Explorer window. Use the Up One Level icon to find and open My Computer. Find the **`saved.xls`** workbook and open it.

13. Return to the Explorer window. Use the My Computer icon to find the LWBASE folder. Right-click the LWBASE folder and select **Add to Favorite Folders**.

14. Use the Up One Level icon to go to the Favorite Folders icon. Open Favorite Folders to see the LWBASE folder.

To change the font of the screen text and dialog box text elements, specify the SYSGUIFONT option during invocation of SAS:

-SYSGUIFONT "*font-name*" <*font-size*>

To open the Explorer window, select **View** ⇨ **Contents Only**.

**Exercise: Refer to your course workbook for Exercise 1.**

## Module 1 Summary

- SAS®9 is built on four cornerstones: Usability, Manageability, Interoperability, and Scalability.
- The Explorer window features enhancements to the Properties window, as well as a new View in Excel feature.
- SAS®9 supports read/write/update access to native SAS 8 and SAS 7 files. For non-native files, you will need to migrate your files to the current release.

29

# Module 2 Base SAS® Procedure Enhancements

# 2.1 Base SAS Procedure Enhancements

## Objectives

- Define multi-threaded processing.
- List Base SAS procedures that support multi-threaded processing.
- Customize the table of contents text and output using the FREQ procedure.
- Create longer format and informat names.
- Control the data set creation date and write out duplicate observations to a new data set using the SORT procedure.

3

## Multi-Threaded Processing

A *thread* is

- a single path of execution
- a basic unit of program execution in a thread-enabled operating environment.

4

## Multi-Threaded Processing

*Multi-threaded processing* is a type of parallel processing introduced in SAS®9. *Parallel processing* means that multiple units of work are available to be scheduled for concurrent execution by the operating system.

This technology takes advantage of hardware that has multiple central processing units, or CPUs, called symmetric multiprocessing machines (SMPs).

5

## Multi-Threaded Processing

A *symmetric multiprocessing environment*

- has multiple CPUs that share the same memory and a thread-enabled operating system
- can spawn and process multiple threads simultaneously using multiple CPUs
- enables the application to coordinate threads from the same process to share data efficiently.

6

## Multi-Threaded Processing

*Threaded kernel technology* is the component in SAS®9 that enables multi-threaded processing. The threaded kernel divides SAS processes into multiple threads, which allows for threaded I/O, parallel data transformation, and analysis.

Processes suitable for threading are

- sorting
- grouping
- summarizing.

7

## Multi-Threaded Procedures in Base SAS

Multi-threading can be enabled or disabled for the following Base SAS procedures:

- MEANS/SUMMARY
- REPORT
- SORT (excludes TAGSORT option)
- SQL (GROUP BY and ORDER BY)
- TABULATE.

8

## Multi-Threaded Procedures in Base SAS

Multi-threaded processing can be controlled via the SAS system option THREADS | NOTHREADS.

```
OPTIONS THREADS | NOTHREADS;
```

The default is THREADS.

9

## Multi-Threaded Procedures in Base SAS

The THREADS | NOTHREADS option also can be specified in the PROC statement, which enables or disables multi-threaded processing of the input data set.

When the option is specified in the PROC statement, it overrides the THREADS | NOTHREADS SAS system option.

Examples:

```
PROC MEANS DATA=SAS-data-set THREADS | NOTHREADS;
```

```
PROC SQL THREADS | NOTHREADS;
```

10

In the SQL procedure, you also can specify THREADS | NOTHREADS in a RESET statement.

## Multi-Threaded Procedures in Base SAS

The number of CPUs to use for processing can be controlled with the CPUCOUNT system option.

```
OPTIONS CPUCOUNT=1-1024 | ACTUAL;
```

1-1024
: is the number of CPUs that SAS will assume are available for use by thread-enabled applications.

ACTUAL
: is the number of CPUs that SAS detects are available for a specific session.

The default is ACTUAL.

11

If you specify a number that is higher than the number of available CPUs, SAS uses the number of CPUs that actually are available. To find out the number of CPUs available to SAS, submit the following code:

```
proc options option=cpucount;
run;
```

## Enhancements to the FORMAT Procedure

With the FORMAT procedure, you now can create formats and informats with longer names.

Format names for character and numeric formats can have a maximum length of **32**, which includes the **$** for character names.

Partial SAS Log

```
proc format;
   value $genderformat "1"="Female"
                       "2"="Male";
NOTE: Format $GENDERFORMAT has been output.
```

15

## Enhancements to the FORMAT Procedure

Informat names for character and numeric informats can have a maximum length of **32**, which includes the @ to distinguish informat names from format names, and the **$** for character names.

Partial SAS Log

```
proc format;
   invalue $deptinformat "A"="Dept A"
                         "B"="Dept B";
NOTE: Informat $DEPTINFORMAT has been output.
```

16

## New Options for the FREQ Procedure

CONTENTS=

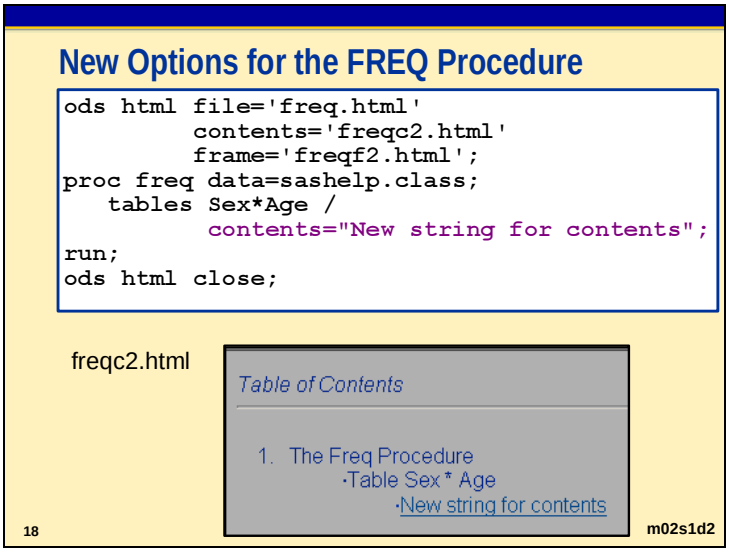
enables you to customize hyperlinks within the HTML contents file.

**TABLES** *variables* / CONTENTS=*string*;

The CONTENTS= option

- is used in the TABLES statement
- is valid only with crosstabulation tables
- affects only the HTML contents file and not the HTML body file.

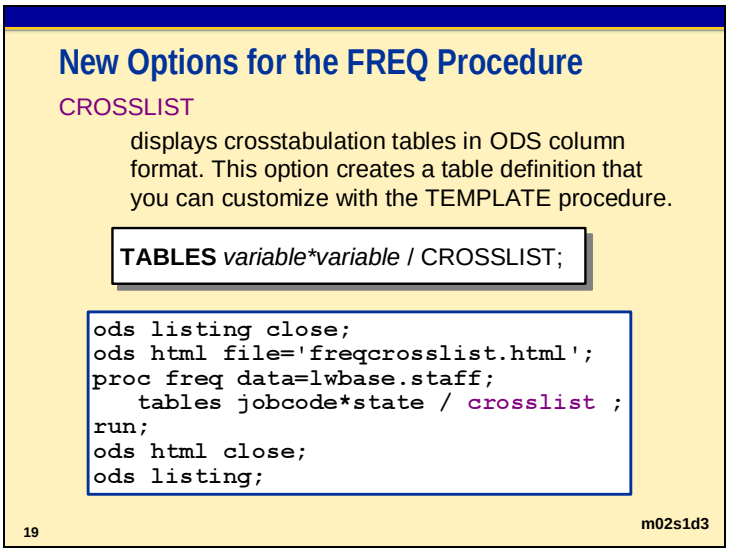
17




FILE= identifies the file that contains the HTML output. The alias is BODY=.

CONTENTS= identifies the file that contains a table of contents to the HTML output. The contents file links to the body file.

FRAME= identifies the file that integrates the table of contents, the page contents, and the body file.



## New Options for the FREQ Procedure

freqcrosslist.html (Partial Output)

Table of JobCode by State

| JobCode | State | Frequency | Percent | Row Percent | Column Percent |
|---|---|---|---|---|---|
| BCK | CT | 2 | 1.35 | 22.22 | 4.55 |
| | NJ | 0 | 0.00 | 0.00 | 0.00 |
| | NY | 7 | 4.73 | 77.78 | 7.87 |
| | Total | 9 | 6.08 | 100.00 | |
| FA1 | CT | 3 | 2.03 | 27.27 | 6.82 |
| | NJ | 1 | 0.68 | 9.09 | 6.67 |
| | NY | 7 | 4.73 | 63.64 | 7.87 |
| | Total | 11 | 7.43 | 100.00 | |
| FA2 | CT | 4 | 2.70 | 25.00 | 9.09 |
| | NJ | 3 | 2.03 | 18.75 | 20.00 |
| | NY | 9 | 6.08 | 56.25 | 10.11 |
| | Total | 16 | 10.81 | 100.00 | |

20

## Creating a Crosstabulation with the CROSSLIST Option

m02s1d3.sas

```
title 'Jobcode by State Default';
ods listing close;
ods html file='freqcrosslist.html';
proc freq data=lwbase.staff;
   tables jobcode*state;
run;
ods html close;
ods listing;
```

Partial Output

*Jobcode by State Default*

*The FREQ Procedure*

| Frequency<br>Percent<br>Row Pct<br>Col Pct | Table of JobCode by State | | | |
|---|---|---|---|---|
| | State | | | |
| JobCode | CT | NJ | NY | Total |
| BCK | 2<br>1.35<br>22.22<br>4.55 | 0<br>0.00<br>0.00<br>0.00 | 7<br>4.73<br>77.78<br>7.87 | 9<br>6.08 |
| FA1 | 3<br>2.03<br>27.27<br>6.82 | 1<br>0.68<br>9.09<br>6.67 | 7<br>4.73<br>63.64<br>7.87 | 11<br>7.43 |
| FA2 | 4<br>2.70<br>25.00<br>9.09 | 3<br>2.03<br>18.75<br>20.00 | 9<br>6.08<br>56.25<br>10.11 | 16<br>10.81 |

m02s1d3.sas

```
title 'Jobcode by State with CROSSLIST';
ods listing close;
ods html file='freqcrosslist.html';
proc freq data=lwbase.staff;
   tables jobcode*state / crosslist;
run;
ods html close;
ods listing;
```

Partial Output

*Jobcode by State with CROSSLIST*

*The FREQ Procedure*

**Table of JobCode by State**

| JobCode | State | Frequency | Percent | Row Percent | Column Percent |
|---|---|---|---|---|---|
| BCK | CT | 2 | 1.35 | 22.22 | 4.55 |
| | NJ | 0 | 0.00 | 0.00 | 0.00 |
| | NY | 7 | 4.73 | 77.78 | 7.87 |
| | Total | 9 | 6.08 | 100.00 | |
| FA1 | CT | 3 | 2.03 | 27.27 | 6.82 |
| | NJ | 1 | 0.68 | 9.09 | 6.67 |
| | NY | 7 | 4.73 | 63.64 | 7.87 |
| | Total | 11 | 7.43 | 100.00 | |

**Exercise: Refer to your course workbook for Exercise 2.**

## Enhancements to the SORT Procedure

Anytime that a SAS process reads and writes to the same data set, SAS resets the Created and Last Modified attributes. This can be a problem when you want to maintain the original creation date.

The DATASETS Procedure

| | | | |
|---|---|---|---|
| Data Set Name | SASUSER.COURSES | Observations | 6 |
| Member Type | DATA | Variables | 4 |
| Engine | V9 | Indexes | 0 |
| Created | Friday, February 14, 2003 03:45:03 | Observation Length | 48 |
| Last Modified | Friday, February 14, 2003 03:45:03 | Deleted Observations | 0 |
| Protection | | Compressed | NO |
| Data Set Type | | Sorted | YES |
| Label | | | |
| Data Representation | WINDOWS_32 | | |
| Encoding | wlatin1 Western (Windows) | | |

Sort Information

| | |
|---|---|
| Sortedby | Course_Code |
| Validated | YES |
| Character Set | ANSI |

24

## Enhancements to the SORT Procedure

```
proc sort data=sasuser.courses;
   by course_title;
run;
```

Data set after sorting:

The DATASETS Procedure

| | | | |
|---|---|---|---|
| Data Set Name | SASUSER.COURSES | Observations | 6 |
| Member Type | DATA | Variables | 4 |
| Engine | V9 | Indexes | 0 |
| Created | Thursday, February 20, 2003 05:41:17 | Observation Length | 48 |
| Last Modified | Thursday, February 20, 2003 05:41:17 | Deleted Observations | 0 |
| Protection | | Compressed | NO |
| Data Set Type | | Sorted | YES |
| Label | | | |
| Data Representation | WINDOWS_32 | | |
| Encoding | wlatin1 Western (Windows) | | |

Sort Information

| | |
|---|---|
| Sortedby | Course_Title |
| Validated | YES |
| Character Set | ANSI |

25

## Maintaining the Data Set Creation Date

The DATECOPY option in the SORT procedure sorts a SAS data set **without** changing the Created and Last Modified attributes.

The DATASETS Procedure

| | | | |
|---|---|---|---|
| Data Set Name | SASUSER.COMPANY | Observations | 8 |
| Member Type | DATA | Variables | 4 |
| Engine | V9 | Indexes | 0 |
| Created | Friday, February 14, 2003 03:44:56 | Observation Length | 48 |
| Last Modified | Friday, February 14, 2003 03:44:56 | Deleted Observations | 0 |
| Protection | | Compressed | NO |
| Data Set Type | | Sorted | YES |
| Label | | | |
| Data Representation | WINDOWS_32 | | |
| Encoding | wlatin1 Western (Windows) | | |

Sort Information

| | |
|---|---|
| Sortedby | Age |
| Validated | YES |
| Character Set | ANSI |

26

## Maintaining the Data Set Creation Date

```
proc sort data=sasuser.company datecopy;
   by Name;
run;
```

Data set after sorting with the DATECOPY option:

The DATASETS Procedure

| | | | |
|---|---|---|---|
| Data Set Name | SASUSER.COMPANY | Observations | 8 |
| Member Type | DATA | Variables | 4 |
| Engine | V9 | Indexes | 0 |
| Created | Friday, February 14, 2003 03:44:56 | Observation Length | 48 |
| Last Modified | Friday, February 14, 2003 03:44:56 | Deleted Observations | 0 |
| Protection | | Compressed | NO |
| Data Set Type | | Sorted | YES |
| Label | | | |
| Data Representation | WINDOWS_32 | | |
| Encoding | wlatin1 Western (Windows) | | |

Sort Information

| | |
|---|---|
| Sortedby | Name |
| Validated | YES |
| Character Set | ANSI |

27

The DATECOPY option cannot be used in conjunction with the OUT= option.

## Enhancements to the SORT Procedure

The DUPOUT= option specifies the output data set to which duplicate observations are written.

```
libname lwbase 'c:\SAS_Education\lwbase';
proc sort data=lwbase.schedule nodupkey
          out=work.sortsched
          dupout=work.dupobs;
   by Course_Code;
run;
```

29 m02s1d4

The DUPOUT= option must be used in conjunction with the NODUPKEY or the NODUPRECS option.

## The DUPOUT= Option

Partial SAS Log

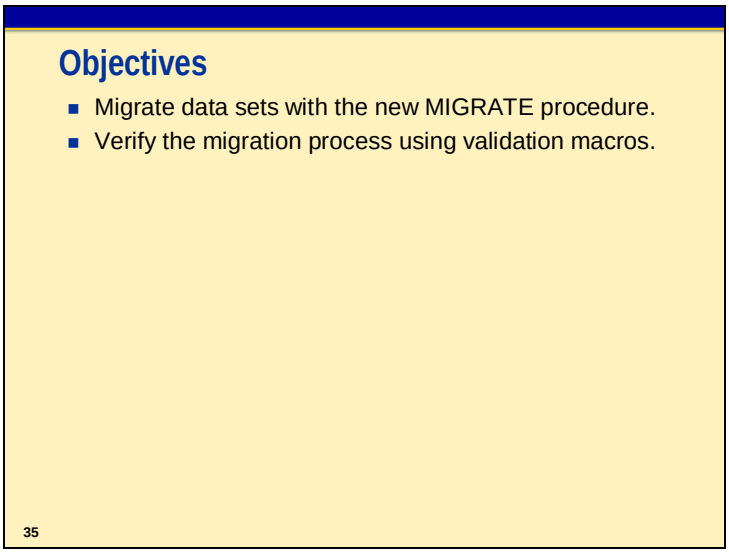
```
16   proc sort data=lwbase.schedule nodupkey
17             out=work.sortsched
18             dupout=work.dupobs;
19      by Course_Code;
20   run;

NOTE: There were 18 observations read from the data set LWBASE.SCHEDULE.
NOTE: 12 observations with duplicate key values were deleted.
NOTE: The data set WORK.SORTSCHED has 6 observations and 5 variables.
NOTE: The data set WORK.DUPOBS has 12 observations and 5 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

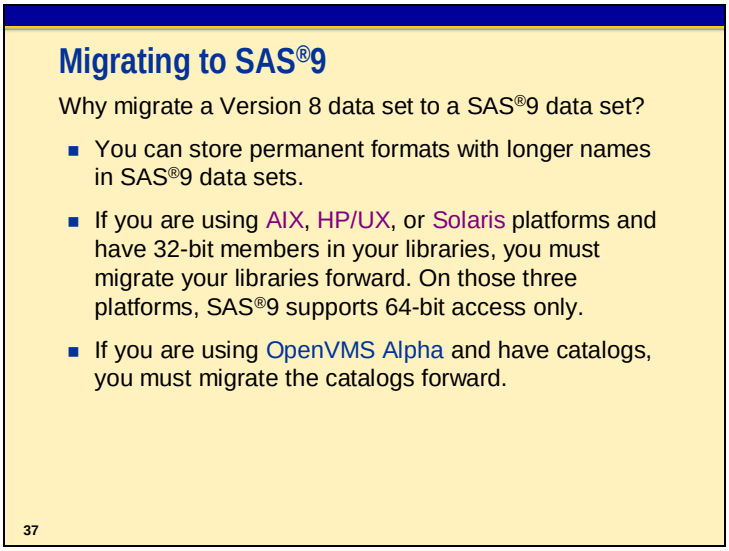
30

For more information on Base SAS procedures, go to http://support.sas.com/rnd/base/index.html.

**Exercise: Refer to your course workbook for Exercise 3.**

# 2.2 The MIGRATE Procedure




You can apply a longer format name to variables in a Version 8 data set within a SAS®9 session; however, you cannot permanently store that format without re-creating the data set as a SAS®9 data set.

## Overview of the MIGRATE Procedure

The MIGRATE procedure

- is the Base SAS tool for transitioning SAS libraries from older releases to the current release
- enables new functionality not found in the COPY, CATALOG, and CPORT/CIMPORT procedures
- does **not** allow you to specify individual files to be migrated.

38

## Overview of the MIGRATE Procedure

The procedure will migrate

- a library to SAS 9.1 beginning with the latest release of Version 6 for your platform
- a 32-bit library to a 64-bit library on the same operating system family
- data files including indexes, integrity constraints, audit trails, compression, encryption, and passwords
- views (DATA step, SQL, and ACCESS)
- catalogs
- MDDBs (other than Version 7).

*continued...*

39

## Overview of the MIGRATE Procedure

The procedure will migrate

- generation data sets
- item stores
- the SAS internal date and time of when the member was created and last modified.

40

## Migrating a SAS Data Library

General form of a PROC MIGRATE step:

```
PROC MIGRATE IN=libref-1 OUT=libref-2 <options>;
RUN;
```

IN= identifies the source library, or the library you want to migrate.

OUT= identifies the target library, or the library to which you want to migrate.

41

Data set options are not allowed in the IN= or the OUT= value.

## Additional MIGRATE Procedure Options

BUFSIZE=***n* | *n*K | *n*M | *n*G | *hex*X | MAX**

determines the buffer page size.

*n* | *n*K | *n*M | *n*G

specify the buffer page size of the members that are being written to the target library. *n* specifies the page size in multiples of 1 (bytes), 1,024 (**k**ilobytes), 1,048,576 (**m**egabytes), or 1,073,741,824 (**g**igabytes).

For example, a value of **8** specifies a page size of 8 bytes, and a value of **4k** specifies a page size of 4096 bytes.

*continued...*

42

## MIGRATE Procedure Options – BUFSIZE=

*hex*X

specifies the page size as a hexadecimal value. You must specify the value beginning with a number (0-9), followed by an X. For example, the value **2dx** sets the page size to 45 bytes.

MAX

sets the page size to the maximum possible number in your operating environment, up to the largest 4-byte, signed integer, which is 232-1, or approximately 2 billion bytes.

The default is the original page size used to create the source library member.

43

## Additional MIGRATE Procedure Options

MOVE

moves SAS files from the source library to the target library and deletes the original files from the source library. Specifying MOVE reduces the scope of the validation tools.

44

## Additional MIGRATE Procedure Options

SLIBREF=*libref*

specifies a libref that is assigned through a SAS/SHARE or SAS/CONNECT server to use for migrating catalogs with the following conditions:

- When the IN=source library is 32 bit and the OUT= target library is 64 bit
- When the IN=source library contains catalogs created before SAS®9 on OpenVMS Alpha or OpenVMS VAX.

45

## Validation Tools

Validation tools

- provide an easy method to document the migration of your libraries
- are used in the pre-migration process to determine the expected behavior of the conversion of a particular library
- are used in the post-migration process to determine whether the conversion produced the expected results.

47

## Validation Tools – %BEFORE

%BEFORE

- creates the macro variables needed for the memtype comparisons
- creates a data set in the ODS library for the contents of the source library before migration
- creates a data set in the ODS library describing the indexes and integrity constraints before migration
- uses the source library data set to create macro variables that contain the total number of files for each memtype, as well as a macro variable that contains the name of each file in the source library.

48

## Validation Tools – %AFTER

%AFTER compares

- the contents of the source library before PROC MIGRATE with the contents of the target library after PROC MIGRATE by comparing the contents of the data sets created in the ODS library and generates a report
- the contents of the source library before PROC MIGRATE with the contents of the source library after PROC MIGRATE and generates a report.

49

## Validation Tools – %CHECKEM

%CHECKEM

uses the information gathered by the %BEFORE macro to produce validation output of all members with the following memtypes:

- catalogs (catalog attributes only)
- data sets, including indexes, audit trails, integrity constraints, and generations
- SQL views.

*continued...*

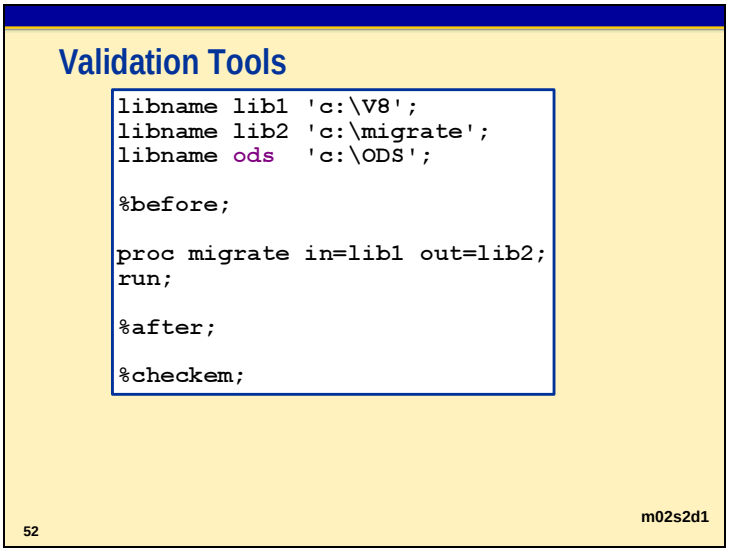
50

## Validation Tools – %CHECKEM

%CHECKEM outputs the following reports:

- a side-by-side comparison of data set attributes between the source and target libraries
- a side-by-side comparison of data set engine or host information
- a comparison of data set contents produced by the CONTENTS procedure.

51




You must issue the ODS libref to store the SAS data sets created by the validation tools.

Partial Output

The SAS System
contents of source library before and after MIGRATE
(OK indicates member was present in source lib before and after MIGRATE)

| Obs | name | Mem Type | result |
|---|---|---|---|
| 1 | CASE | DATA | OK |
| 2 | DEMOTRADES | DATA | OK |
| 3 | FLIGHTDELAYS | DATA | OK |
| 4 | FLIGHTSCHEDULE | DATA | OK |
| 5 | FREQUENTFLYERS | DATA | OK |
| 6 | FREQUENTFLYERS2 | DATA | OK |
| 7 | INTERNATIONALFLIGHTS | DATA | OK |
| 8 | MARCHFLIGHTS | DATA | OK |
| 9 | MECHANICSLEVEL1 | DATA | OK |
| 10 | MECHANICSLEVEL2 | DATA | OK |
| 11 | MECHANICSLEVEL3 | DATA | OK |
| 12 | ONE | DATA | OK |
| 13 | PAYROLLCHANGES | DATA | OK |
| 14 | PAYROLLMASTER | DATA | OK |
| 15 | PAYROLLMASTER2 | DATA | OK |
| 16 | PUBLICCANCELS | DATA | OK |
| 17 | R_TRANSACTION | DATA | OK |
| 18 | STAFFCHANGES | DATA | OK |
| 19 | STAFFMASTER | DATA | OK |
| 20 | STAFFMASTER2 | DATA | OK |
| 21 | SUPERVISORS | DATA | OK |
| 22 | TWO | DATA | OK |
| 23 | PAYROLLMASTER | INDEX | OK |

```
                                The SAS System
                                  DEMOTRADES
                      Number 2 of 22 data sets in source library
                    Differences in PROC CONTENTS header information
                    Note:  all other header information was the same
                Number 1 of 3 reports for this data set (from checkdata macro)

                  Obs    attribute    source              target

                    1     Encoding     Default    wlatin1  Western (Windows)
```

```
                                The SAS System
                                  DEMOTRADES
                      Number 2 of 22 data sets in source library
                  Differences in PROC CONTENTS engine/host information
                  Note:  all other engine/host information was the same
                Number 2 of 3 reports for this data set (from checkdata macro)

                         Obs       attribute        source      target

                          1     Host Created       WIN_PRO     XP_PRO
                          2     Release Created    8.0202M0    9.0101M0
```

```
                                The SAS System
                                  DEMOTRADES
                      Number 2 of 22 data sets in source library
                                 PROC COMPARE of data
                Number 3 of 3 reports for this data set (from checkdata macro)

                                The COMPARE Procedure
                      Comparison of LIB1.DEMOTRADES with LIB2.DEMOTRADES
                                  (Method=EXACT)

NOTE: No unequal values were found. All values compared are exactly equal.
```

For more information on PROC MIGRATE and to load the validation tools, use this link:

http://support.sas.com/rnd/migration/resources/procmigrate/validtools.html

## Module 2 Summary

The following procedures support multi-threaded processing:

- MEANS/SUMMARY
- REPORT
- SQL
- TABULATE
- SORT.

```
OPTIONS THREADS | NOTHREADS;
```

55

## Module 2 Summary

To specify the number of CPUs available:

```
OPTIONS CPUCOUNT=1-1024 | ACTUAL;
```

PROC FORMAT now supports format and informat names up to 32 characters long, including the $ for character names.

56

## Module 2 Summary

The DATECOPY option in PROC SORT retains the original creation and modification date of your data set.

```
PROC SORT DATA=data-set DATECOPY;
     BY variable(s);
RUN;
```

57

## Module 2 Summary

The DUPOUT= option enables you to write duplicate observations to a new SAS data set. It must be used in conjunction with the NODUPRECS or NODUPKEY option.

```
PROC SORT DATA=data-set  NODUPRECS | NODUPKEY
                  DUPOUT=data-set-2 <OUT=data-set-3>;
     BY variable(s);
RUN;
```

58

## Module 2 Summary

PROC FREQ supports new options for crosstabulations.

```
TABLES variables / CONTENTS=string;
```

```
TABLES variable*variable / CROSSLIST;
```

The new MIGRATE procedure migrates your data sets to the current SAS release.

```
PROC MIGRATE IN=libref-1 OUT=libref-2 <options>;
RUN;
```

59

# Module 3 SAS Language Enhancements

# 3.1 SAS System Options

## Objectives

- Customize your SAS session with new system options.

3

## DTRESET | NODTRESET System Options

DTRESET

specifies that SAS update the date and time in the titles of the SAS log and the listing file.

NODTRESET

specifies that SAS **not** update the date and time in the titles of the SAS log and the listing file. Instead, the time and date that you invoked your SAS session is displayed.

The default is NODTRESET.

5

## DATESTYLE= System Option

DATESTYLE=
identifies the order of month, day, and year when the ANYDTDTE., ANYDTDTM., or ANYDTTME. informat data is ambiguous.

**DATESTYLE**=MDY | MYD | YMD | YDM | DMY | DYM | LOCALE

The default value is LOCALE.

6

The ANYDTDTE., ANYDTDTM., and ANYDTTME. informats are referenced later in this module.

## LOCALE= System Option

LOCALE=
specifies a set of attributes in a SAS session that reflect the language, local conventions, and culture for a geographical region to be used for such features as formatting dates and referencing currency.

7

The LOCALE= option applies to the National Language formats that are referenced later in this module.

Examples of LOCALE= and the corresponding DATESTYLE= settings:

| LOCALE= | DATESTYLE= |
|---|---|
| Dutch_Belgium | DMY |
| English_UnitedKingdom | DMY |
| English_UnitedStates | MDY |
| French_Canada | DMY |
| Hungarian_Hungary | YMD |
| Japanese_Japan | YMD |
| Spanish_PuertoRico | MDY |

See the National Language Support documentation for a complete listing.

## VALIDFMTNAME= System Option

VALIDFMTNAME=
controls the length of informat and format names that you can use when you create new SAS data sets and format catalogs.

**VALIDFMTNAME**= LONG | FAIL | WARN

9

## VALIDFMTNAME= System Option

LONG
specifies that informat and format names can be up to 32 alphanumeric characters. Names must begin with A-Z or an underscore (_). LONG is the default.

FAIL writes an error message to the log if you attempt to use or create a format or informat name that is longer than 8 characters.

WARN
writes a warning message to the log to remind you that the format or informat you used or created cannot be used with releases prior to SAS®9.

10

# 3.2 SAS Functions and Perl Regular Expressions

## Objectives

- Concatenate data with the CAT functions.
- Change the case of character data with the PROPCASE function.
- Search for character strings with the COUNT and FIND functions.
- Search for pattern matches with Perl regular expressions.

15

## SAS Functions

**lwbase.employee** (Partial)

| LASTNAME | FIRSTNAME | MIDDLENAME |
|---|---|---|
| O'REILLY | RICHARD | BINGHAM |
| KRAUSE | KARL-HEINZ | G. |
| LOVELL | WILLIAM | SINCLAIR |
| RODRIGUES | JUAN | M. |
| NISHIMATSU-LYNCH | CAROL | ANNE |

| LASTNAME $16 | FIRSTNAME $12 | MIDDLENAME $9 |
|---|---|---|
| | | |

How can you create a full name?

16

## The CAT Function

The CAT function concatenates character strings **without** removing leading or trailing blanks.

**CAT**(*string-1, string-2, … string-n*)

*string* specifies a character constant, variable, or expression.

18

## The CAT Function

```
data names;
   set lwbase.employee;
   FullName=cat(firstname, middlename,
                lastname);
proc print data=names;
run;
```

Partial Output

| FullName | | |
|---|---|---|
| RICHARD | BINGHAM | O'REILLY |
| KARL-HEINZ | G. | KRAUSE |
| WILLIAM | SINCLAIR | LOVELL |
| JUAN | M. | RODRIGUES |
| CAROL | ANNE | NISHIMATSU-LYNCH |

19 m03s2d1

The default length of the new variable is 200.

Equivalent code:

```
FullName=firstname || middlename || lastname;
```

## The CATT Function

The CATT function concatenates character strings and removes **trailing blanks** only.

**CATT**(*string-1, string-2, … string-n*)

*string* specifies a character constant, variable, or expression.

20

## The CATT Function

```
data names;
   set lwbase.employee;
   FullName=catt(firstname, middlename,
                 lastname);
proc print data=names;
run;
```

Partial Output

```
                FullName

RICHARDBINGHAMO'REILLY
KARL-HEINZG.KRAUSE
WILLIAMSINCLAIRLOVELL
JUANM.RODRIGUES
CAROLANNENISHIMATSU-LYNCH
```

m03s2d2

21

The default length of the new variable is 200.

Equivalent code:

```
FullName=trim(firstname) || trim(middlename) ||
         lastname;
```

## The CATT Function – Numeric Variable

```
data catnumber;
   set lwbase.staff_salary;
   Key=catt(State, Num_Employees, Jobcode);
run;
proc print data=catnumber;
   var State Num_Employees Jobcode Key;
run;
```

m03s2d2a

22

## The CATT Function – Numeric Variable

Partial Output

CATT Function with a Numeric

| Obs | State | num_ employees | Job Code | Key |
|---|---|---|---|---|
| 1 | CT | 1 | FA1 | CT1FA1 |
| 2 | CT | 2 | FA2 | CT2FA2 |
| 3 | CT | 2 | FA3 | CT2FA3 |
| 4 | CT | 1 | ME1 | CT1ME1 |
| 5 | CT | 3 | ME2 | CT3ME2 |

23

Equivalent code:

```
data catnumber;
   set lwbase.staff_salary;
   Key=trim(State) || trim(left(put(Num_Employees, 2.)))
       || JobCode;
run;
```

## The CATS Function

The CATS function concatenates character strings and removes **leading** and **trailing** blanks.

**CATS**(*string-1, string-2, … string-n*)

*string* specifies a character constant, variable, or expression.

24

## The CATS Function

```
data names;
   set lwbase.employee;
   FullName=cats(firstname, middlename,
                 lastname);
proc print data=names;
run;
```

Partial Output

```
FullName

RICHARDBINGHAMO'REILLY
KARL-HEINZG.KRAUSE
WILLIAMSINCLAIRLOVELL
JUANM.RODRIGUES
CAROLANNENISHIMATSU-LYNCH
```

25 m03s2d3

The default length of the new variable is 200.

Equivalent code:

```
FullName=trim(left(firstname)) ||
         trim(left(middlename)) || left(lastname);
```

## The CATX Function

The CATX function concatenates character strings, removes **leading** and **trailing** blanks, and **inserts separators**.

**CATX**(*separator*, *string-1, … string-n*)

*separator* specifies a character string that is used as a delimiter or separator between concatenated strings.

*string* specifies a character constant, variable, or expression.

26

## The CATX Function

```
data names;
   set lwbase.employee;
   FullName=catx(' ',firstname, middlename,
                 lastname);
proc print data=names;
run;
```

Partial Output

| FullName |
|---|
| RICHARD BINGHAM O'REILLY |
| KARL-HEINZ G. KRAUSE |
| WILLIAM SINCLAIR LOVELL |
| JUAN M. RODRIGUES |
| CAROL ANNE NISHIMATSU-LYNCH |

27 m03s2d4

The default length of the new variable is 200.

Equivalent code:

```
FullName=trim(left(firstname)) || ' ' ||
         trim(left(middlename)) || ' ' ||
         left(lastname);
```

You can also provide a variable list when you use any of the CAT functions:

```
FullName=cat(of name1-name3);
FullName=catx(' ', prefix, of name1-name3);
FullName=catx(' ', prefix, of firstname--lastname)
```

## Using the CATX Function

m03s2d4a.sas

```
options dtreset;
data names;
   set lwbase.employee;
   FullName=catx(' ',firstname, middlename, lastname);
   FullName2=catx(' ', trim(left(lastname))||',',
                  firstname, middlename);
proc print data=names;
   var FullName FullName2;
run;
```

Partial Output

```
                        The SAS System          1:38 Thursday, July 22, 2004

   Obs    FullName                          FullName2

     1    G. ANDREA WOLF-PROVENZA           WOLF-PROVENZA, G. ANDREA
     2    S. RACHAEL HAMMERSTEIN            HAMMERSTEIN, S. RACHAEL
     3    CHRIS J. VARGAS                   VARGAS, CHRIS J.
     4    VLADIMIR JORAN MEDER              MEDER, VLADIMIR JORAN
     5    CLARA JANE CHOULAI                CHOULAI, CLARA JANE
     6    STEPHANIE J. HEMESLY              HEMESLY, STEPHANIE J.
     7    MARIE-LOUISE TERESA WACHBERGER    WACHBERGER, MARIE-LOUISE TERESA
     8    PRUDENCE VALENTINE PURINTON       PURINTON, PRUDENCE VALENTINE
     9    GILBERT IRVINE SMITH              SMITH, GILBERT IRVINE
    10    R. STEPHEN BATTERSBY              BATTERSBY, R. STEPHEN
```

## The PROPCASE Function

The PROPCASE function returns a string in proper (mixed) case.

**PROPCASE**(*string <,delimiter(s)>*)

*string* specifies a character constant, variable, or expression.

*delimiter(s)* specifies any character or characters to be used as a delimiter. The default delimiters are blank, tab, hyphen, open parenthesis, period, and forward slash.

29

## The PROPCASE Function

```
data mixed;
   set names;
   Proper=propcase(fullname);
proc print data=mixed;
run;
```

Partial Output

```
                Proper

Richard Bingham O'reilly
Karl-Heinz G. Krause
William Sinclair Lovell
Juan M. Rodrigues
Carol Anne Nishimatsu-Lynch
```

30

m03s2d5

## The PROPCASE Function

When you specify delimiters, the default delimiters are no longer in effect.

```
data mixed;
   set names;
   Proper=propcase(fullname, " '-");
proc print data=mixed;
run;
```

Partial Output

```
                Proper

Richard Bingham O'Reilly
Karl-Heinz G. Krause
William Sinclair Lovell
Juan M. Rodrigues
```

31 m03s2d6

Be sure to specify a space as your delimiter. If your delimiter is a character string, it must be enclosed in quotes.

**Exercise: Refer to your course workbook for Exercises 5a and 5b.**

## The COUNT Function

The COUNT function counts the number of times that a specific set of characters appears within a specified character string.

**COUNT**(*string*,*substring*,<*modifier(s)*>)

The COUNT function searches *string* from **left** to **right** for the number of occurrences of the specified *substring* and returns that number of occurrences. If *substring* is **not** found in *string*, COUNT returns a value of **0**.

33

*string* specifies a character constant, variable, or expression.

*substring* is a character constant, variable, or expression that specifies the substring of characters to count in *string*.

*modifier(s)* is a character constant, variable, or expression that specifies one or more modifiers. The following modifiers can be in uppercase or lowercase:

I ignores character case during the count. If this modifier is not specified, COUNT counts only the character substrings with the same case as the characters in *substring*.

T trims trailing blanks from *string* and *substring*.

If *string*, *substring*, or *modifier(s)* is a character constant, enclose it in quotes.

## The COUNT Function

**lwbase.medals** (Partial)

| Medals | Sport |
|---|---|
| Silver,Silver,Gold,Bronze | Swimming |
| Silver,bronze,Gold | Wrestling |
| Gold,Gold,gold,Silver,Bronze | Ice Skating |
| Silver,Bronze | Archery |
| Bronze,Bronze,Silver | Track |
| gold,Silver,Bronze | Basketball |

How many gold medals were awarded in each sport?

34

## The COUNT Function

```
data CountGold;
   set lwbase.medals;
   Gold=count(medals,'gold',"i");
proc print data=CountGold;
run;
```

Partial Output

```
                        The SAS System
Obs   Medals                          Sport          Gold

 1    Silver,Silver,Gold,Bronze       Swimming          1
 2    Silver,bronze,Gold              Wrestling         1
 3    Gold,Gold,gold,Silver,Bronze    Ice Skating       3
 4    Silver,Bronze                   Archery           0
 5    Bronze,Bronze,Silver            Track             0
 6    gold,Silver,Bronze              Basketball        1
```

35 m03s2d7

To specify both modifiers:

```
Gold=count(Medals, 'gold', "i t");
```

## The FIND Function

The FIND function searches for a specific substring of characters within a specified character string.

**FIND**(*string,substring*<,*modifier(s)*><,*startpos*>)

The FIND function searches *string* for the **first occurrence** of the specified *substring*, and it returns

- the **position** of the substring if it is found in *string*
- a value of **0** if the substring is **not** found.

37

*string* specifies a character constant, variable, or expression.

*substring* is a character constant, variable, or expression that specifies the substring of characters to count in *string*.

*modifier(s)* is a character constant, variable, or expression that specifies one or more modifiers. The following modifiers can be in uppercase or lowercase:

I ignores character case during the count. If this modifier is not specified, FIND only finds character substrings with the same case as the characters in *substring*.

T trims trailing blanks from *string* and *substring*.

*startpos* is an integer that specifies the position at which the search should start and the direction of the search. The default is position 1. The starting position can be a negative number.

If *string*, *substring*, or *modifier(s)* is a character constant, enclose it in quotes.

## The FIND Function

**lwbase.medals** (Partial)

| Medals | Sport |
|---|---|
| Silver,Silver,Gold,Bronze | Swimming |
| Silver,bronze,Gold | Wrestling |
| Gold,Gold,gold,Silver,Bronze | Ice Skating |
| Silver,Bronze | Archery |
| Bronze,Bronze,Silver | Track |
| gold,Silver,Bronze | Basketball |

Which sports received a gold medal after the first medal was awarded?

38

## The FIND Function

```
data NextGold;
   set lwbase.medals;
   Gold=find(medals,'gold',"i", 5);
proc print data=NextGold;
run;
```

Partial Output

| Obs | Medals | Sport | Gold |
|---|---|---|---|
| 1 | Silver,Silver,Gold,Bronze | Swimming | 15 |
| 2 | Silver,bronze,Gold | Wrestling | 15 |
| 3 | Gold,Gold,gold,Silver,Bronze | Ice Skating | 6 |
| 4 | Silver,Bronze | Archery | 0 |
| 5 | Bronze,Bronze,Silver | Track | 0 |
| 6 | gold,Silver,Bronze | Basketball | 0 |

39 m03s2d8

To specify both modifiers:

```
Gold=find(Medals, 'gold', "i t", 5);
```

**Exercise: Refer to your course workbook for Exercise 5c.**

## Perl Regular Expressions

A Perl regular expression is a **pattern**, or **template**, to be matched against a string.

Selected Perl metacharacters (or symbols):

| Symbol | Meaning | Example |
|---|---|---|
| **/** | start/end Perl expression | / *Perl-expression* / |
| **\w** | any word character | A-Z, a-z, _, 0-9 |
| **\d** | any digit | 0-9 |
| **\s** | white-space character | space, tab, carriage return |
| **( )** | grouping | |
| **+** | match one or more times | |
| **{*n*}** | match exactly *n* times | {2} means match two times |
| **[ ]** | character class | |
| **^** | negate character class | [^,] means not a comma |
| **$*n*** | capture buffer | $1 |

42

Perl: Practical Extraction and Report Language

## The PRXMATCH Function

The PRXMATCH function uses a Perl regular expression to search for a **pattern match** and returns the **starting** position at which the pattern is found.

**PRXMATCH**(*Perl-regular-expression, source*)

*Perl-regular-expression*
specifies a Perl regular expression.

*source*
specifies the character expression that you want to search.

43

## The PRXMATCH Function

Find all the names that have a valid social security number construct of *ddd-dd-dddd*.

**lwbase.names** (Partial)

| Name | SSN |
|---|---|
| SANCHEZ, CARLOS | |
| O'BRIEN, MARY | 651-14-7894 |
| PYLES, JANE | 99-7845-251 |
| HOFFMAN, VALERIE | 877-89-3216 |
| DAWN, JENNIFER | |
| VAN HUSEN, JEFF | 364-9A-7412 |
| SIM-SMITH, ANGELA | |
| TIMMONS, DAVID | 641-78-6143 |
| BENJAMIN, CATHERINE | 492-11-9998 |

44

## The PRXMATCH Function

```
data validssn;
   set lwbase.names;
   where prxmatch('/\d{3}-\d{2}-\d{4}/', ssn);
proc print data=validssn;
   var Name SSN;
run;
```

Partial Output

```
                The SAS System

Obs    Name                      SSN
  1    O'BRIEN, MARY          651-14-7894
  2    HOFFMAN, VALERIE       877-89-3216
  3    TIMMONS, DAVID         641-78-6143
  4    BENJAMIN, CATHERINE    492-11-9998
  5    BELLUM, SARAH          987-41-8145
  6    GARCIA, TRACY          333-44-4789
```

45 m03s2d9

## The PRXMATCH Function

Starts the regular expression.

```
data validssn;
   set lwbase.names;
   where prxmatch('/\d{3}-\d{2}-\d{4}/', ssn);
proc print data=validssn;
   var Name SSN;
run;
```

Partial Output

```
                The SAS System

Obs    Name                      SSN
  1    O'BRIEN, MARY          651-14-7894
  2    HOFFMAN, VALERIE       877-89-3216
  3    TIMMONS, DAVID         641-78-6143
  4    BENJAMIN, CATHERINE    492-11-9998
  5    BELLUM, SARAH          987-41-8145
  6    GARCIA, TRACY          333-44-4789
```

46

## The PRXMATCH Function

Look for exactly three digits.

```
data validssn;
   set lwbase.names;
   where prxmatch('/\d{3}-\d{2}-\d{4}/', ssn);
proc print data=validssn;
   var Name SSN;
run;
```

Partial Output

```
                The SAS System

Obs    Name                        SSN
  1    O'BRIEN, MARY           651-14-7894
  2    HOFFMAN, VALERIE        877-89-3216
  3    TIMMONS, DAVID          641-78-6143
  4    BENJAMIN, CATHERINE     492-11-9998
  5    BELLUM, SARAH           987-41-8145
  6    GARCIA, TRACY           333-44-4789
```

47

## The PRXMATCH Function

Followed by a dash

```
data validssn;
   set lwbase.names;
   where prxmatch('/\d{3}-\d{2}-\d{4}/', ssn);
proc print data=validssn;
   var Name SSN;
run;
```

Partial Output

```
                The SAS System

Obs    Name                        SSN
  1    O'BRIEN, MARY           651-14-7894
  2    HOFFMAN, VALERIE        877-89-3216
  3    TIMMONS, DAVID          641-78-6143
  4    BENJAMIN, CATHERINE     492-11-9998
  5    BELLUM, SARAH           987-41-8145
  6    GARCIA, TRACY           333-44-4789
```

48

## The PRXMATCH Function

Followed by exactly two digits

```
data validssn;
   set lwbase.names;
   where prxmatch('/\d{3}-\d{2}-\d{4}/', ssn);
proc print data=validssn;
   var Name SSN;
run;
```

Partial Output

```
                The SAS System

Obs    Name                        SSN
  1    O'BRIEN, MARY           651-14-7894
  2    HOFFMAN, VALERIE        877-89-3216
  3    TIMMONS, DAVID          641-78-6143
  4    BENJAMIN, CATHERINE     492-11-9998
  5    BELLUM, SARAH           987-41-8145
  6    GARCIA, TRACY           333-44-4789
```

49

## The PRXMATCH Function

Followed by another dash

```
data validssn;
   set lwbase.names;
   where prxmatch('/\d{3}-\d{2}-\d{4}/', ssn);
proc print data=validssn;
   var Name SSN;
run;
```

Partial Output

```
                The SAS System

Obs    Name                          SSN
  1    O'BRIEN, MARY            651-14-7894
  2    HOFFMAN, VALERIE         877-89-3216
  3    TIMMONS, DAVID           641-78-6143
  4    BENJAMIN, CATHERINE      492-11-9998
  5    BELLUM, SARAH            987-41-8145
  6    GARCIA, TRACY            333-44-4789
```

50

## The PRXMATCH Function

Followed by exactly four digits

```
data validssn;
   set lwbase.names;
   where prxmatch('/\d{3}-\d{2}-\d{4}/', ssn);
proc print data=validssn;
   var Name SSN;
run;
```

Partial Output

```
                The SAS System

Obs    Name                          SSN
  1    O'BRIEN, MARY            651-14-7894
  2    HOFFMAN, VALERIE         877-89-3216
  3    TIMMONS, DAVID           641-78-6143
  4    BENJAMIN, CATHERINE      492-11-9998
  5    BELLUM, SARAH            987-41-8145
  6    GARCIA, TRACY            333-44-4789
```

51

## The PRXMATCH Function

Ends regular expression.

```
data validssn;
   set lwbase.names;
   where prxmatch('/\d{3}-\d{2}-\d{4}/', ssn);
proc print data=validssn;
   var Name SSN;
run;
```

Partial Output

```
                The SAS System

Obs    Name                          SSN
  1    O'BRIEN, MARY            651-14-7894
  2    HOFFMAN, VALERIE         877-89-3216
  3    TIMMONS, DAVID           641-78-6143
  4    BENJAMIN, CATHERINE      492-11-9998
  5    BELLUM, SARAH            987-41-8145
  6    GARCIA, TRACY            333-44-4789
```

52

## The PRXMATCH Function

Variable containing pattern to be tested

```
data validssn;
   set lwbase.names;
   where prxmatch('/\d{3}-\d{2}-\d{4}/', ssn);
proc print data=validssn;
   var Name SSN;
run;
```

Partial Output

```
                The SAS System

Obs    Name                        SSN
  1    O'BRIEN, MARY           651-14-7894
  2    HOFFMAN, VALERIE        877-89-3216
  3    TIMMONS, DAVID          641-78-6143
  4    BENJAMIN, CATHERINE     492-11-9998
  5    BELLUM, SARAH           987-41-8145
  6    GARCIA, TRACY           333-44-4789
```

53

Equivalent code:

```
where ssn like '___-__-___';
```

However, the LIKE operator would select `364-9A-7412` as a valid value for **SSN** because it cannot distinguish letters from digits. You would need to add the VERIFY function:

```
where ssn like '___-__-____' and
      verify(ssn,'0,1,2,3,4,5,6,7,8,9,-') = 0;
```

To look for social security numbers that do not match the pattern, use this statement:

```
where not prxmatch ('/\d{3}-\d{2}-\d{4}/', ssn);
```

## The PRXCHANGE Function (Optional)

The PRXCHANGE function uses a Perl regular expression to perform a pattern-match replacement.

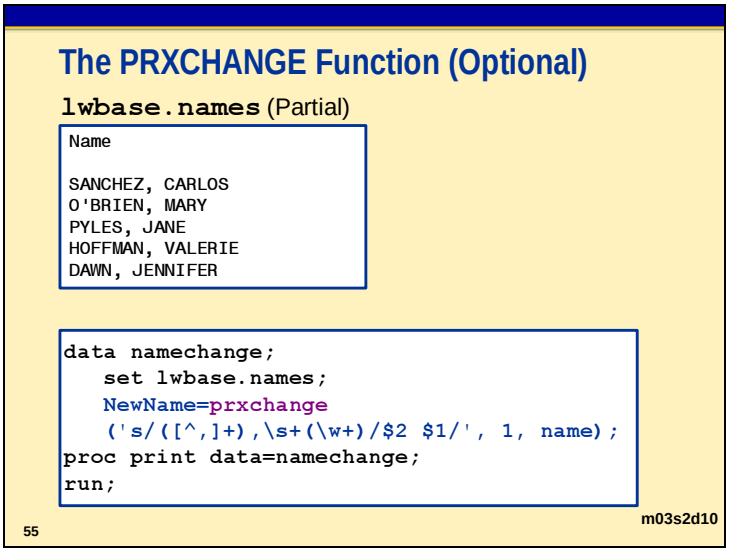
**PRXCHANGE**(*Perl-regular-expression, times, source*)

Perl-regular expression
: specifies a Perl regular expression.

times
: is the numeric value that specifies the number of times to search for a match and replace a matching pattern.

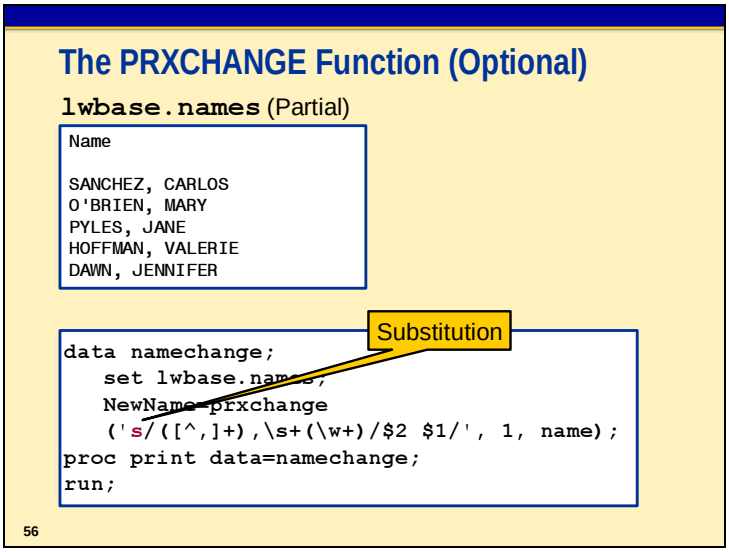
source
: specifies the character expression that you want to search.

54




If the Perl regular expression spans two or more lines, use quotes and the || operator:

```
NewName=prxchange("s/([^,]+),\s+"||
                  "(\w+)/$2 $1/", 1, name);
```



## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

Start of the regular expression

57

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

Start capture buffer #1 for last name.

58

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

- The **[ ]** matches any one of the characters inside the square braces.
- The **^** means NOT, and the **,** means to look for a comma.
- **[^,]** means to look for a character that is not a comma.

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

59

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

The **+** means to match one or more times the preceding subexpression. **[^,]+** means to match non-comma characters one or more times.

```
data namec
   set lwbase.n
   NewName=prx      nge
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

60

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

End of capture buffer #1

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

61

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

Look for a comma.

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

62

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

Look for one or more space(s).

63

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

Start capture buffer #2 to look for first name.

64

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

The **\w+** means to look for one or more word characters.

65

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

End capture buffer #2.

66

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

The end of the regexp and start of the replacement text

67

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

**$2** means to insert the match for capture buffer #2, which contains the first name.

68

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

Space means to insert a space.

69

## The PRXCHANGE Function (Optional)

**lwbase.names** (Partial)

```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1', 1, name);
proc print data=namechange;
run;
```

**$1** means to insert the match for capture buffer #1, which contains the last name.

70

## The PRXCHANGE Function (Optional)

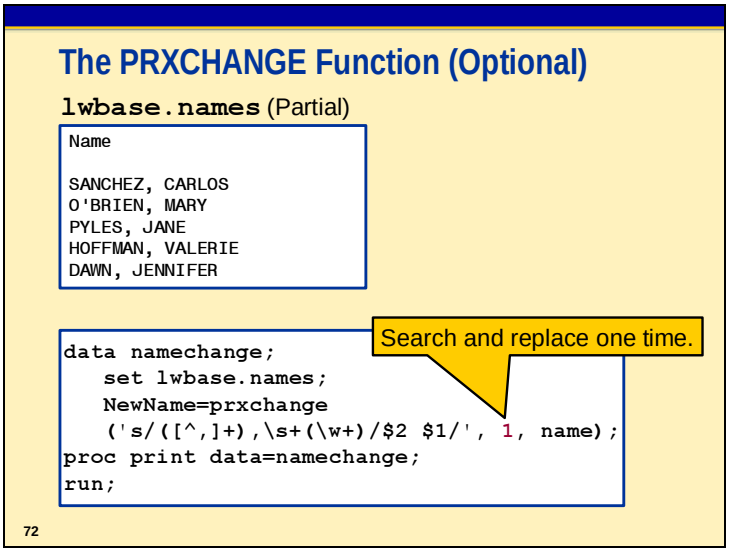
**lwbase.names** (Partial)

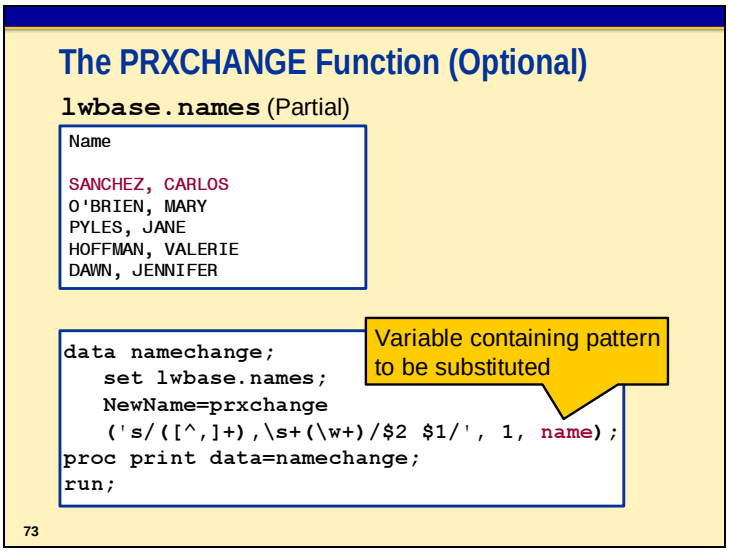
```
Name

SANCHEZ, CARLOS
O'BRIEN, MARY
PYLES, JANE
HOFFMAN, VALERIE
DAWN, JENNIFER
```

```
data namechange;
   set lwbase.names;
   NewName=prxchange
   ('s/([^,]+),\s+(\w+)/$2 $1/', 1, name);
proc print data=namechange;
run;
```

The end of the replacement text

71




Equivalent code:

```
data namechange(drop=First Last);
   set lwbase.names;
   length First Last $20;
   First=scan(Name, 2, ' ');
   Last=scan(Name,1, ', ');
   NewName=trim(First) || ' ' || Last;
proc print data=namechange;
run;
```

## The PRXCHANGE Function (Optional)

Partial Output

The SAS System

| Obs | Name | ID | NewName |
|---|---|---|---|
| 1 | SANCHEZ, CARLOS | 9309 | CARLOS SANCHEZ |
| 2 | O'REILY, MARY | 1234 | MARY O'REILY |
| 3 | PYLES, JANE | 9813 | JANE PYLES |
| 4 | HOFFMAN, VALERIE | 6344 | VALERIE HOFFMAN |
| 5 | DAWN, JENNIFER | 1823 | JENNIFER DAWN |
| 6 | VAN HUSEN, JEFF | 1887 | JEFF VAN HUSEN |
| 7 | SIM-SMITH, ANGELA | 5190 | ANGELA SIM-SMITH |
| 8 | TIMMONS, DAVID | 9234 | DAVID TIMMONS |
| 9 | BENJAMIN, CATHERINE | 2557 | CATHERINE BENJAMIN |
| 10 | WINDSOR, STEPHEN | 8311 | STEPHEN WINDSOR |
| 11 | RICHARDSON, LARRY | 2098 | LARRY RICHARDSON |
| 12 | BELLUM, SARAH | 4333 | SARAH BELLUM |

74

For more information on Perl Regular Expressions, go to http://support.sas.com/rnd/base/topics/datastep/perl_regexp/.

**Exercise: Refer to your course workbook for Exercise 5d.**

# 3.3 SAS Formats and Informats

## Objectives

- Read raw data with the ANYDT. family of informats.
- Display SAS data with new NLS formats.

79

## The ANYDT. Informats

**datestimes.dat** (Partial)

```
27jan2003           08:44:46      01JAN2002:08:44:46 PM
12/4/2002           12:54:23 AM   5/18/2003
4/30/2002           4:15:51       14DEC2002 04:15:51
October 6, 2002     7:29 PM       October 6, 2002
```

How can you read these fields when the constructs are different?

80

## The ANYDTDTE. Informat

The ANYDTDTE. informat reads various date, time, and datetime constructs in your data field and converts each into a SAS **date** value.

You can specify a length after the ANYDTDTE. informat name. The default value is **9**, and the range is from 1 to 32; for example, `anydtdte9.` illustrates the default.

81

## The ANYDTDTM. Informat

The ANYDTDTM. informat reads various date, time, and datetime constructs in your data field and converts each into a SAS **datetime** value.

You can specify a length after the ANYDTDTM. informat name. The default value is **19**, and the range is from 1 to 32; for example, `anydtdtm19.` illustrates the default.

82

## The ANYDTTME. Informat

The ANYDTTME. informat reads any date, time, or datetime constructs in your data field and converts each into a SAS **time** value.

You can specify a length after the ANYDTTME. informat name. The default value is **8**, and the range is from 1 to 32; for example, `anydttme8.` illustrates the default.

83

## The ANYDT. Informats

The constructs for the ANYDT. informats include

- DATE
- DATETIME
- DDMMYY
- JULIAN
- MMDDYY
- MONYY
- TIME
- YYMMDD
- YYQ.

84

## The ANYDT. Informats

**datestimes.dat** (Partial)

```
27jan2003          08:44:46      01JAN2002:08:44:46 PM
12/4/2002          12:54:23 AM   5/18/2003
4/30/2002          4:15:51       14DEC2002 04:15:51
October 6, 2002    7:29 PM       October 6, 2002
```

```
data various;
   infile 'datestimes.dat' truncover;
   input @1  Dates anydtdte15.
         @20 Times anydttme11.
         @35 Datetimes anydtdtm21.;
proc print data=various;
   format dates date9. times time8.
          datetimes datetime.;
run;
```

85 m03s3d1

## The ANYDT. Informats

Partial Output

The SAS System

| Obs | Dates | Times | Datetimes |
|---|---|---|---|
| 1 | 27JAN2003 | 8:44:46 | 01JAN02:20:44:46 |
| 2 | 04DEC2002 | 0:54:23 | 18MAY03:00:00:00 |
| 3 | 30APR2002 | 4:15:51 | 14DEC02:04:15:51 |
| 4 | 06OCT2002 | 19:29:00 | 06OCT02:00:00:00 |

86

## The NLDATE. Format

The NLDATE. format converts SAS date values to the date string for the locale.

```
options locale=french_canada;
proc print data=various;
   format dates nldate. times time8.
          datetimes datetime.;
run;
```

```
                    The SAS System

Obs         Dates           Times       Datetimes

  1     27 janvier 2003     8:44:46    01JAN02:20:44:46
  2     04 décembre 2002    0:54:23    18MAY03:00:00:00
  3     30 avril 2002       4:15:51    14DEC02:04:15:51
  4     06 octobre 2002    19:29:00    06OCT02:00:00:00
```

87 m03s3d2

## The NLDATEMN. Format

The NLDATEMN. format converts SAS date values to the date string representing the month, based on the locale.

```
options locale=french_canada;
proc print data=various;
   format dates nldatemn. times time8.
          datetimes datetime.;
run;
```

```
                    The SAS System

Obs    Dates          Times       Datetimes

 1     janvier       8:44:46    01JAN02:20:44:46
 2     décembre      0:54:23    18MAY03:00:00:00
 3     avril         4:15:51    14DEC02:04:15:51
 4     octobre      19:29:00    06OCT02:00:00:00
```

88 m03s3d3

**Exercise: Refer to your course workbook for Exercise 6.**

## Module 3 Summary

There are several new system options to control date processing and format usage.

OPTIONS **DTRESET | NODTRESET**;

OPTIONS **DATESTYLE**=MDY | MYD | YMD | YDM | DMY | DYM | LOCALE ;

OPTIONS **LOCALE**=English_UnitedStates;

OPTIONS **VALIDFMTNAME**= LONG | FAIL | WARN;

*continued...*

92

## Module 3 Summary

The CAT family of functions enable you to concatenate strings.

**CAT**(*string-1, string-2, … string-n*)

**CATT**(*string-1, string-2, … string-n*)

**CATS**(*string-1, string-2, … string-n*)

**CATX**(*separator*, *string-1, string-2, … string-n*)

*continued...*

93

## Module 3 Summary

The following functions enable you to convert your data to mixed case and search for substrings:

**PROPCASE**(*string <,delimiter(s)>*)

**COUNT**(*string,substring,<modifier(s)>*)

**FIND**(*string,substring<,modifier(s)><,startpos>*)

*continued...*

94

## Module 3 Summary

Perl regular expressions enable you to search for pattern matches and do pattern substitution.

**PRXMATCH**(*Perl-regular-expression, source*)

**PRXCHANGE**(*Perl-regular-expression, times, source*)

*continued...*

95

## Module 3 Summary

The ANYDTDTE. informat reads various date, time, and datetime constructs in your data field and converts each into a SAS **date** value.

The ANYDTDTM. informat reads various date, time, and datetime constructs in your data field and converts each into a SAS **datetime** value.

The ANYDTTME. informat reads any date, time, or datetime construct in your data field and converts each into a SAS **time** value.

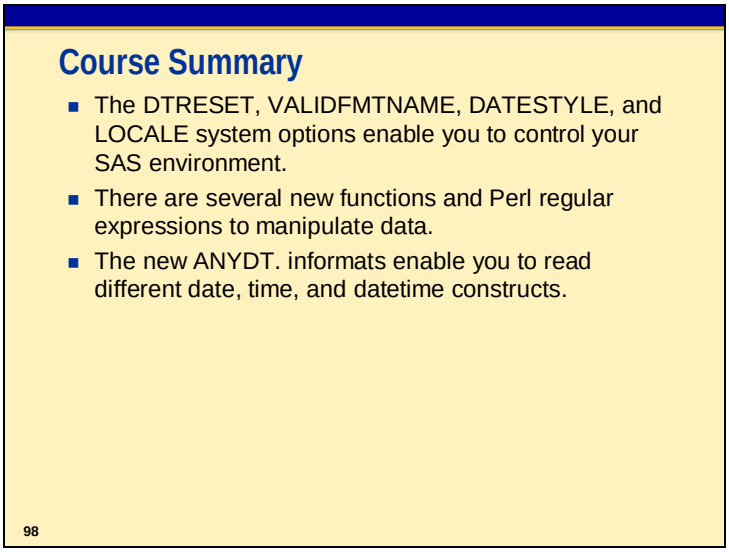
96

## Course Summary

- The Explorer window contains new enhancements to investigate metadata and view your data in Microsoft Excel.
- Many Base SAS procedures support multithreaded processing for potentially faster processing.
- PROC FREQ features new ODS enhancements for crosstabulations.
- PROC SORT allows you to maintain the creation and modification dates, as well as write out duplicate observations to a new data set.
- The new MIGRATE procedure simplifies the migration of your data sets to SAS®9.

*continued...*

97




For more information on SAS®9 courses, go to http://support.sas.com/training.