

A Brief Introduction to SAS®



The SAS System

Obs	Name	Division	Years	Sales	Expense	State
1	CHRIS	H	2	233.11	94.12	WI
2	MARK	H	5	298.12	52.65	WI
3	SARAH	S	6	301.21	65.17	MN
4	PAT	H	4	4009.21	322.12	IL
5	JOHN	H	7	678.43	150.11	WI
6	WILLIAM	H	11	3231.75	644.55	MN
7	ANDREW	S	24	1762.11	476.13	MN
8	BENJAMIN	S	3	201.11	25.21	IL
9	JANET	S	1	98.11	125.32	WI
10	STEVE	H	21	6153.32	1507.12	WI
11	JENNIFER	S	1	542.11	134.24	IL
12	JOY	S	12	2442.22	761.98	WI
13	MARY	S	14	5691.78	2452.11	WI
14	TOM	S	5	5669.12	798.15	MN
15	BETH	H	12	4822.12	982.10	WI



SYSTEMS SEMINAR CONSULTANTS, INC.

2997 Yarmouth Greenway Drive
Madison, WI 53711
(608) 278-9964
train@sys-seminar.com

Introduction

Welcome to Systems Seminar Consultants, Inc.



Systems Seminar Consultants, Inc. is a SAS Alliance Quality Partner™ of SAS. Our team of SAS software experts has a broad base of knowledge and experience working with a variety of complex systems in a number of diverse industry settings. This knowledge and experience is leveraged to help you effectively achieve your business goals.



Free SAS Newsletter

Our popular publication, The Missing Semicolon™, shares SAS software solutions developed by our staff and provides additional technical assistance to our customers.

SAS Training Services

For over 1,000 students each year, we make SAS software easier to understand, use, and support.

- Public training schedules are posted on our web site.
- Private on-site training options are also available.

Introduction

Additional SAS Services



SAS Consulting Services

Our staff of SAS consultants is well-versed in a variety of business areas.

Our specialty areas include:

- Data systems development
- Decision support and business consultation
- Market research and analysis

SAS Help Desk Services

Our team of experts is available to solve your company's daily SAS problems. Call us to develop a customized support plan that meets your company's needs.

For More Information

Call (608) 278-9964 to receive additional information about our services or discuss a specific cost-effective solution for your company. Details can also be found at www.sys-seminar.com.

Presenter



Steven J. First, President



- Over 30 years of SAS experience, including hundreds of manufacturing, retail, government, marketing, and financial applications.
- Over 25 years as President and Founder of SSC
- Founder of WISAS and WISUG
- Invited speaker at local, regional, and international user groups

Objectives



- Identify the components of SAS.
- Review SAS history and design.
- Use SAS pre-written procedures (PROCs) to analyze data.
- Sort data.
- Count values using PROC FREQ.
- Compute simple statistics such as mean, std, sum.
- Summarize data values at different classification levels.
- Format SAS data values.

Introduction to SAS



SAS is an integrated computer system for data analysis and reporting.

Partial list of SAS products:

Base SAS®	the core of SAS system
SAS/STAT®	statistical SAS procedures
SAS/GRAPH®	color graphics on printers and plotters
SAS/FSP®	full screen data management functions
SAS/IML®	interactive matrix language
SAS/OR®	operations research procedures
SAS/ETS®	econometric and time series analysis
SAS/QC®	quality control analysis
SAS/AF®	applications facility for menuing systems
SAS/ACCESS®	access to non-SAS databases (e.g., DB2, etc.)

Introduction to SAS (continued)



SAS/CONNECT®

remote processing and file transfer

SAS/BI tools

controlled Business Intelligence tools

MUCH, MUCH, MORE!

SAS Features



SAS is a full-featured computer system.

Input to SAS can come from:

- raw files
- non-SAS database management systems such as Oracle, DB2
- data lines included in the SAS program
- other SAS datasets already stored on your computer
- SAS datasets located on remote computers

SAS Features (continued)



SAS provides a full programming language to:

- calculate mathematical and statistical results
- move and manipulate numeric and character data
- select and transform data
- rearrange the order of observations (sorting)
- break datasets into smaller pieces (subsetting)
- stack datasets (concatenation)
- combine related data files (merging)

SAS Features (continued)



SAS reducing and summarizing procedures provide:

- summary statistics and tables
- descriptive statistics (e.g., mean, std, sum, etc.)
- frequencies and cross tabulations

BASE SAS provides statistical analysis to:

- calculate univariate statistics and distributions
- calculate correlations and ranks

SAS Features (continued)



SAS reporting features provide:

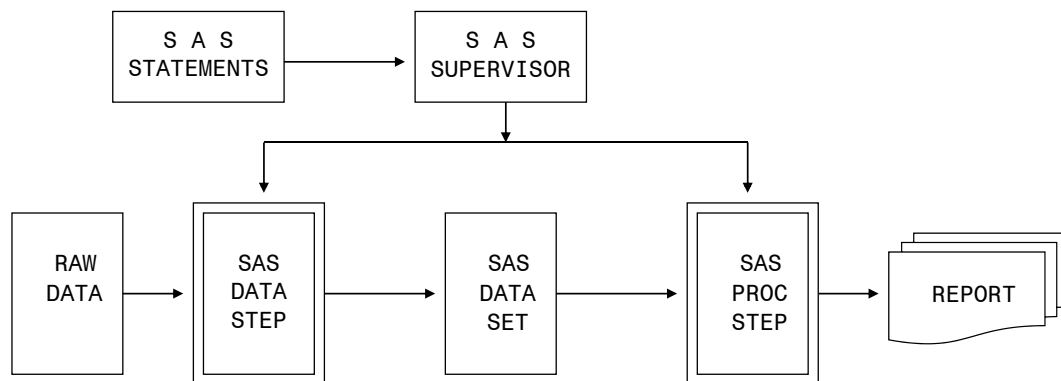
- detail reports via several SAS procs
- high resolution SAS/GRAPH output
- summary reports providing totals at several levels
- tabular reports and mailing labels
- user-written custom reports
- output that can be directed to printers, files, or html files

Structure of SAS



SAS consists of:

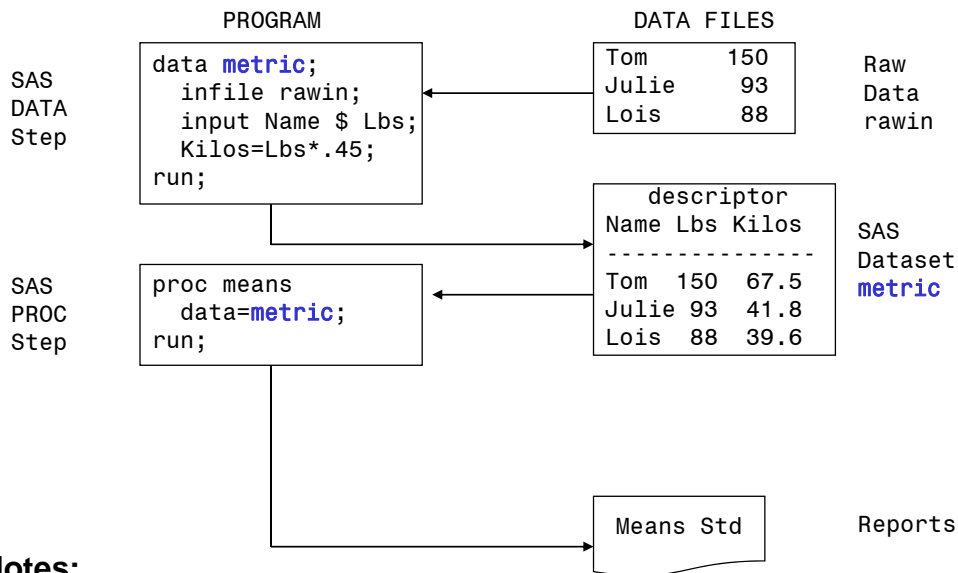
1. a data handling language (DATA step)
2. a library of pre-written procedures (PROC step)



SAS DATA and PROC Steps



DATA steps create SAS datasets, PROC steps process or analyze them.



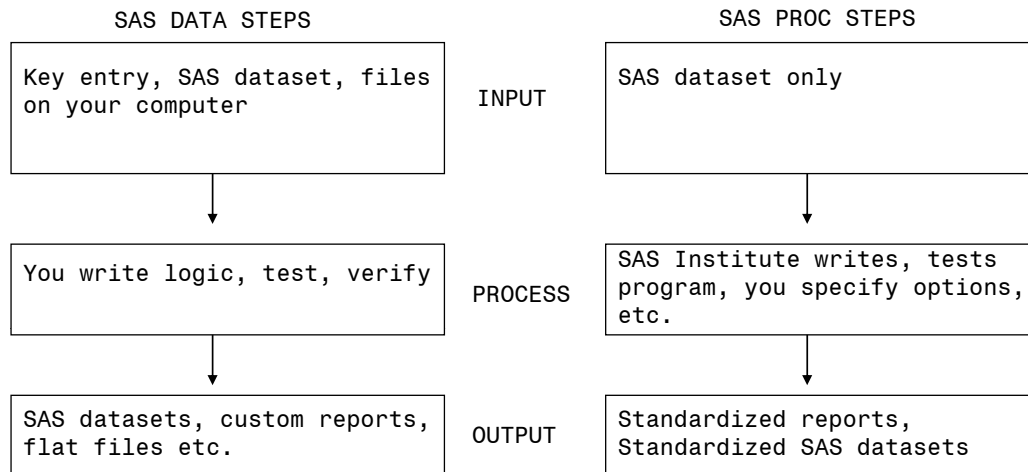
Notes:

- RUN; marks the end of a SAS step.
- More steps may follow.

SAS DATA and PROC Steps (continued)



DATA step and PROC step:



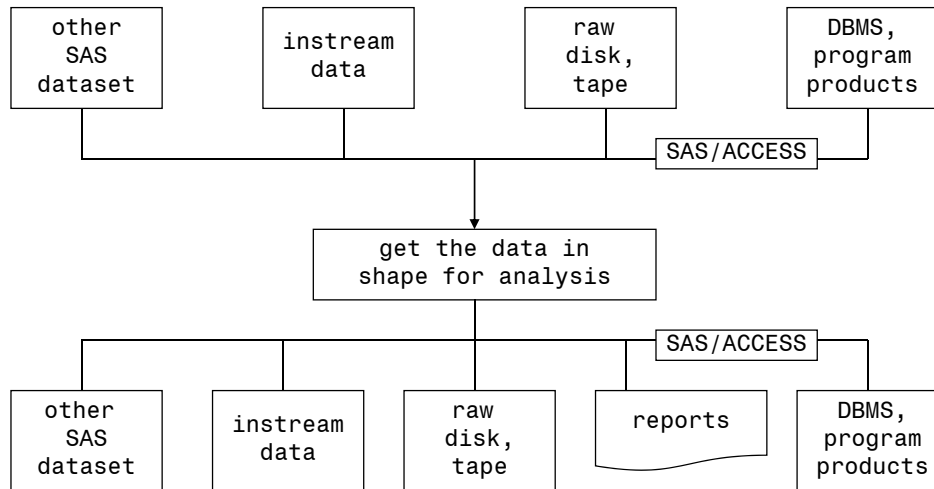
Notes:

- A SAS program consists of one or more SAS steps.

SAS DATA Step Overview



DATA steps can read and write most types of data stored on your computer.



Notes:

- DATA step output is usually a SAS dataset but can be other files.
- Access to non-SAS database management systems requires the SAS/ACCESS product.

Introduction

16

PROC Step Overview



Over 100 programs to analyze and process SAS datasets.

Introductory PROCs:

PROC PRINT	print values in a SAS dataset
PROC REPORT	more complex reports of SAS dataset values
PROC CONTENTS	print SAS dataset descriptor
PROC SORT	rearrange the order of observations
PROC FREQ	count variables and create crosstabs
PROC MEANS/ SUMMARY	summarize and condense SAS datasets
PROC FORMAT	create your own formats and informats
PROC DATASETS	copy, maintain SAS datasets and libraries

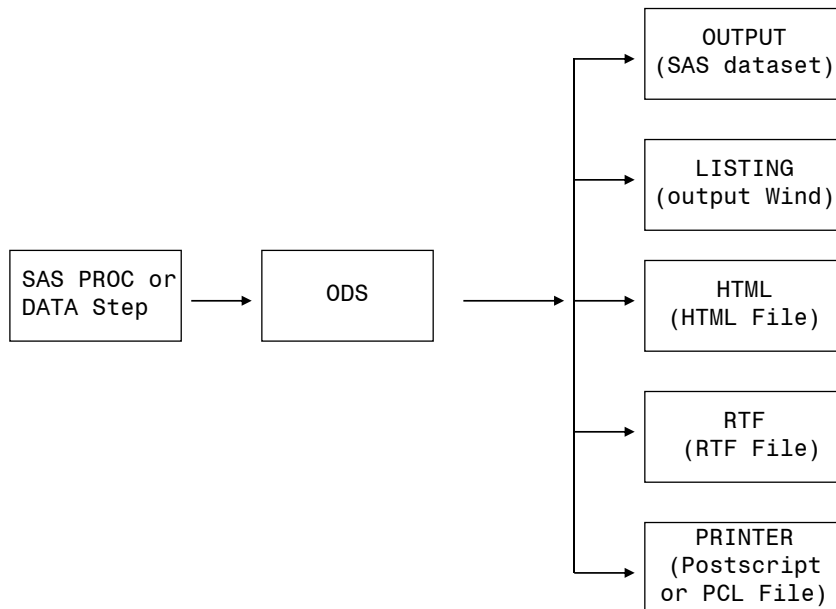
Introduction

17

The Output Delivery System (ODS)



ODS can route reports and output to several destinations.



Processing SAS Datasets



After you build a SAS dataset, you can forget most details except names.

Example:

```
data softsale;
  infile rawin;
  input Name $1-10
        Division $12
        Years 15-16
        Sales 19-25
        Expense 28-34
        State $36-37;
run;

proc print data=softsale;
run;
```

Raw file rawin

CHRIS	H	2	233.11	94.12	WI
.
BETH	H	12	4822.12	982.10	WI

SAS Dataset softsale

Descriptor					
Name	Division	Years	Sales	Expense	State
CHRIS	H	2	233.11	94.12	WI
.
BETH	H	12	4822.12	982.10	WI

A DATALINES Example



You can enter your data as part of the DATA step.

```
Program  [ data softsale;
           input Name $1-10 Division $12 Years 15-16
           Sales 19-25 Expense 28-34 State $36-37;
           datalines;
Raw       [ CHRIS      H   2   233.11   94.12 WI
input     [ MARK       H   5   298.12   52.65 WI
data      [ SARAH      S   6   301.21   65.17 MN
           [ PAT        H   4  4009.21  322.12 IL
           [ .          .   .     .       .       .
           [ BETH      H  12  4822.12  982.10 WI
Program  [ ;
           run;
           proc print data=softsale;
           run;
```

Notes:

- The semicolon (;), that terminates the instream data, **must** be on a separate line.

Reading a Raw Data File With Column Input (continued)



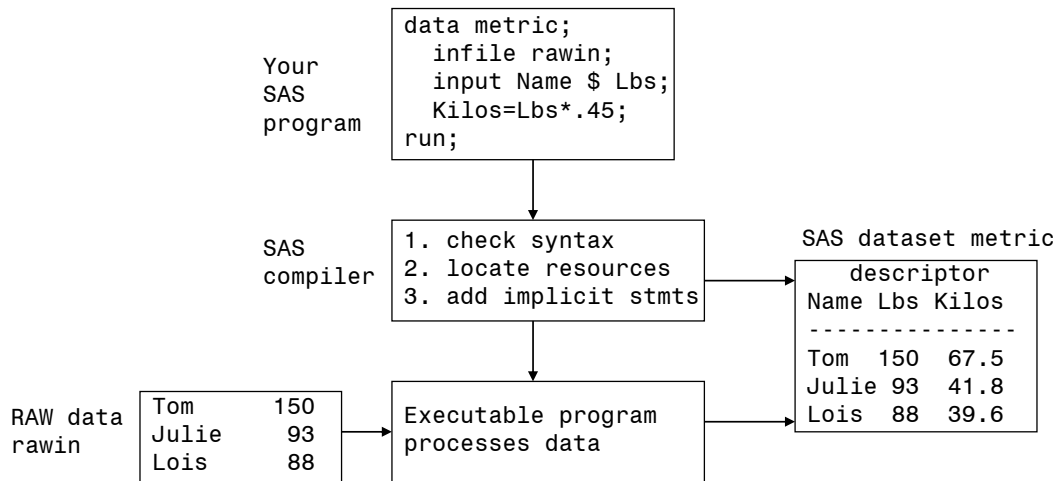
```
data softsale;
  infile rawin;
  input Name $1-10 Division $12 Years 15-16
        Sales 19-25 Expense 28-34 State $36-37;
run;

proc print data=softsale;
run;
```

DATA Step Flow



SAS compiles your program, then executes it.



Compile Processing (continued)



Implicit executable “statements” that SAS adds.

```
data softsale;
  initialize PDV
  infile rawin;
  if at EOF then stop
  input Name          $1-10
         Division     $12
         Years        15-16
         Sales        19-25
         Expense      28-34
         State        $36-37;
```

output to SAS Dataset
goto top of Data Step

run;

Notes:

- You do not code the italicized statements.

Executing the DATA Step



Executing the DATA Step Example

Start → **End**

```

data softsale;
  init buffer, PDV
  infile rawin;
  if at EOF then stop
  input Name $1-10 Division $12 Years 15-16
        Sales 19-25 Expense 28-34 State $36-37;
  output to SAS Dataset
  goto top of Data Step
run;

```

	1	2	3	4	8
1234567890123456789012345678901234567					
CHRIS	H	2	233.11	94.12	WI
MARK	H	5	298.12	52.65	WI
SARAH	S	6	301.21	65.17	MN

Input buffer: 1234567890123456789012345678901234567890...0

Logical Program Data Vector	Name	Division	Years	Sales	Expense	State
(Descriptor)	Name	Division	Years	Sales	Expense	State
(Data)						

Dataset work.softsale

Introduction 24

A List Input Example



The list input allows 'freeform input':

Start → **End**

```

data softsal2;
  infile rawin2;
  input Name $ Division $ Years Sales Expense State $;
run;
proc print data=softsal2; run;

```

	1	2	3
1234567890123456789012345678901234			
CHRIS H 2 233.11 94.12 WI			
MARK H 5 298.12 52.65 WI			
SARAH S 6 301.21 65.17 MN			
PAT H 4 4009.21 322.12 IL			
JOHN H 7 678.43 150.11 WI			
WILLIAM H 11 3231.75 644.55 MN			
ANDREW S 24 1762.11 476.13 MN			
BENJAMIN S 3 201.11 25.21 IL			
JANET S 1 98.11 125.32 WI			
STEVE H 21 6153.32 1507.12 WI			
JENNIFER S 1 542.11 134.24 IL			
JOY S 12 2442.22 761.98 WI			
MARY S 14 5691.78 2452.11 WI			
TOM S 5 5669.12 798.15 MN			
BETH H 12 4822.12 982.10 WI			

fileref rawin2

Introduction 25

A Formatted Input Example



Read the SOFTSALE data again.

Start → **End**

```

data softsale;
  infile rawin;
  input @1 Name $10.
        @12 Division $1.
        @15 Years 2.
        @19 Sales 7.2
        @28 Expense 7.2
        @36 State $2.;

run;
proc print data=softsale;
run;

```

	@1	@12	@15	@19	@28	@36
CHRIS	H	2	233.11	94.12	WI	
MARK	H	5	298.12	52.65	WI	
SARAH	S	6	301.21	65.17	MN	
PAT	H	4	4009.21	322.12	IL	
JOHN	H	7	678.43	150.11	WI	
WILLIAM	H	11	3231.75	644.55	MN	
ANDREW	S	24	1762.11	476.13	MN	
BENJAMIN	S	3	201.11	25.21	IL	
.
BETH	H	12	4822.12	982.10	WI	

\$10. \$1. 2. 7.2 7.2 \$2.

Introduction 26

Processing SAS Datasets



The PROC step:

Syntax:

PROC *procedure-name* **DATA**=SASdataset(*dataset-options*) *proc-options*;

SAS Procedures are programs that:

- are written and tested by someone else
- read existing SAS datasets
- can use a subset of the dataset
- sometimes build new SAS datasets
- sometimes compute various statistics
- usually print results

Processing Data in Subgroups (continued)



Example: Draw separate plots for each value of Division.

```
proc plot data=softsale;  
  plot years*sales;  
      by division;  
quit;
```

Notes:

- If a BY statement is coded, the data **must** be sorted or indexed.
- BY does not **do** the sort; it only recognizes the sort order.
- The BY statement can list several variables.

The WHERE Statement



The WHERE only includes rows that meet the where condition.

Syntax:

WHERE *where condition;*

Sorting Your Data



PROC SORT rearranges the order of the observations in a SAS dataset and replaces the dataset, or creates a new dataset.

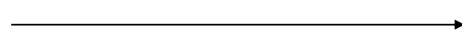
Features:

- single or multiple sort variables
- ascending or descending sequence
- does not print any output
- missing values are lowest value to PROC SORT
- records that are exact duplicates or have duplicate keys can be ignored

How Does PROC SORT Work?



Start



End

```
proc sort data=softsale;  
  by name;  
run;
```

Partial softsale

Name	Division	Years	Sales	Expense	State
CHRIS	H	2	233.11	94.12	WI
TOM	S	5	5669.12	798.15	MN
BETH	H	12	4822.12	982.10	WI

Sortwork
areas



Using OVERWRITE to Save Space



Delete the original dataset before copying the data to the output dataset.

Example:

```
proc sort data=softsale overwrite;  
  by name;  
run;
```

Notes:

- The resulting saving in workspace can be significant.
- Caution: Because the dataset is deleted before the new dataset is written out, data should be temporary, or backed up first.

How Does OVERWRITE Work?



Start



End

```
proc sort data=softsale  
  overwrite;  
  by name;  
run;
```

Partial softsale

Name	Division	Years	Sales	Expense	State
CHRIS	H	2	233.11	94.12	WI
TOM	S	5	5669.12	798.15	MN
BETH	H	12	4822.12	982.10	WI

Sortwork
areas



Printing SAS Data With PROC PRINT



PROC PRINT prints the data values in a SAS dataset.

Features:

- offers automatic page layout and spacing
- offers automatic pagination and numbering
- uses variable names or labels for column labels
- optimizes each page separately
- can produce summaries and control breaks

A Basic PROC PRINT



Print the SOFTSALE dataset.

```
proc print data=softsale;  
run;
```

The SAS System						
Obs	Name	Division	Years	Sales	Expense	State
1	CHRIS	H	2	233.11	94.12	WI
2	MARK	H	5	298.12	52.65	WI
3	SARAH	S	6	301.21	65.17	MN
4	PAT	H	4	4009.21	322.12	IL
5	JOHN	H	7	678.43	150.11	WI
6	WILLIAM	H	11	3231.75	644.55	MN
7	ANDREW	S	24	1762.11	476.13	MN
8	BENJAMIN	S	3	201.11	25.21	IL
9	JANET	S	1	98.11	125.32	WI
10	STEVE	H	21	6153.32	1507.12	WI
11	JENNIFER	S	1	542.11	134.24	IL
12	JOY	S	12	2442.22	761.98	WI
13	MARY	S	14	5691.78	2452.11	WI
14	TOM	S	5	5669.12	798.15	MN
15	BETH	H	12	4822.12	982.10	WI

More Enhancing



Add a footnote.

```
proc print data=softsale noobs;  
  title 'Softco Inc. Sales Summary';  
  footnote 'As of January 2006';  
run;
```

The Resulting Output



Softco Inc. Sales Summary					
Name	Division	Years	Sales	Expense	State
CHRIS	H	2	233.11	94.12	WI
MARK	H	5	298.12	52.65	WI
SARAH	S	6	301.21	65.17	MN
PAT	H	4	4009.21	322.12	IL
JOHN	H	7	678.43	150.11	WI
WILLIAM	H	11	3231.75	644.55	MN
ANDREW	S	24	1762.11	476.13	MN
BENJAMIN	S	3	201.11	25.21	IL
JANET	S	1	98.11	125.32	WI
STEVE	H	21	6153.32	1507.12	WI
JENNIFER	S	1	542.11	134.24	IL
JOY	S	12	2442.22	761.98	WI
MARY	S	14	5691.78	2452.11	WI
TOM	S	5	5669.12	798.15	MN
BETH	H	12	4822.12	982.10	WI

As of January 2006

Selecting the Columns for PROC PRINT



The VAR can select columns after the ID variable.

```
proc print data=softsale;  
  title 'Softco Inc. Sales Summary';  
  footnote 'As of January 2006';  
  id name;  
  var sales;  
run;
```

The Results



Softco Inc. Sales Summary

Name	Sales
CHRIS	233.11
MARK	298.12
SARAH	301.21
PAT	4009.21
JOHN	678.43
WILLIAM	3231.75
ANDREW	1762.11
BENJAMIN	201.11
JANET	98.11
STEVE	6153.32
JENNIFER	542.11
JOY	2442.22
MARY	5691.78
TOM	5669.12
BETH	4822.12

As of January 2006

Changing the OBS Label



OBS= can specify a column header to print instead of "OBS".

```
proc print data=softsale obs= 'Record no.';
  title 'Softco Inc. Sales Summary ';
  footnote 'As of January 2006';
  var name sales;
run;
```

The Resulting Output



```
Softco Inc. Sales Summary

Record
no.    Name      Sales
1      CHRIS     233.11
2      MARK      298.12
3      SARAH     301.21
4      PAT       4009.21
5      JOHN      678.43
6      WILLIAM   3231.75
7      ANDREW    1762.11
8      BENJAMIN  201.11
9      JANET     98.11
10     STEVE     6153.32
11     JENNIFER  542.11
12     JOY       2442.22
13     MARY      5691.78
14     TOM       5669.12
15     BETH      4822.12
```

As of January 2006

A LABEL Example



The LABEL option along with LABEL statements give more descriptive labels as column headers.

```
proc print data=softsale label;  
  title 'Softco Inc. Sales Summary';  
  label sales='Employee Sales';  
  footnote 'As of January 2006';  
  var name sales;  
run;
```

Notes:

- PROC PRINT requires the LABEL option **AND** a LABEL must be assigned.

The Resulting Output



Softco Inc. Sales Summary		
Obs	Name	Employee Sales
1	CHRIS	233.11
2	MARK	298.12
3	SARAH	301.21
4	PAT	4009.21
5	JOHN	678.43
6	WILLIAM	3231.75
7	ANDREW	1762.11
8	BENJAMIN	201.11
9	JANET	98.11
10	STEVE	6153.32
11	JENNIFER	542.11
12	JOY	2442.22
13	MARY	5691.78
14	TOM	5669.12
15	BETH	4822.12

As of January 2006

Notes:

- SAS may split the header at a blank or an upper/lower case change.

Vertical Headers



HEADING=V will rotate headers to save horizontal space.

```
proc print data=softsale(obs=10) heading=v;
  title 'Softco Inc. Sales Summary';
  footnote 'As of January 2006';
run;
```

Notes:

- PROC PRINT may automatically rotate if it helps.

The Resulting Output



Softco Inc. Sales Summary						
		D i v i s i o n	Y e a r s	S a l e s	E x p e n s e	S t a t e
1	CHRIS	H	2	233.11	94.12	WI
2	MARK	H	5	298.12	52.65	WI
3	SARAH	S	6	301.21	65.17	MN
4	PAT	H	4	4009.21	322.12	IL
5	JOHN	H	7	678.43	150.11	WI
6	WILLIAM	H	11	3231.75	644.55	MN
7	ANDREW	S	24	1762.11	476.13	MN
8	BENJAMIN	S	3	201.11	25.21	IL
9	JANET	S	1	98.11	125.32	WI
10	STEVE	H	21	6153.32	1507.12	WI

As of January 2006

Forcing Horizontal Headers



HEADING=H will always use horizontal labels.

```
proc print data=softsale heading=h;  
  title 'Softco Inc. Sales Summary';  
  footnote 'As of January 2006';  
run;
```

PROC PRINT with a BY Statement



PROC PRINT can print control breaks.

```
proc sort data=softsale;  
  by division;  
run;  
  
proc print data=softsale;  
  title 'Softco Sales Summary';  
  by division;  
run;
```

PROC PRINT with a BY Statement (continued)



Softco Sales Summary					
----- Division=H -----					
Obs	Name	Years	Sales	Expense	State
1	CHRIS	2	233.11	94.12	WI
2	MARK	5	298.12	52.65	WI
3	PAT	4	4009.21	322.12	IL
4	JOHN	7	678.43	150.11	WI
5	WILLIAM	11	3231.75	644.55	MN
6	STEVE	21	6153.32	1507.12	WI
7	BETH	12	4822.12	982.10	WI
----- Division=S -----					
Obs	Name	Years	Sales	Expense	State
8	SARAH	6	301.21	65.17	MN
9	ANDREW	24	1762.11	476.13	MN
10	BENJAMIN	3	201.11	25.21	IL
11	JANET	1	98.11	125.32	WI
12	JENNIFER	1	542.11	134.24	IL
13	JOY	12	2442.22	761.98	WI
14	MARY	14	5691.78	2452.11	WI
15	TOM	5	5669.12	798.15	MN

PRINT with BY and ID



When ID and BY specify the same variable, an alternate format is used.

```
proc sort data=softsale;
  by division;
run;

proc print data=softsale;
  title 'Softco Sales Summary';
  id division;
  by division;
run;
```

PRINT with BY and ID (continued)



Softco Sales Summary					
Division	Name	Years	Sales	Expense	State
H	CHRIS	2	233.11	94.12	WI
	MARK	5	298.12	52.65	WI
	PAT	4	4009.21	322.12	IL
	JOHN	7	678.43	150.11	WI
	WILLIAM	11	3231.75	644.55	MN
	STEVE	21	6153.32	1507.12	WI
	BETH	12	4822.12	982.10	WI
S	SARAH	6	301.21	65.17	MN
	ANDREW	24	1762.11	476.13	MN
	BENJAMIN	3	201.11	25.21	IL
	JANET	1	98.11	125.32	WI
	JENNIFER	1	542.11	134.24	IL
	JOY	12	2442.22	761.98	WI
	MARY	14	5691.78	2452.11	WI
	TOM	5	5669.12	798.15	MN

Summing Numeric Columns



SUM gives subtotals and grand totals for variables listed.

```
proc sort data=softsale;
  by division state;
run;

proc print data=softsale;
  title 'Softco Sales Summary';
  id division state;
  by division state;
  sum sales expense;
run;
```


A PROC FREQ Example



Which columns below would be candidates for FREQ counts?

Softsale						
Obs	Name	Division	Years	Sales	Expense	State
1	CHRIS	H	2	233.11	94.12	WI
2	MARK	H	5	298.12	52.65	WI
3	SARAH	S	6	301.21	65.17	MN
4	PAT	H	4	4009.21	322.12	IL
5	JOHN	H	7	678.43	150.11	WI
6	WILLIAM	H	11	3231.75	644.55	MN
7	ANDREW	S	24	1762.11	476.13	MN
8	BENJAMIN	S	3	201.11	25.21	IL
9	JANET	S	1	98.11	125.32	WI
10	STEVE	H	21	6153.32	1507.12	WI
11	JENNIFER	S	1	542.11	134.24	IL
12	JOY	S	12	2442.22	761.98	WI
13	MARY	S	14	5691.78	2452.11	WI
14	TOM	S	5	5669.12	798.15	MN
15	BETH	H	12	4822.12	982.10	WI

A PROC FREQ Example (continued)



We might want to know:

- How many rows for each value of Division?
- How many rows for each value of Years?
- How many rows for each value of State?

Notes:

- Percentages may also be desired.
- We probably would not count identifiers (Name), or continuous variables (Sales, Expense) without some kind of summarizing.

A One-Way Table



TABLE State will give counts and percentages for State.

```
proc freq data=softsale;
  title 'Softco State Distributions';
  table state;
run;
```

Softco State Distributions					
The FREQ Procedure					
State	Frequency	Percent	Cumulative Frequency	Cumulative Percent	
IL	3	20.00	3	20.00	
MN	4	26.67	7	46.67	
WI	8	53.33	15	100.00	

Select Some Rows



Use WHERE to select only Hardware employees.

```
proc freq data=softsale;
  title 'Softco Hardware State Distributions';
  table state;
  where division='H';
run;
```

Softco Hardware State Distributions					
The FREQ Procedure					
State	Frequency	Percent	Cumulative Frequency	Cumulative Percent	
IL	1	14.29	1	14.29	
MN	1	14.29	2	28.57	
WI	5	71.43	7	100.00	

Two One-way Tables



Two separate one-way tables are produced.
NOCUM eliminates cumulative statistics.

```
proc freq data=softsale;  
  title 'Selected Frequencies';  
  table state division / nocum;  
run;
```

The Resulting Output



Two tables and no cumulative statistics.

Selected Frequencies		
The FREQ Procedure		
State	Frequency	Percent
IL	3	20.00
MN	4	26.67
WI	8	53.33

Division	Frequency	Percent
H	7	46.67
S	8	53.33

Multiple Tables



You can also have multiple TABLES statements in a single PROC FREQ.

Example:

```
proc freq data=softsale;  
  title 'PROC FREQ with 2 TABLES Statements';  
  tables state / nocum;  
  tables division / nopercent;  
run;
```

The Resulting Output



PROC FREQ with 2 TABLES Statements		
The FREQ Procedure		
State	Frequency	Percent
IL	3	20.00
MN	4	26.67
WI	8	53.33

Division	Frequency	Cumulative Frequency
H	7	7
S	8	15

Two-way Tables



Specify two variables to see combinations between them.

```
proc freq data=softsale;  
  title 'A Two Way Table';  
  table division * state;  
run;
```

The Resulting Output



Specify two variables to see combinations between them.

A Two Way Table				
The FREQ Procedure				
Table of Division by State				
Division	State			
Frequency	IL	MN	WI	Total
Percent				
Row Pct				
Col Pct				
H	1	1	5	7
	6.67	6.67	33.33	46.67
	14.29	14.29	71.43	
	33.33	25.00	62.50	
S	2	3	3	8
	13.33	20.00	20.00	53.33
	25.00	37.50	37.50	
	66.67	75.00	37.50	
Total	3	4	8	15
	20.00	26.67	53.33	100.00

Suppressing Unwanted Options



Leave out row, column, and cell percentages.

```
proc freq data=softsale;
  title 'Two Way Table With Options';
  table division * state/nocol norow nopercnt;
run;
```

Two Way Table With Options

The FREQ Procedure

Table of Division by State

Division	State			
Frequency	IL	MN	WI	Total
H	1	1	5	7
S	2	3	3	8
Total	3	4	8	15

Introduction

64

The LIST Option



Prints in a list rather than cross-tabulation format.

```
proc freq data=softsale;
  title 'Two Way Table With The List Option';
  table division * state/list;
run;
```

Two Way Table With The List Option

The FREQ Procedure

Division	State	Frequency	Percent	Cumulative Frequency	Cumulative Percent
H	IL	1	6.67	1	6.67
H	MN	1	6.67	2	13.33
H	WI	5	33.33	7	46.67
S	IL	2	13.33	9	60.00
S	MN	3	20.00	12	80.00
S	WI	3	20.00	15	100.00

Introduction

65

The OUT= Option



The OUT= option stores results in a SAS dataset instead of printing.

```
proc freq data=softsale;
  table division * state/out=freqds noprint;
run;
proc print data=freqds;
  title 'PROC FREQ Output Dataset';
run;
```

PROC FREQ Output Dataset					
Obs	Division	State	COUNT	PERCENT	
1	H	IL	1	6.6667	
2	H	MN	1	6.6667	
3	H	WI	5	33.3333	
4	S	IL	2	13.3333	
5	S	MN	3	20.0000	
6	S	WI	3	20.0000	

Notes:

- Coding NOPRINT on the table statement eliminates the default PROC FREQ report.

Enhancing The Report



PROC PRINT can use optional statements to enhance reports.

```
proc print data=freqds label;
  title 'PROC FREQ Output Dataset';
  by division;
  id division;
  label count='Number of Employees';
  sum count percent;
run;
```

Default PROC FREQ Output



When you do not specify TABLES, you get one-way frequencies for ALL values of all variables.

```
proc freq data=softsale;  
  title 'PROC FREQ with No TABLE Statements';  
run;
```

Notes:

- This can be a very expensive program for large datasets with many variables or variables with many values.

Partial Output



PROC FREQ with No TABLE Statements				
The FREQ Procedure				
Name	Frequency	Percent	Cumulative Frequency	Cumulative Percent
ANDREW	1	6.67	1	6.67
BENJAMIN	1	6.67	2	13.33
BETH	1	6.67	3	20.00
CHRIS	1	6.67	4	26.67
JANET	1	6.67	5	33.33
JENNIFER	1	6.67	6	40.00
JOHN	1	6.67	7	46.67
JOY	1	6.67	8	53.33
MARK	1	6.67	9	60.00
MARY	1	6.67	10	66.67
PAT	1	6.67	11	73.33
SARAH	1	6.67	12	80.00
STEVE	1	6.67	13	86.67
TOM	1	6.67	14	93.33
WILLIAM	1	6.67	15	100.00

Continued Output



Division	Frequency	Percent	Cumulative Frequency	Cumulative Percent
H	7	46.67	7	46.67
S	8	53.33	15	100.00

Notes:

- There is much more output.

Other Possible PROC FREQ Applications



Any time a count or percentage is needed:

- Validate dates on a dataset
- Show distribution of customers, products
- Help in testing and validation.

PROC MEANS/SUMMARY



PROC MEANS and PROC SUMMARY condense and summarize SAS datasets.

PROC SUMMARY and PROC MEANS Features:

- compute selected univariate statistics
- CLASS and BY variables define subgroups
- CLASS does not require sorting
- SUMMARY gives summary statistics in the output dataset (default)
- SUMMARY allows the option to print the output
- MEANS defaults to producing a report
- MEANS can also produce an output dataset

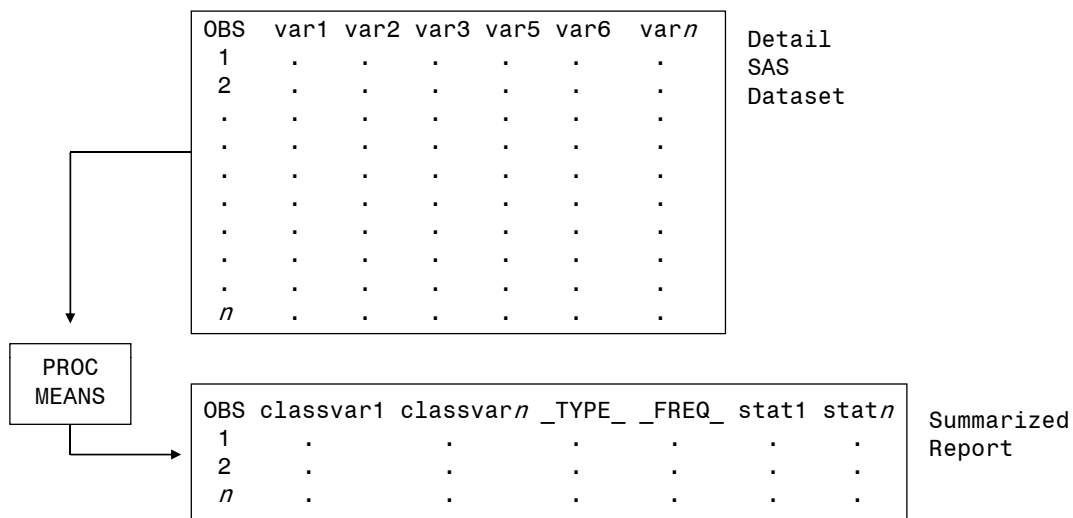
Notes:

- MEANS usually does a report; SUMMARY usually produces a dataset.
- All other options and statements are virtually the same.

PROC MEANS



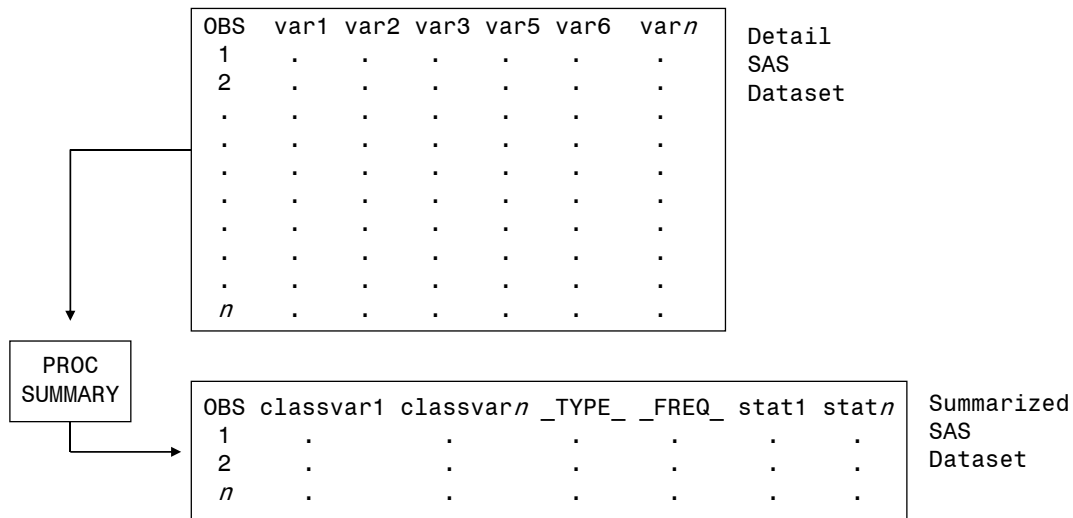
Diagram of PROC MEANS:



PROC SUMMARY



Diagram of PROC SUMMARY:



PROC MEANS / SUMMARY Syntax



Syntax:

PROC MEANS | SUMMARY DATA=SASdataset options;

Options: (partial)

- NOPRINT** suppress report (PROC SUMMARY default)
- MAXDEC=n** limit number of decimals for statistics
- N** calculate number of non-missing values
- NMISS** calculate number of missing values
- NWAY** calculate only the highest level of interaction
- MEAN** calculate mean
- MAX** calculate highest value
- MIN** calculate lowest value
- RANGE** compute difference between min and max

PROC MEANS / SUMMARY Syntax (continued)



SUM	sum variables
VAR	compute variance
STD	compute standard deviation
MISSING	treat missing values as a valid subgroup value
IDMIN	ID variable should use the lowest value
DESCENDING	arrange the lowest summary levels first
PRINT	produce a printed report (PROC MEANS default)
ORDER =	specify the sort order of the CLASS variables

Optional statements (partial list):

BY *variable(s)*;
CLASS *variable(s)*;
VAR *variable(s)*;
OUTPUT OUT= *SASdataset options*;
WHERE *where condition*;

PROC MEANS Applications



PROC MEANS produces simple univariate statistics.

PROC MEANS features:

- default statistics for all numeric variables
- default report
- output dataset is optional
- BY or CLASS statement gives subgroup statistics

PROC MEANS Example



Default: Calculate statistics on all numeric variables in Softsale.

```
proc means data=softsale;  
  title 'PROC MEANS Defaults';  
run;
```

PROC MEANS Defaults					
The MEANS Procedure					
Variable	N	Mean	Std Dev	Minimum	Maximum
Years	15	8.5333333	7.0494850	1.0000000	24.0000000
Sales	15	2408.92	2318.15	98.1100000	6153.32
Expense	15	572.7386667	674.2543676	25.2100000	2452.11

A Selective PROC MEANS Example



Use PROC options to specify desired statistics and the VAR statement to select variables to analyze.

```
proc means data=softsale min max mean;  
  title '3 Statistics on Sales and Expense';  
  var sales expense;  
run;
```

3 Statistics on Sales and Expense			
The MEANS Procedure			
Variable	Minimum	Maximum	Mean
Sales	98.1100000	6153.32	2408.92
Expense	25.2100000	2452.11	572.7386667

PROC SUMMARY On the Same Data



PROC SUMMARY with the PRINT option gives virtually identical results.

```
proc summary data=softsale min max mean print;  
  var sales expense;  
  title 'PROC SUMMARY Statistics';  
run;
```

PROC SUMMARY Statistics			
The SUMMARY Procedure			
Variable	Minimum	Maximum	Mean
Sales	98.110000	6153.32	2408.92
Expense	25.210000	2452.11	572.7386667

The CLASS Statement



CLASS produces separate statistics for each division;
MAXDEC limits statistics to 3 decimals.

```
proc means data=softsale min max mean maxdec=3;  
  class division;  
  var sales expense;  
  title 'CLASS and MAXDEC';  
run;
```

CLASS and MAXDEC					
The MEANS Procedure					
Division	N Obs	Variable	Minimum	Maximum	Mean
H	7	Sales	233.110	6153.320	2775.151
		Expense	52.650	1507.120	536.110
S	8	Sales	98.110	5691.780	2088.471
		Expense	25.210	2452.110	604.789

Creating a Dataset with MEANS Output



The OUTPUT statement creates a dataset containing selected statistics.

```
proc means data=softsale noprint;
  class division;
  var sales expense;
  output out=meansout
         mean(sales)=Meansale
         mean(expense)=Avgexp
         n(sales)=Count;
run;

proc print data=meansout;
  title 'Statistics Stored in a Dataset';
run;
```

Notes:

- NOPRINT tells PROC MEANS to skip printing.

Creating a Dataset with MEANS Output (continued)



Obs	Division	_TYPE_	_FREQ_	Meansale	Avgexp	Count
1		0	15	2408.92	572.739	15
2	H	1	7	2775.15	536.110	7
3	S	1	8	2088.47	604.789	8

Notes:

- _TYPE_ = 0 represents the grand total.
- _TYPE_ = 1 represents each division.

PROC MEANS with NWAY



NWAY limits output to the most detailed level of interaction.

```
proc means data=softsale noprint nway;  
  class division;  
  var sales expense;  
  output out=meansout mean=Meansale Avgexp n=Count;  
run;  
proc print data=meansout;  
  title 'Statistics Stored in a Dataset with NWAY';  
run;
```

Notes:

- This is an alternate syntax for the OUTPUT statement; the first variable following the equal sign is given the statistic for the first variable on the VAR statement. The second variable is given the statistic for the second variable on the VAR statement, etc.

PROC MEANS with NWAY (continued)



Statistics Stored in a Dataset with NWAY						
Obs	Division	_TYPE_	_FREQ_	Meansale	Avgexp	Count
1	H	1	7	2775.15	536.110	7
2	S	1	8	2088.47	604.789	8

Notes:

- Compare the NWAY output with the output in the previous example.

PROC MEANS with BY



Sorting and the BY statement can also be used with MEANS and SUMMARY.

```
proc sort data=softsale;
  by division;
run;
```

```
proc means data=softsale noprint;
  by division;
  var sales expense;
  output out=meansout mean=Meansale Avgexp n=Count;
run;
```

```
proc print data=meansout;
  title 'Means With the BY Statement';
run;
```

PROC MEANS with the BY Statement



Means With the BY Statement						
Obs	Division	_TYPE_	_FREQ_	Meansale	Avgexp	Count
1	H	0	7	2775.15	536.110	7
2	S	0	8	2088.47	604.789	8

Notes:

- Results are almost identical to CLASS and NWAY.
- All records are now _TYPE_=0.
- Because BY may require sorting, CLASS is almost always better.

Creating a Dataset with No Specified Statistics



If no statistics are specified on the OUTPUT statement, defaults are used.

```
proc means data=softsale noprint;
  class division;
  var sales expense;
  output out=meansout;
run;

proc print data=meansout;
  title 'Output Dataset With Default Statistics';
run;
```

The Resulting Output



Output Dataset With Default Statistics						
Obs	Division	_TYPE_	_FREQ_	_STAT_	Sales	Expense
1		0	15	N	15.00	15.00
2		0	15	MIN	98.11	25.21
3		0	15	MAX	6153.32	2452.11
4		0	15	MEAN	2408.92	572.74
5		0	15	STD	2318.15	674.25
6	H	1	7	N	7.00	7.00
7	H	1	7	MIN	233.11	52.65
8	H	1	7	MAX	6153.32	1507.12
9	H	1	7	MEAN	2775.15	536.11
10	H	1	7	STD	2391.91	544.27
11	S	1	8	N	8.00	8.00
12	S	1	8	MIN	98.11	25.21
13	S	1	8	MAX	5691.78	2452.11
14	S	1	8	MEAN	2088.47	604.79
15	S	1	8	STD	2364.79	807.97

Notes:

- Statistics must be given on Output statement (Not PROC Statement) or defaults will be used.

A PROC SUMMARY Problem



Every time First Department Store makes a sale, they record Store, Department, Amount, and Coupon in a SAS dataset INVNTORY.

First Department Store				
OBS	Store	Dept	Amount	Coupon
1	101	SHOES	14.22	.
2	101	SHOES	22.85	.
3	101	SPORTS	17.11	0.99
4	102	SHOES	26.78	4.99
5	101	SPORTS	11.97	.
6	102	SHOES	54.22	12.99
7	102	SPORTS	11.16	.
8	103	SHOES	41.22	.
9	103	SPORTS	13.78	2.89
10	102	SHOES	13.72	4.49
11	101	SHOES	21.75	.
12	101	SPORTS	12.71	.
13	102	SHOES	21.78	1.88
14	103	SPORTS	51.57	6.99
15	101	SHOES	54.25	.
16	102	SPORTS	11.56	0.49
17	101	SHOES	41.52	.
18	101	SPORTS	15.79	.

A PROC SUMMARY Problem (continued)



Question: If you worked for First Department Stores, what could you do with the data from the previous slide?

Or another way of asking it might be:

1. How do we classify the data?
2. Which statistics do we want?

A PROC SUMMARY Solution



PROC SUMMARY is fast and does not require sorting.

```
proc summary data=inventory;      /* input dataset */
  class store dept;              /* class vars */
  var amount coupon;            /* analysis vars */
  output out=sumds              /* summary ds */
        n(amount)=Salesnum      /* sums => salesnum */
        mean(amount)=Avgsale    /* ave to avg sale */
        sum(amount)=Totsale     /* sum in totsales */
        mean(coupon)=Meancoup  /* ave coupon */
        sum(coupon)=Sumcoup;   /* sum of coupon */
run;

proc print data=sumds;
  title 'First Department Store Sales Summary';
run;
```

A PROC SUMMARY Solution (continued)



FIRST DEPARTMENT STORE SALES SUMMARY									
OBS	Store	Dept	_TYPE_	_FREQ_	Salesnum	Avgsale	Totsales	Meancoup	Sumcoup
1			0	18	18	25.4422	457.96	4.46375	35.71
2		SHOES	1	10	10	31.2310	312.31	6.08750	24.35
3		SPORTS	1	8	8	18.2062	145.65	2.84000	11.36
4	101		2	9	9	23.5744	212.17	0.99000	0.99
5	102		2	6	6	23.2033	139.22	4.96800	24.84
6	103		2	3	3	35.5233	106.57	4.94000	9.88
7	101	SHOES	3	5	5	30.9180	154.59	.	.
8	101	SPORTS	3	4	4	14.3950	57.58	0.99000	0.99
9	102	SHOES	3	4	4	29.1250	116.50	6.08750	24.35
10	102	SPORTS	3	2	2	11.3600	22.72	0.49000	0.49
11	103	SHOES	3	1	1	41.2200	41.22	.	.
12	103	SPORTS	3	2	2	32.6750	65.35	4.94000	9.88

Other Applications



Other applications appropriate for PROC MEANS/SUMMARY might be:

- any system needing to reduce detail
- testing and validation while testing
- summarizing using different class levels

Formatting SAS Data Values



FORMATS change the way data values are printed and can make output values more presentable.

- FORMATS can be specified in the DATA step.
- FORMATS can be overridden in the PROC step.
- You can use a SAS-written format.
- You can create user-written formats.

Syntax:

FORMAT *variables format . . . ;*

Notes:

- LABEL, introduced earlier, changes the way variable names display.

What is a FORMAT?

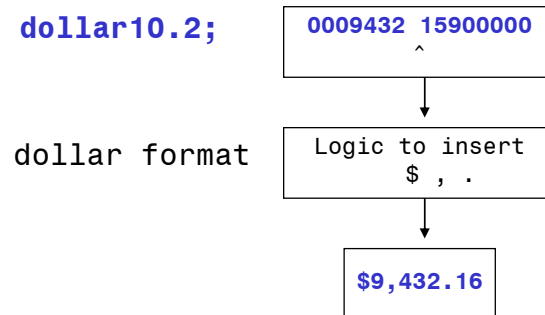


A FORMAT is a routine that all variables pass through when output.

Example:

1. Sales has a value of 9432.159
2. Print Sales with dollar signs and commas.
3. Allow a total of 10 digits (including \$, .).
4. Print two digits after the decimal point.

```
format sales dollar10.2;
```



Notes:

- If you don't specify a format, SAS chooses one.

Introduction

96

SAS Provided Formats



There are many ways to print data values.

<i>w.</i>	standard numeric
<i>w.d</i>	numeric with decimal
BEST <i>w.</i>	SAS chooses best notation
COMMA <i>w.d</i>	commas in numbers
DOLLAR <i>w.d</i>	insert dollar sign, commas
E <i>w.</i>	scientific notation
FRACT <i>w.</i>	fractions
HEX <i>w.</i>	numeric hexadecimal
IB <i>w.d</i>	integer binary
PD <i>w.d</i>	packed decimal
ROMAN <i>w.</i>	Roman numerals
SSN <i>w.</i>	Social Security Numbers
WORDF <i>w.</i>	words with fractions
WORDS <i>w.</i>	numbers as words

Introduction

97

SAS Provided Formats (continued)



Z <i>w.d</i>	print leading zeros
ZD <i>w.d</i>	zoned decimal
\$ <i>w.</i>	standard character
\$CHAR <i>w.</i>	character with leading blanks
\$HEX <i>w.</i>	character hexadecimal
\$VARYING <i>w.</i>	variable length character

Notes:

- *w.* values specify the width to allow for output.
- For numerics, if you don't give a format, SAS uses BEST*w.* format.

An Example Using SAS Provided FORMATS



Formats can be coded in the DATA step.

```
data softsale;
  infile rawin;
  input @1 Name      $10.
        @12 Division $1.
        @15 Years    2.
        @19 Sales    7.2
        @28 Expense  7.2
        @36 State    $2.;
  format name $4. sales comma9.2 expense comma9.2;
run;
proc print data=softsale;
  title 'Proc Print with formats';
run;
```

CHRIS	H	2	233.11	94.12	WI
MARK	H	5	298.12	52.65	WI
SARAH	S	6	301.21	65.17	MN
...					
BETH	H	12	4822.12	982.10	WI

The Resulting Output



Proc Print with formats

Obs	Name	Division	Years	Sales	Expense	State
1	CHRI	H	2	233.11	94.12	WI
2	MARK	H	5	298.12	52.65	WI
3	SARA	S	6	301.21	65.17	MN
4	PAT	H	4	4,009.21	322.12	IL
5	JOHN	H	7	678.43	150.11	WI
6	WILL	H	11	3,231.75	644.55	MN
7	ANDR	S	24	1,762.11	476.13	MN
8	BENJ	S	3	201.11	25.21	IL
9	JANE	S	1	98.11	125.32	WI
10	STEV	H	21	6,153.32	1,507.12	WI
11	JENN	S	1	542.11	134.24	IL
12	JOY	S	12	2,442.22	761.98	WI
13	MARY	S	14	5,691.78	2,452.11	WI
14	TOM	S	5	5,669.12	798.15	MN
15	BETH	H	12	4,822.12	982.10	WI

Notes:

- These formats are stored in the data descriptor.
- All later steps will use these formats unless overridden.

Another FORMAT Example



Formats can also be coded in the PROC step.

```
data softsale;
  infile rawin;
  input @1 Name $10.
        @12 Division $1.
        @15 Years 2.
        @19 Sales 7.2
        @28 Expense 7.2
        @36 State $2.;
run;
proc print data=softsale;
  title 'Proc Print with formats';
  format name $4. sales comma9.2 expense comma9.2;
run;
```

CHRIS	H	2	233.11	94.12	WI
MARK	H	5	298.12	52.65	WI
SARAH	S	6	301.21	65.17	MN
...					
BETH	H	12	4822.12	982.10	WI

The Resulting Output



Proc Print with formats

Obs	Name	Division	Years	Sales	Expense	State
1	CHRI	H	2	233.11	94.12	WI
2	MARK	H	5	298.12	52.65	WI
3	SARA	S	6	301.21	65.17	MN
4	PAT	H	4	4,009.21	322.12	IL
5	JOHN	H	7	678.43	150.11	WI
6	WILL	H	11	3,231.75	644.55	MN
7	ANDR	S	24	1,762.11	476.13	MN
8	BENJ	S	3	201.11	25.21	IL
9	JANE	S	1	98.11	125.32	WI
10	STEV	H	21	6,153.32	1,507.12	WI
11	JENN	S	1	542.11	134.24	IL
12	JOY	S	12	2,442.22	761.98	WI
13	MARY	S	14	5,691.78	2,452.11	WI
14	TOM	S	5	5,669.12	798.15	MN
15	BETH	H	12	4,822.12	982.10	WI

Notes:

- These FORMATS are used in this step only.

Introduction

102

Creating Your Own FORMATS



PROC FORMAT creates picture and value labeling FORMATS.

Syntax:

PROC FORMAT options;

VALUE *name(options)*
 range1='label1'
 range2='label2'
 ...;

PICTURE *name(options)*
 range1='picture1'(options)
 range2='picture2'(options)
 ...;

Introduction

103

Creating Your Own FORMATS (continued)



FORMAT names:

- are 1-32 characters long (8 characters in V8 and earlier)
- must start with a letter or underscore
- must begin with \$ for character variables
- cannot end with a number
- do not end in period when created
- DO end in period when used

A PROC FORMAT Example



Recode the RESPONSE; display PHONE in a more familiar format.

fileref	1 6085552424
rawin	2 3123432424

```
proc format ;  
  value respfmt 1='agree' 2='disagree';  
  picture phonefmt low-high='999/999-9999';  
run;
```

```
data phonelst;  
  infile rawin;  
  input Response Phone;  
run;
```

```
proc print data=phonelst;  
  format response respfmt. phone phonefmt. ;  
  title 'Proc print with user formats';  
run;
```

The Resulting Output



Proc print with user formats		
Obs	Response	Phone
1	agree	608/555-2424
2	disagree	312/343-2424

Notes:

- FORMATS don't change the values, they just change the way the values are printed.

The VALUE Statement



Print labels corresponding to a range instead of an actual value.

Ranges can be a single value:

```
value respfmt  
  1='agree'  
  2='disagree';
```

Ranges can specify a range of values:

```
value gradefmt  
  low-69 ='FAILING'  
  70-high='PASSING';
```

Ranges can specify a string of values:

```
value qltyfmt  
  1,10-20,30-high ='ACCEPT'  
  other='REJECT';
```

The VALUE Statement (continued)



Ranges can specify a combination:

```
value gradefmt
  1-69 = 'FAILING'
  70-high = 'PASSING'
  0 = 'INCOMPLETE';
```

Character fields require a "\$" format:

```
value $STATFMT
  'AZ' = 'ARIZONA'
  'TX' = 'TEXAS';
```

```
value $divfmt
  'H' = 'HARDWARE'
  'S' = 'SOFTWARE'
  other = 'INVALID';
```

A PROC FORMAT Example



Create some formats for SOFTCO.

```
                                fileref
                                rawin
                                CHRIS      H      2      233.11    94.12 WI
                                MARK       H      5      298.12    52.65 WI
                                SARAH      S      6      301.21    65.17 MN
                                ...
                                BETH      H     12    4822.12   982.10 WI

data softsale;
  infile rawin;
  input @1 Name      $10.
        @12 Division $1.
        @15 Years    2.
        @19 Sales    7.2
        @28 Expense  7.2
        @36 State    $2.;

run;
```

A PROC FORMAT Example (continued)



```
proc format;
  value $divfmt
    'H' = 'HARDWARE'
    'S' = 'SOFTWARE'
    other = 'INVALID';
  value yearfmt
    low-<5 = 'Under 5'
    5-10   = '5-10'
    10<-high = 'Over 10';
run;

proc print data=softsale;
  format division $divfmt.  years yearfmt.;
  title 'Softco with value label formats';
run;
```

The Resulting Output



Softco with value label formats						
Obs	Name	Division	Years	Sales	Expense	State
1	CHRIS	HARDWARE	Under 5	233.11	94.12	WI
2	MARK	HARDWARE	5-10	298.12	52.65	WI
3	SARAH	SOFTWARE	5-10	301.21	65.17	MN
4	PAT	HARDWARE	Under 5	4009.21	322.12	IL
5	JOHN	HARDWARE	5-10	678.43	150.11	WI
6	WILLIAM	HARDWARE	Over 10	3231.75	644.55	MN
7	ANDREW	SOFTWARE	Over 10	1762.11	476.13	MN
8	BENJAMIN	SOFTWARE	Under 5	201.11	25.21	IL
9	JANET	SOFTWARE	Under 5	98.11	125.32	WI
10	STEVE	HARDWARE	Over 10	6153.32	1507.12	WI
11	JENNIFER	SOFTWARE	Under 5	542.11	134.24	IL
12	JOY	SOFTWARE	Over 10	2442.22	761.98	WI
13	MARY	SOFTWARE	Over 10	5691.78	2452.11	WI
14	TOM	SOFTWARE	5-10	5669.12	798.15	MN
15	BETH	HARDWARE	Over 10	4822.12	982.10	WI

A PROC FREQ Job Without Formatting



TABLE YEARS produces detailed counts.

```
proc freq data=softsale;
  table years;
run;
```

Years	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	2	13.33	2	13.33
2	1	6.67	3	20.00
3	1	6.67	4	26.67
4	1	6.67	5	33.33
5	2	13.33	7	46.67
6	1	6.67	8	53.33
7	1	6.67	9	60.00
11	1	6.67	10	66.67
12	2	13.33	12	80.00
14	1	6.67	13	86.67
21	1	6.67	14	93.33
24	1	6.67	15	100.00

PROC FREQ With Formatting



A FORMAT causes PROC FREQ to group table values.

```
proc format;
  value yearfmt low-<5 = 'Under 5'
                5-10 = '5-10'
                10<-high = 'Over 10';
run;

proc freq data=softsale;
  table years;
  format years yearfmt.;
run;
```

Years	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Under 5	5	33.33	5	33.33
5-10	4	26.67	9	60.00
Over 10	6	40.00	15	100.00

Formats From a SAS Dataset



Hardcoding recoded values can be inconvenient if:

- There are many lines to type.
- They may already exist in another file.
- They change often.

Is there some way to read the format recoding from a file rather than typing the values?

The CNTLIN Option



PROC FORMAT can read the values from a SAS dataset with special variables.

Syntax:

```
PROC FORMAT CNTLIN=sasdataset;  
RUN;
```

The CNTLIN Dataset



The CNTLIN dataset must contain the following variables:

Fmtname	Name of the format	Ex. \$DIVFMT
Start	Starting value for range	Ex. 'H'
Label	Recoded value to use	Ex. 'Hardware'

OPTIONAL variables include:

End	Ending value for range	Ex. 'I'
-----	------------------------	---------

A CNTLIN Example



We can build a format from the following flat file.

```
data formds;
infile dfile;
  input @1 Start $1.
        @4 Label $12.;
  retain Fmtname '$divfmt';
run;
```

fileref
dfile

H	Hardware
P	Publications
S	Software

```
proc print data=formds;
  title 'Format Dataset';
run;
```

```
proc format cntlin=formds;
run;
```

Partial Log and Output Files



```
proc format cntlin=formds;
NOTE: Format $DIVFMT has been output.
run;
NOTE: PROCEDURE FORMAT used:
      real time          0.04 seconds

NOTE: There were 3 observations read from the data set WORK.FORMDS.
```

Format Dataset				
Obs	Start	Label	Fmtname	
1	H	Hardware	\$divfmt	
2	P	Publications	\$divfmt	
3	S	Software	\$divfmt	

Using the Format



Read, format, and print the data as usual.

```
data softsale;
  infile rawin;
  input @1 Name $10.
        @12 Division $1.
        @15 Years 2.
        @19 Sales 7.2
        @28 Expense 7.2
        @36 State $2.;

run;

proc print data=softsale;
  Title 'Print With Formats';
  format division $divfmt. ;
run;
```

The Resulting Output



Print With Formats						
Obs	Name	Division	Years	Sales	Expense	State
1	CHRIS	Hardware	2	233.11	94.12	WI
2	MARK	Hardware	5	298.12	52.65	WI
3	SARAH	Software	6	301.21	65.17	MN
4	PAT	Hardware	4	4009.21	322.12	IL
5	JOHN	Hardware	7	678.43	150.11	WI
6	WILLIAM	Hardware	11	3231.75	644.55	MN
7	ANDREW	Software	24	1762.11	476.13	MN
8	BENJAMIN	Software	3	201.11	25.21	IL
9	JANET	Software	1	98.11	125.32	WI
10	STEVE	Hardware	21	6153.32	1507.12	WI
11	JENNIFER	Software	1	542.11	134.24	IL
12	JOY	Software	12	2442.22	761.98	WI
13	MARY	Software	14	5691.78	2452.11	WI
14	TOM	Software	5	5669.12	798.15	MN
15	BETH	Hardware	12	4822.12	982.10	WI

The PICTURE Statement (optional)



The PICTURE statement creates display layouts for numeric variables.

- Pictures are for NUMERIC variables only.
- Pictures are limited to 24 or less positions.
- Pictures are specified with 3 types of characters.

Types of picture characters:

- 0** print digit; suppress leading zeros
- 1-9** print digit; print leading zeros
- All others** print that character unchanged

The PICTURE Statement (optional)



Examples:

```
proc format;
  picture negfmt    low-<0 = '00,000.99-'
                   0-high = '00,009.99+' ;
  picture phonefmt  other  = '999-999-9999' ;
run;
```

Notes:

- The first picture character must be numeric.
- FILL and PREFIX allow non-numeric characters first.

A Dataset Needing Formatting (optional)



Phone numbers, balances, and past-due values.

```
data phonelst;
  infile rawin;
  input Phonenum Balance Pastdue;
run;

proc print data=phonelst;
  title 'Phone List Without Formatting';
run;
```

filedef	5552226721	33.22	2216.00
rawin	6083442121	16.11	10.00
	3412552765	-15.6	-5.20

Phone List Without Formatting				
	OBS	Phonenum	Balance	Pastdue
	1	5552226721	33.22	2216.00
	2	6083442121	16.11	10.00
	3	3412552765	-15.60	-5.20

Format the Values (optional)



Picture formats make the values more presentable.

```
data phonelst;
  infile rawin;
  input Phonenum Balance Pastdue;
run;

proc format ;
  picture negfmt    low-<0 = '00,000.99- '
                    0-high = '00,009.99+' ;

  picture phonefmt other  = '999-999-9999' ;
run;

proc print data=phonelst;
  format phonenum phonefmt.;
  format balance pastdue negfmt.;
  title 'Phone List with Formatting';
run;
```

filedef	5552226721	33.22	2216.00
rawin	6083442121	16.11	10.00
	3412552765	-15.6	-5.20

The Resulting Output (optional)



Phone List with Formatting			
OBS	Phonenum	Balance	Pastdue
1	555-222-6721	33.22+	2,216.00+
2	608-344-2121	16.11+	10.00+
3	341-255-2765	15.60-	5.20-

Summary



- SAS is a large, comprehensive system
- The Base product has wonderful design and tools
- And provides a solid beginning.

Contact Us



SYSTEMS SEMINAR CONSULTANTS, INC.

SAS® Training, Consulting, & Help Desk Services

2997 Yarmouth Greenway Drive • Madison, WI 53711

(608) 278-9964 • Fax (608) 278-0065

www.sys-seminar.com



Steven First

President

sfirst@sys-seminar.com

