SAS Views®: SAS® Basics

SAS Institute Inc.
SAS Circle □ Box 8000
Cary, NC 27512-8000

Property of
Chris Bray

SAS Views®: SAS® Basics

SAS Institute Inc.
SAS Circle □ Box 8000
Cary, NC 27512-8000

SAS Views®: SAS® Basics

SAS Institute Inc.
SAS Circle □ Box 8000
Cary, NC 27512-8000

SAS Views®: SAS® Basics was written by **Larry Stewart** and edited by **Marla Z. Hudnell**. Associate editors were **B. Claire McCullough** and **Maureen Fitzgerald**.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. *SAS Views®: SAS® Basics*. Cary, NC: SAS Institute Inc., 1986. 664 pp.

SAS Views®: SAS® Basics

ISBN 1-55544-004-5

87 86 4 3

Base SAS® software, the foundation of the SAS System, provides data retrieval and management, programming, statistical, and reporting capabilities. Also in the SAS System are SAS/GRAPH®, SAS/FSP®, SAS/ETS®, SAS/IMS-DL/I®, SAS/OR®, SAS/AF®, SAS/REPLAY-CICS®, SAS/IML™, SAS/DMI™, SAS/RTERM™, SAS/STAT™, SAS/QC™, and SAS/SHARE™ software. These products and SYSTEM 2000® Data Base Management System, including the basic SYSTEM 2000®, QueX™, Multi-User™, Screen Writer™, CREATE™, DISKMGR™, and CICS interface products, are available from SAS Institute Inc., a private company devoted to the support and further development of the software and related services. *SAS Communications®*, *SAS Training®*, *SAS Views®*, and SASware Ballot® are published by SAS Institute Inc.

Contents

Preface

SAS Views: SAS Basics serves as course notes for the SAS Basics for Mainframes and SAS Basics for Minicomputers courses and is a review text for beginning SAS users.

The SAS Basics course is offered as a three-day public course in two formats: lecture and lecture/workshop. Lecture courses are taught in hotels around the United States. Lecture/workshop courses are taught at SAS Institute training centers.

The standard public course can be altered for a two- or three-day on-site presentation, with or without a workshop, at customer sites. The topics presented at an on-site course are determined by the site sponsoring the course and the Education Division of SAS Institute Inc.

Topics and exercises that are not presented in the course should be studied independently by the user. Solutions to all exercises presented in this course appear in Appendix B.

Prerequisites

The SAS Basics course is **not** an introductory data processing course. It is designed for people who have a data processing background, but little or no SAS programming experience. Specifically, the prerequisites for the course include:

- a minimum of three months experience in data processing, including writing programs in some language
- the ability to submit batch jobs or execute jobs interactively under your operating system
- knowledge of the system commands needed to read from or write to external tape or disk files.

Course Objectives

After completion of this course, you should:

Chapter 1. Introduction

- be familiar with the capabilities of base SAS software
- be aware of SAS software products, services, and literature

Chapter 2. Fundamental Concepts

- know SAS syntax and other SAS rules
- understand the structure of SAS data sets

Chapter 3. Reading and Processing Raw Data

- know how to create SAS data sets
- be able to read raw data
- understand the internal flow of the SAS DATA step

Chapter 4. Listing and Sorting Data

- know how to invoke SAS procedures
- be able to place titles on SAS output
- be familiar with output from the PRINT procedure
- know how to sort SAS data sets
- be able to process data in groups (BY statement)

Chapter 5. Data Transformations

- know how to create variables
- be familiar with SAS functions
- understand how to conditionally execute SAS statements
- understand how the lengths of character variables are defined.

Course Objectives

After completion of this course, you should:

Chapter 6. Errors and Missing Values in SAS Jobs

- be able to locate and understand SAS syntax error messages
- be able to locate and understand SAS data error messages
- understand how SAS software handles missing data

Chapter 7. Summarizing Data

- know how to produce various descriptive statistics with the MEANS procedure
- know how to obtain frequency distributions and crosstabulations with the FREQ procedure

Chapter 8. Reading and Writing SAS Data Sets

- know how to select a subset of observations
- understand how to control when an observation is written to a SAS data set
- be able to read an existing SAS data set
- understand how to control which variables are read from a SAS data set
- understand how to control which variables are written to a SAS data set

Chapter 9. Formatting SAS Output

- be able to assign descriptive labels to variables
- know how to assign specific output formats to variables
- be able to create user-defined formats.

Course Objectives

After completion of this course, you should:

Chapter 10. Processing Date and Time Values

- understand SAS date and time values
- be familiar with SAS input and output formats for date and time data
- be familiar with SAS functions for date and time data

Chapter 11. Printer Graphics

- know how to invoke the PLOT procedure
- be familiar with PLOT procedure options
- know how to invoke the CHART procedure
- understand how to specify the physical form of a chart
- be familiar with CHART procedure options

Chapter 12. Introduction to SAS Data Set Management

- understand how to combine data from several SAS data sets
- understand the function of the BY statement in SAS data set management

Chapter 13. Reading and Writing Raw Data

- understand the basic method of pointing to external files
- understand how the SAS System links to external files
- be able to read external files.

Course Objectives

After completion of this course, you should:

Chapter 14. Permanent SAS Data Sets

- understand how to permanently store SAS data sets
- understand how to access permanently stored SAS data sets

Chapter 15. Introduction to Report Writing

- be able to produce simple reports with the PRINT procedure
- know how to use PUT and FILE statements to produce customized reports
- realize the potential of the SAS report writing system.

1. Introduction

1.1 The SAS System

1.2 The DATA Step

1.3 The PROC Step

1.1 The SAS System

What Is the SAS System?

The SAS System is a family of software products that provide a variety of data processing and analysis capabilities including

- data retrieval, report writing, and statistical analysis
- information processing
- color graphics
- graphics replay under CICS
- planning and forecasting
- project management
- applications development
- DL/I interface
- interactive matrix manipulations
- data base management.

What Is the SAS System?

The SAS System also includes

training
: instructor-based, video-based, and computer-based courses

documentation
: user's guides, introductory guides, student workbooks, manuals for training courses, technical reports, and companions for specific operating systems

technical support
: specialists for all software product areas and operating systems.

Base SAS Software

Base SAS software, the foundation of the SAS System, is an all-purpose data management, data analysis, and report writing tool featuring a powerful programming language and many ready-to-use procedures.

Specific capabilities include

retrieval
: flexible input techniques

transformations
: a programming language with statistical and mathematical functions

maintenance
: storing, documenting, updating, and editing data sets

manipulation
: sorting, subsetting, concatenating, and merging data sets

report writing
: printing information using program statements

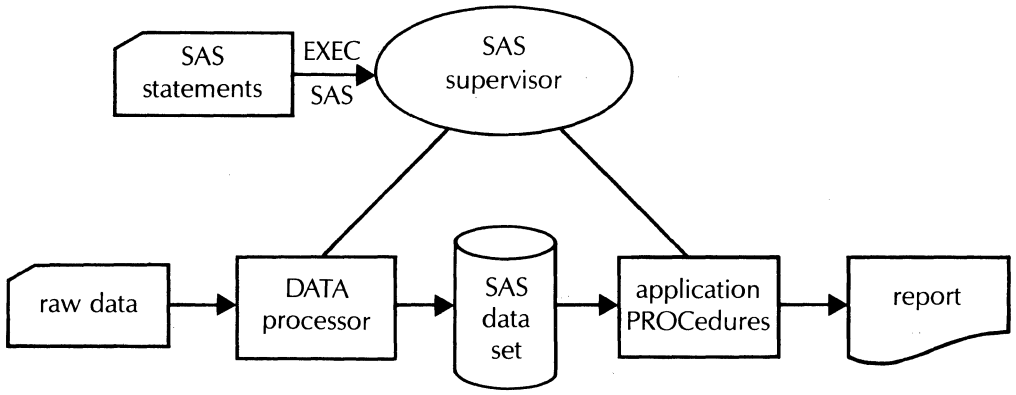
printer graphics
: charts and two-dimensional plots

data reduction and summarization
: descriptive statistics

statistical analysis
: simple crosstabulations to complex multivariate techniques.

SAS Processing

The SAS System consists of a data-handling language and a library of procedures that work together as a system.



SAS Jobs

All SAS jobs are a sequence of SAS steps.

There are only two kinds of SAS steps:

- DATA steps prepare SAS data sets
- PROCedure steps analyze or process SAS data sets.

Example: A company has recorded revenue and expense data yearly from 1982 through 1985.

Compute the difference between expenses and revenue (income) for each year and compute the average income across all the years.

Data Fields →

year			revenue						expenses				
1	2	3	4	5	6	7	8	9	10	11	12	13	14
8	2		4	5	0	0	0		3	2	0	0	0
8	3		5	3	0	0	0		3	6	0	0	0
8	4		5	4	0	0	0		3	8	0	0	0
8	5		6	5	0	0	0		4	3	0	0	0

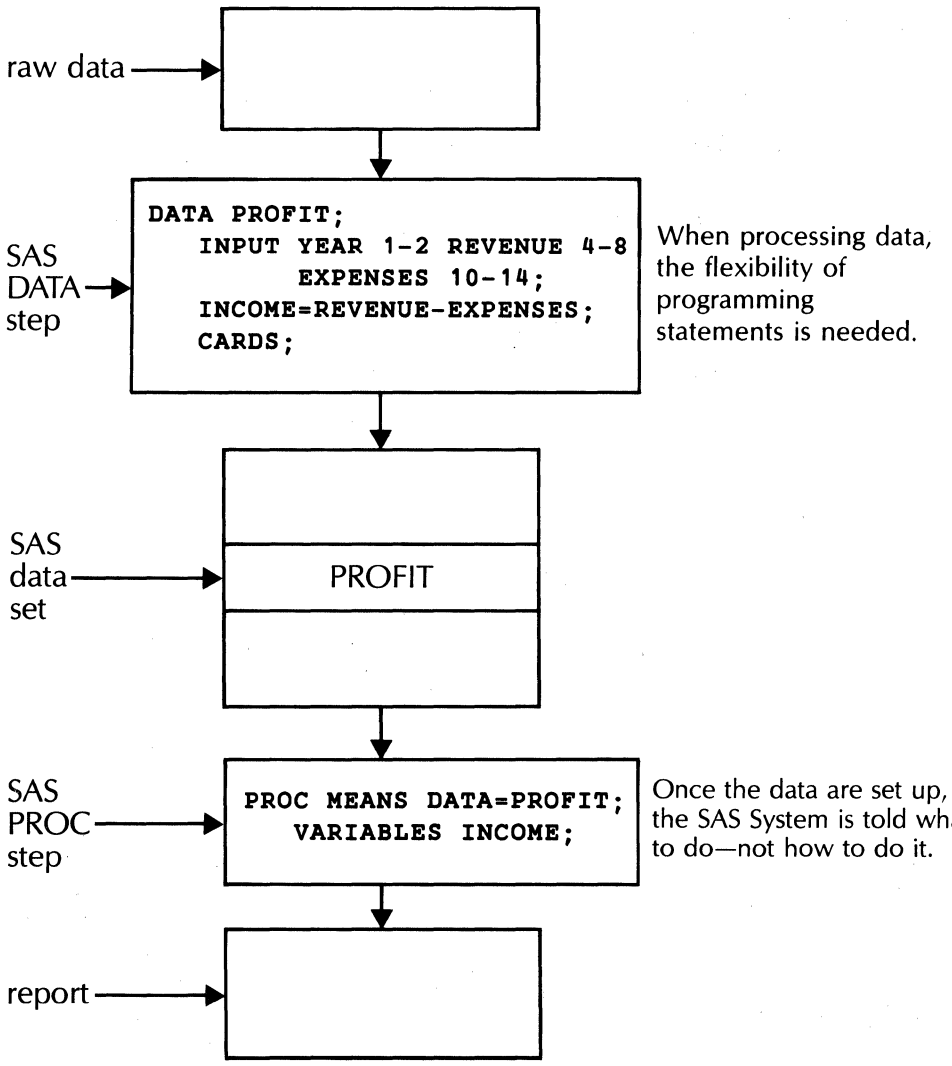
Raw Data →

Variables →

YEAR	REVENUE	EXPENSES	INCOME= REVENUE−EXPENSES
82	45000	32000	13000
83	53000	36000	17000
84	54000	38000	16000
85	65000	43000	22000

SAS Data Set →

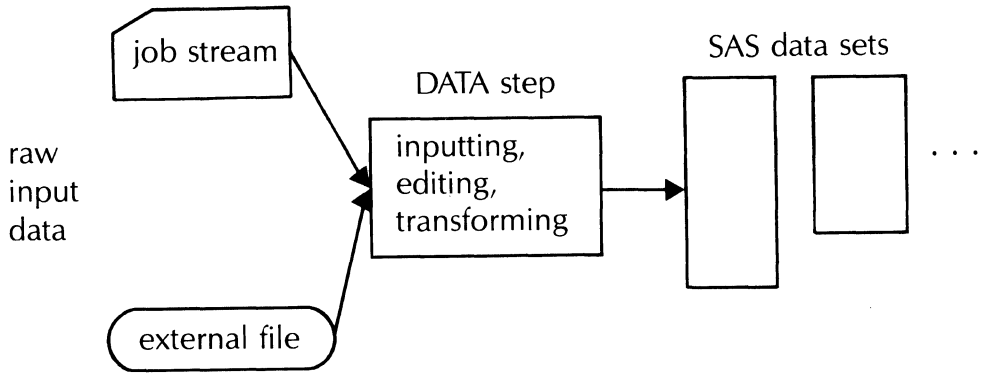
SAS Jobs


raw data
DATA PROFIT;
INPUT YEAR 1-2 REVENUE 4-8
EXPENSES 10-14;
INCOME=REVENUE-EXPENSES;
CARDS;
SAS DATA step
When processing data, the flexibility of programming statements is needed.
PROFIT
SAS data set
PROC MEANS DATA=PROFIT;
VARIABLES INCOME;
SAS PROC step
Once the data are set up, the SAS System is told what to do—not how to do it.
report

1.2 The DATA Step

Overview of the DATA Step

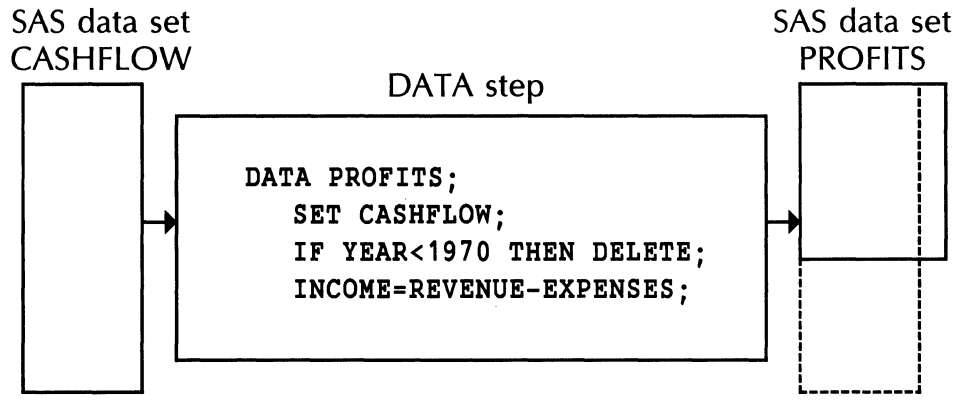
Retrieval



Transferring and Reshaping

Example: SAS data set CASHFLOW contains the variables YEAR, REVENUE, and EXPENSES.

Delete observations for years before 1970 and compute the income for each remaining year.

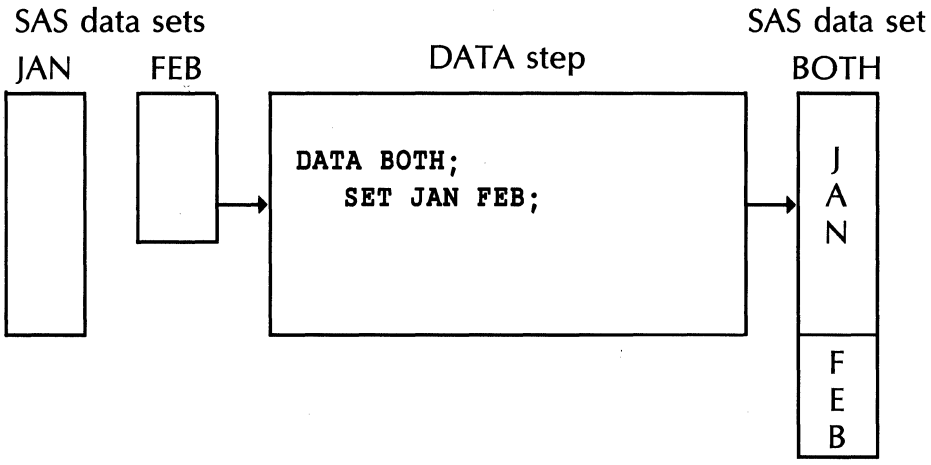


Overview of the DATA Step

Concatenation

Example: SAS data set JAN contains January sales data and SAS data set FEB contains February sales data.

Concatenate the two SAS data sets into one SAS data set that contains the sales data for both months.

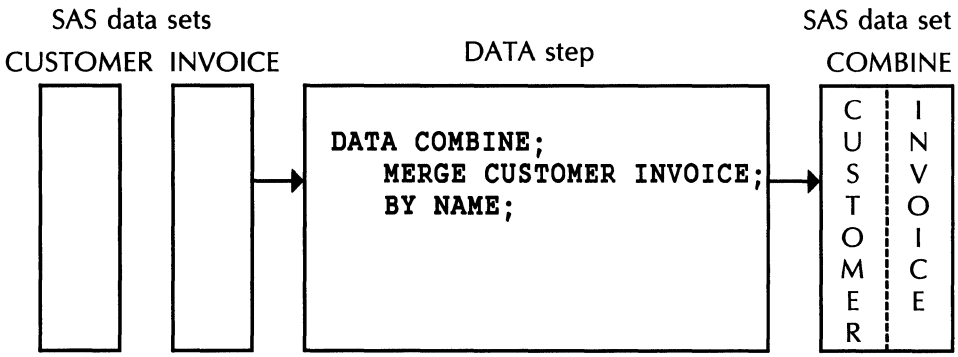


Overview of the DATA Step

Merging

Example: SAS data set CUSTOMER contains names and addresses for customers, and SAS data set INVOICE contains names and invoices for recent sales to the customers.

Merge the data sets so that names and addresses are matched with the appropriate invoices.

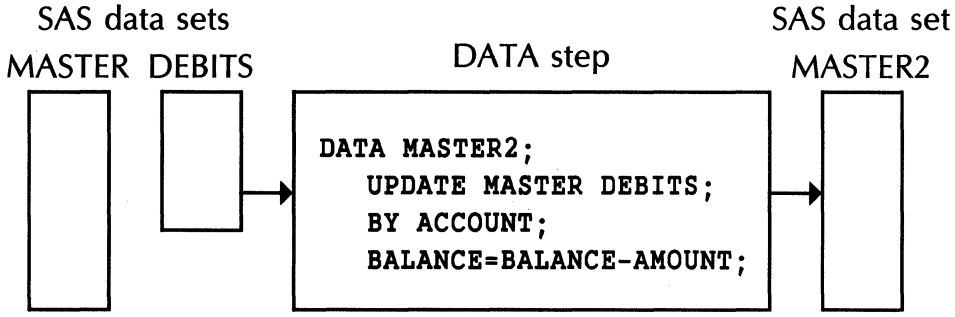


Overview of the DATA Step

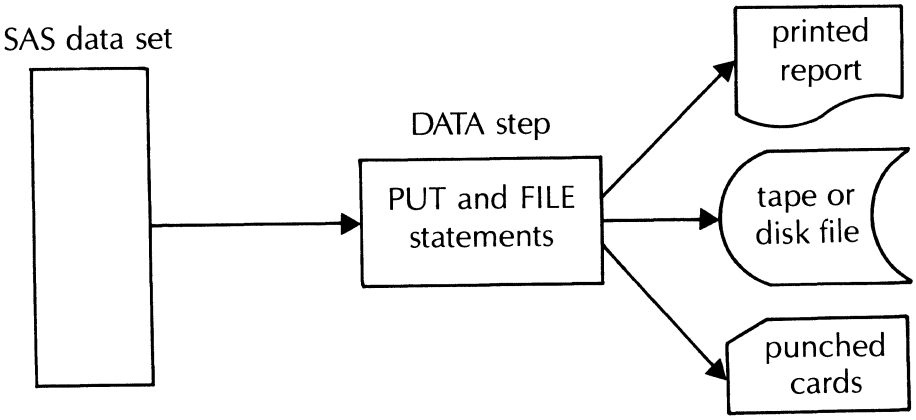
Updating

Example: SAS data set MASTER contains bank account numbers and balances, and SAS data set DEBITS contains account numbers and check amounts.

Update the MASTER data set by subtracting the amounts on the checks from the appropriate accounts.



Report Writing and Creating Files

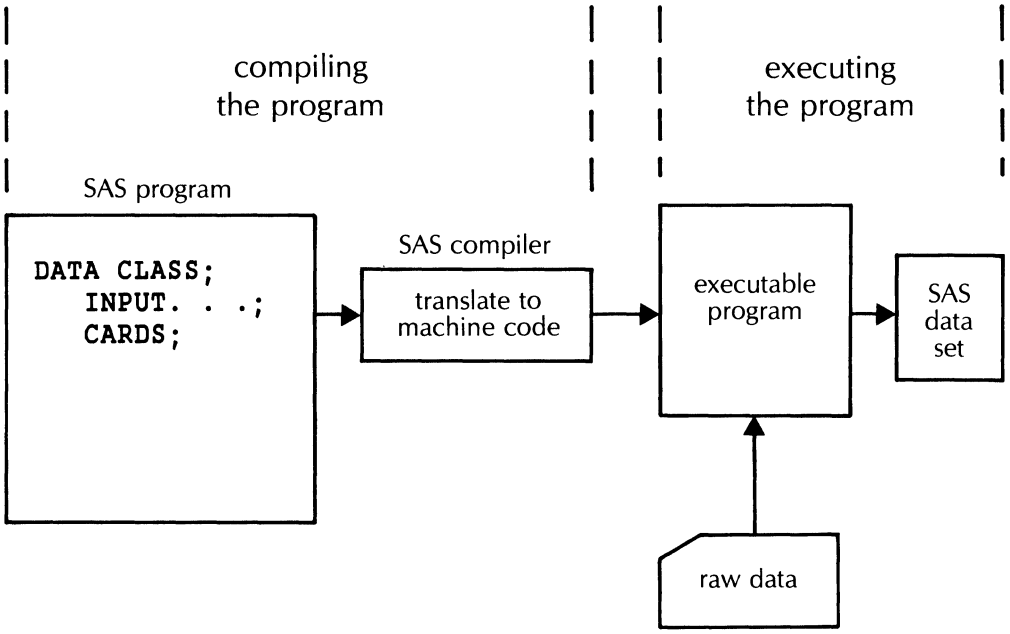


Overview of the DATA Step

The DATA step involves writing a compact SAS program to process data.

The SAS System processes the program in two steps.

- It compiles the program.
- It executes the program.



1.3 The PROC Step

Procedure Libraries

SAS procedures are programs that are designed to perform specific data processing and analysis tasks on SAS data sets.

Base SAS procedures fall into the following categories:

- descriptive statistics
- reporting
- SAS utilities
- OS utilities
- regression analysis
- analysis of variance
- categorical data analysis
- multivariate analysis
- survival analysis
- discriminant analysis
- cluster analysis
- scoring.

2. Fundamental Concepts

2.1 A Simple SAS Job

Supported Environments

The SAS System can run in the following mainframe and minicomputer environments:

Mainframes	**Minicomputers**
OS batch	AOS/VS
TSO (OS interactive)	VMS
CMS	PRIMOS
VSE	
ICCF (VSE interactive)	

The system commands in the various environments differ, but **the SAS statements are almost always the same.**

Note: the SAS System also runs under PC DOS.

Invoking the SAS System

You can invoke the SAS System to execute in

- batch mode
- noninteractive mode
- interactive line mode
- interactive full-screen mode (SAS Display Manager System).

The mode you use will depend upon the operating system at your site, the application you have, and any hardware limitations you have at your site.

Batch and display manager examples are presented in Section 2.2 for mainframe users.

Noninteractive and display manager examples are presented in Section 2.3 for minicomputer users.

Note: see Appendix A in this manual for additional information on invoking the SAS System.

A Problem to Solve

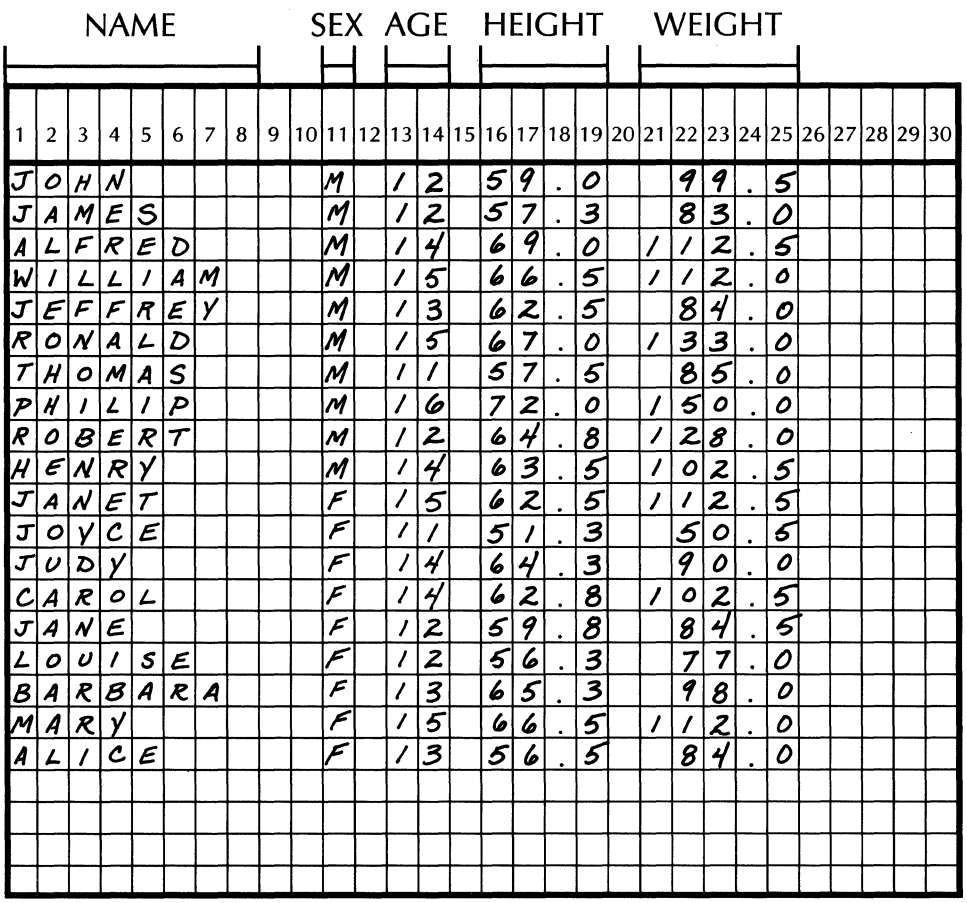
Example: The name, sex, age, height, and weight of each student in a summer camp were recorded and stored with the following record format.

Data Field Description	Field Position
NAME	1-8
SEX	11
AGE	13-14
HEIGHT	16-19
WEIGHT	21-25

Objectives:

1. Obtain a listing of the data.
2. Compute some descriptive statistics for the heights and weights such as the mean, minimum, and maximum.
3. Produce a plot of height versus weight.

A Listing of the Raw Data



Note: these data can be on cards, disk, tape, or lines entered at a terminal.

A SAS Program

```
OPTIONS LS=72 PS=42;
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;
JOHN       M 12 59.0  99.5
JAMES      M 12 57.3  83.0
ALFRED     M 14 69.0 112.5
WILLIAM    M 15 66.5 112.0
JEFFREY    M 13 62.5  84.0
RONALD     M 15 67.0 133.0
THOMAS     M 11 57.5  85.0
PHILIP     M 16 72.0 150.0
ROBERT     M 12 64.8 128.0
HENRY      M 14 63.5 102.5
JANET      F 15 62.5 112.5
JOYCE      F 11 51.3  50.5
JUDY       F 14 64.3  90.0
CAROL      F 14 62.8 102.5
JANE       F 12 59.8  84.5
LOUISE     F 12 56.3  77.0
BARBARA    F 13 65.3  98.0
MARY       F 15 66.5 112.0
ALICE      F 13 56.5  84.0
;
PROC PRINT DATA=CLASS;
PROC MEANS DATA=CLASS;
   VARIABLES HEIGHT WEIGHT;
PROC PLOT DATA=CLASS;
   PLOT WEIGHT*HEIGHT;
```

Note: the LS= and PS= options control the line size and page size for SAS output.

Although they are not always shown, these options are used to control the line size and page size in many examples throughout this course.

2.2 Mainframe Environments

Batch Execution

OS JCL

```
//C02EX1 JOB accounting info, name...
// EXEC SAS
SAS program
```

VSE JCL

```
* $$ JOB JNM=C02EX1,CLASS=2
* $$ LST LST=00E,FNO=STD.,CLASS=A,DISP=D,JSEP=1
* $$ LST LST=01E,FNO=STD.,CLASS=A,DISP=D,JSEP=1
// JOB C02EX1
// LIBDEF CL,SEARCH=(SASCLB,PLICLB),TEMP
// EXEC PROC=$$WRKCKD
// ASSGN SYS011,01E
// ASSGN SYSLST,00E
// ASSGN SYS006,SYSLST
// ASSGN SYSIPT,SYSRDR
// EXEC SASVSE,SIZE=(SASVSE,42K)
SAS program
```

SAS Log: Batch

```
          SAS(R) LOG   OS SAS 5.XX           MVS/XA JOB C02EX1

NOTE: COPYRIGHT (C) 1984 SAS INSTITUTE INC., CARY, N.C.  27511, U.S.A.
NOTE: THE JOB C02EX1 HAS BEEN RUN UNDER RELEASE 5.XX OF SAS AT
SAS INSTITUTE DATA CENTER (00000000).

NOTE: SAS OPTIONS SPECIFIED ARE:
      NODATE,NONUMBER SORT=4

1          OPTIONS LS=72 PS=42;
2          DATA CLASS;
3             INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
4                   HEIGHT 16-19 WEIGHT 21-25;
5             CARDS;

NOTE: DATA SET WORK.CLASS HAS 19 OBSERVATIONS AND 5 VARIABLES. 515 OBS/T
RK
NOTE: THE DATA STATEMENT USED 0.06 SECONDS AND 120K.

25         ;
26         PROC PRINT DATA=CLASS;
NOTE: THE PROCEDURE PRINT USED 0.09 SECONDS AND 308K
      AND PRINTED PAGE 1.

27         PROC MEANS DATA=CLASS;
28            VARIABLES HEIGHT WEIGHT;
NOTE: THE PROCEDURE MEANS USED 0.08 SECONDS AND 328K
      AND PRINTED PAGE 2.

29         PROC PLOT DATA=CLASS;
30            PLOT WEIGHT*HEIGHT;
NOTE: THE PROCEDURE PLOT USED 0.11 SECONDS AND 304K
      AND PRINTED PAGE 3.
NOTE: SAS USED 328K MEMORY.

NOTE: SAS INSTITUTE INC.
      SAS CIRCLE
      PO BOX 8000
      CARY, N.C. 27511-8000
```

PROC PRINT Output: Batch

SAS

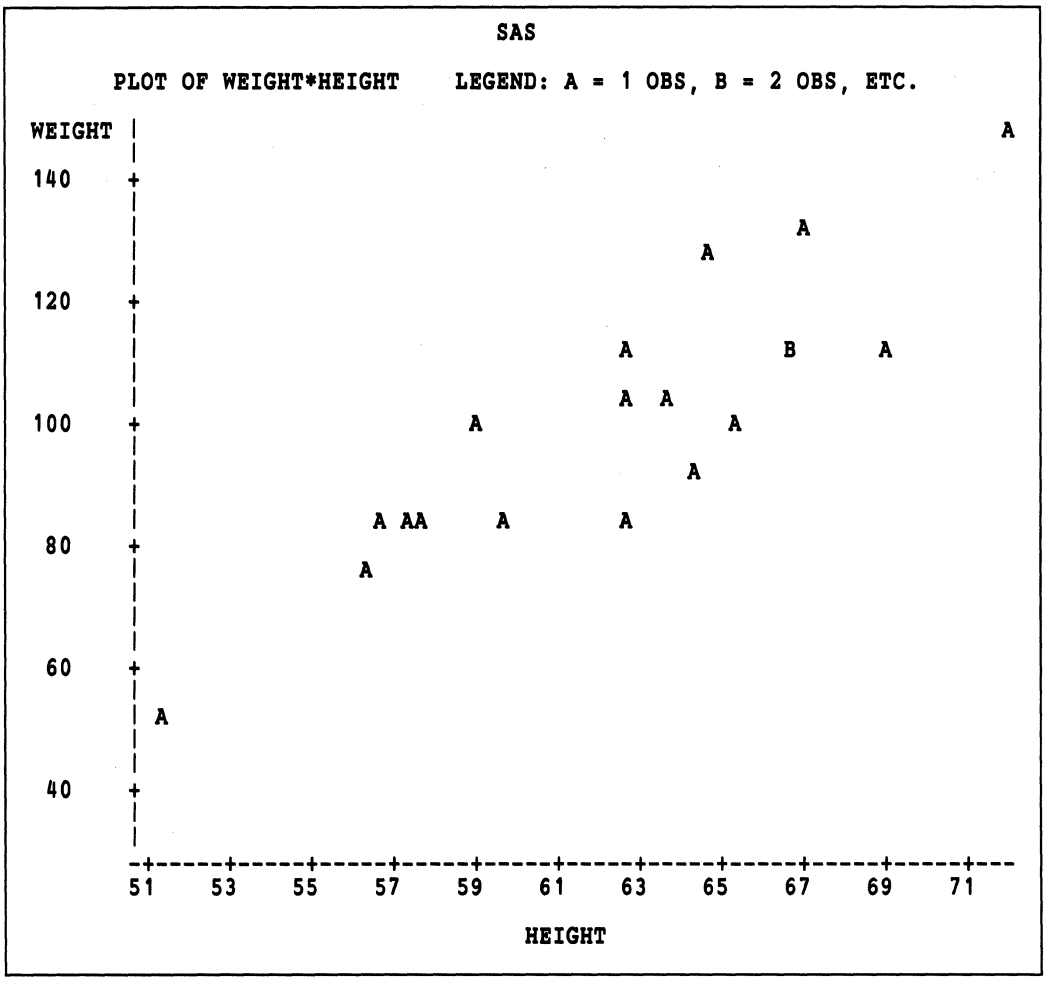
OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0
3	ALFRED	M	14	69.0	112.5
4	WILLIAM	M	15	66.5	112.0
5	JEFFREY	M	13	62.5	84.0
6	RONALD	M	15	67.0	133.0
7	THOMAS	M	11	57.5	85.0
8	PHILIP	M	16	72.0	150.0
9	ROBERT	M	12	64.8	128.0
10	HENRY	M	14	63.5	102.5
11	JANET	F	15	62.5	112.5
12	JOYCE	F	11	51.3	50.5
13	JUDY	F	14	64.3	90.0
14	CAROL	F	14	62.8	102.5
15	JANE	F	12	59.8	84.5
16	LOUISE	F	12	56.3	77.0
17	BARBARA	F	13	65.3	98.0
18	MARY	F	15	66.5	112.0
19	ALICE	F	13	56.5	84.0

PROC MEANS Output: Batch

SAS

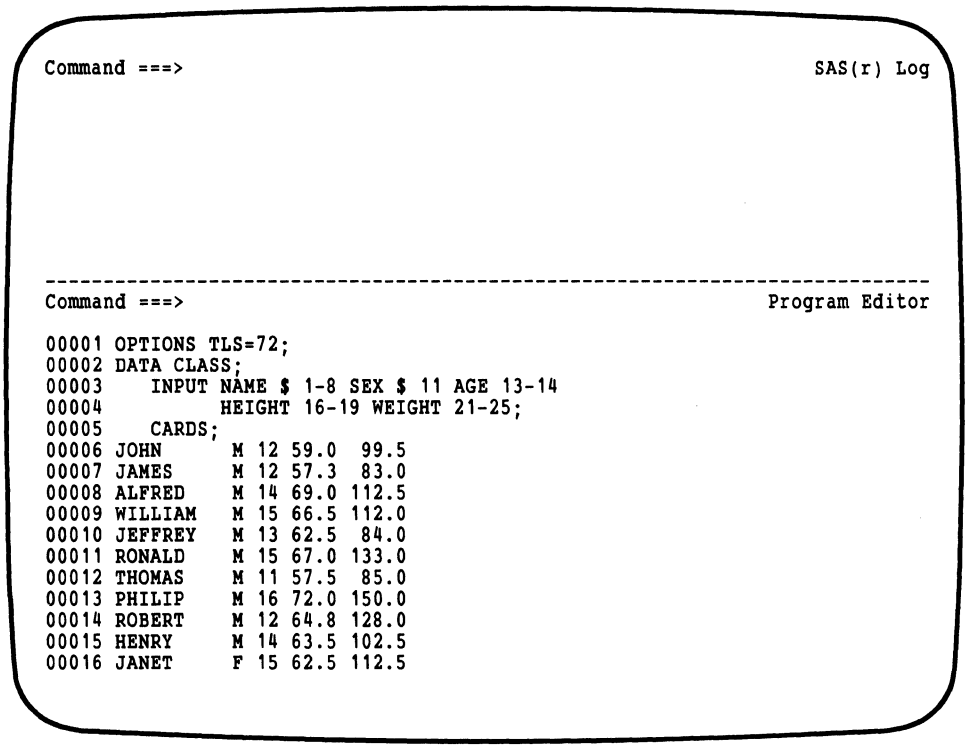
VARIABLE	N	MEAN	STANDARD DEVIATION	MINIMUM VALUE	MAXIMUM VALUE
HEIGHT	19	62.33684211	5.12707525	51.30000000	72.0000000
WEIGHT	19	100.02631579	22.77393349	50.50000000	150.0000000

PROC PLOT Output: Batch



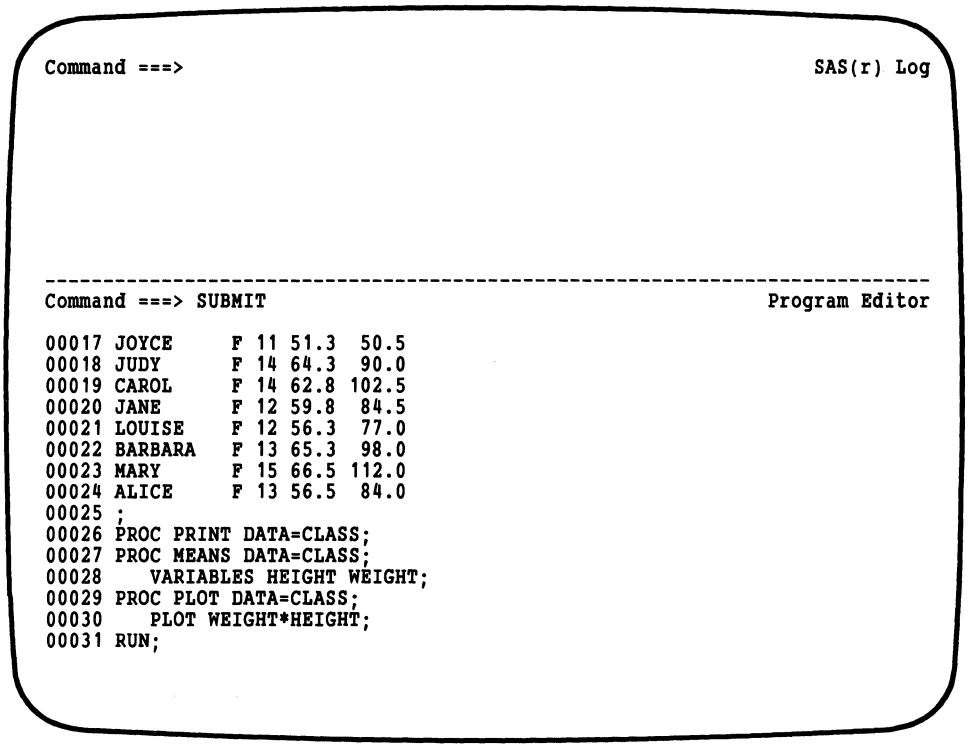
The SAS Display Manager System

Enter your SAS program into the program editor screen.



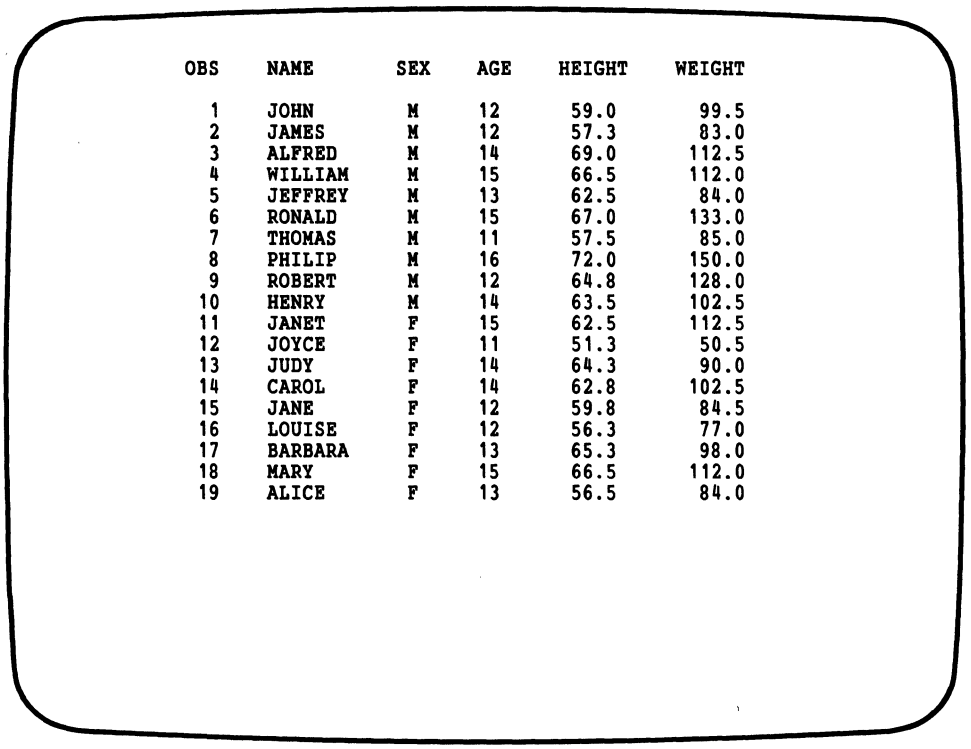
The SAS Display Manager System

Scroll forward to enter the rest of the program and issue the SUBMIT command to execute the program.



The SAS Display Manager System

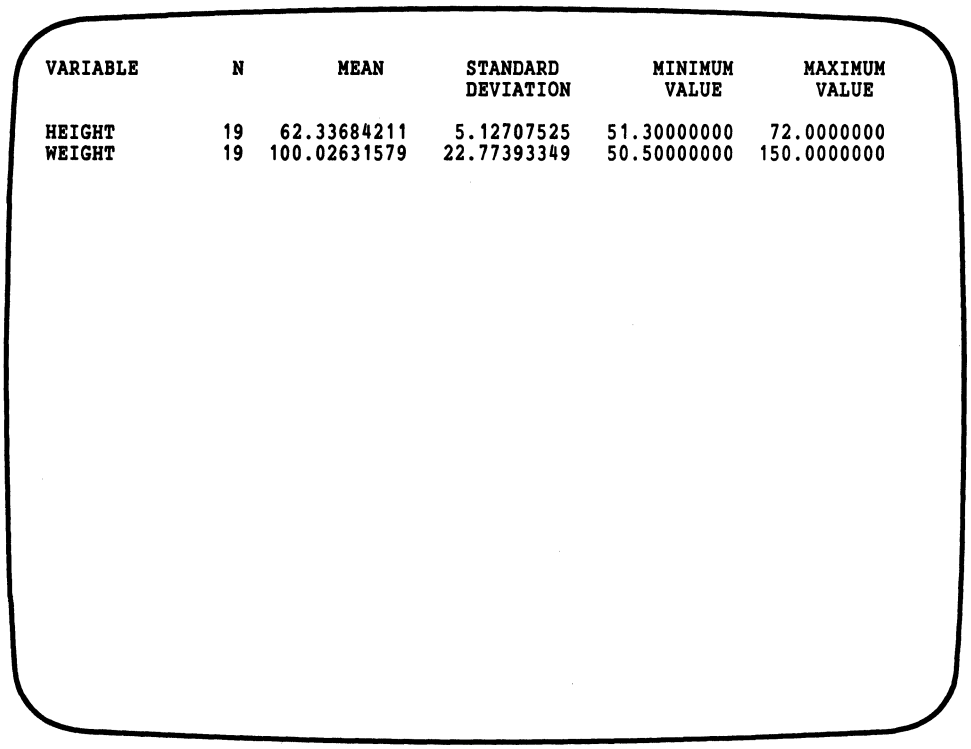
When the first PROC step (PROC PRINT in this example) has finished executing, the results are displayed on the output screen.



OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0
3	ALFRED	M	14	69.0	112.5
4	WILLIAM	M	15	66.5	112.0
5	JEFFREY	M	13	62.5	84.0
6	RONALD	M	15	67.0	133.0
7	THOMAS	M	11	57.5	85.0
8	PHILIP	M	16	72.0	150.0
9	ROBERT	M	12	64.8	128.0
10	HENRY	M	14	63.5	102.5
11	JANET	F	15	62.5	112.5
12	JOYCE	F	11	51.3	50.5
13	JUDY	F	14	64.3	90.0
14	CAROL	F	14	62.8	102.5
15	JANE	F	12	59.8	84.5
16	LOUISE	F	12	56.3	77.0
17	BARBARA	F	13	65.3	98.0
18	MARY	F	15	66.5	112.0
19	ALICE	F	13	56.5	84.0

The SAS Display Manager System

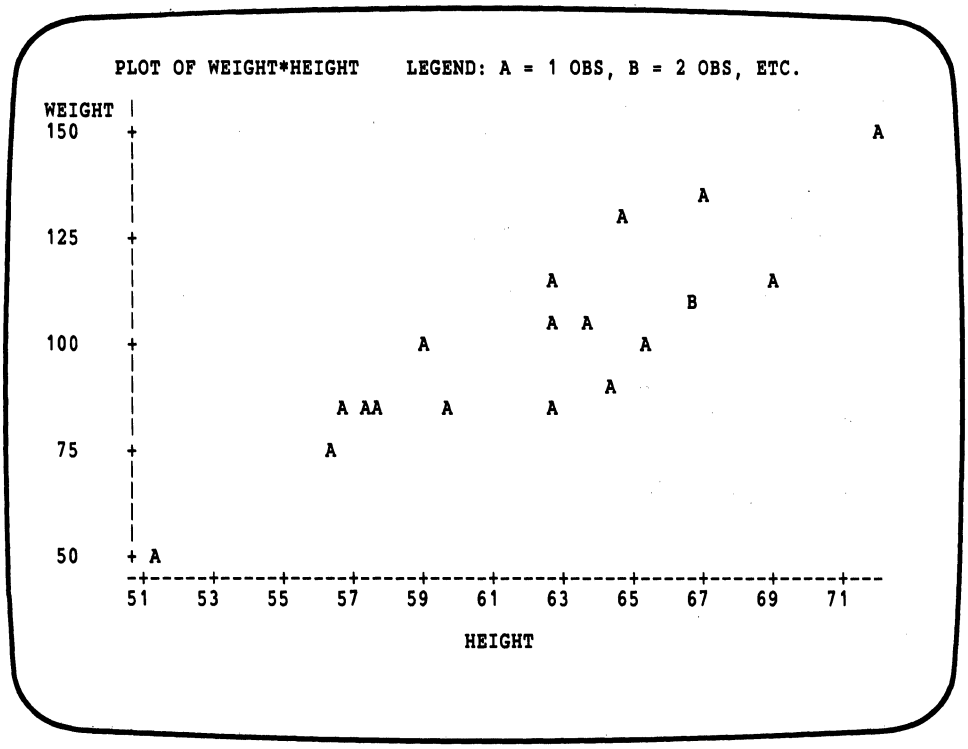
Press the END key to display the output from the second PROC step (PROC MEANS in this example).



VARIABLE	N	MEAN	STANDARD DEVIATION	MINIMUM VALUE	MAXIMUM VALUE
HEIGHT	19	62.33684211	5.12707525	51.30000000	72.0000000
WEIGHT	19	100.02631579	22.77393349	50.50000000	150.0000000

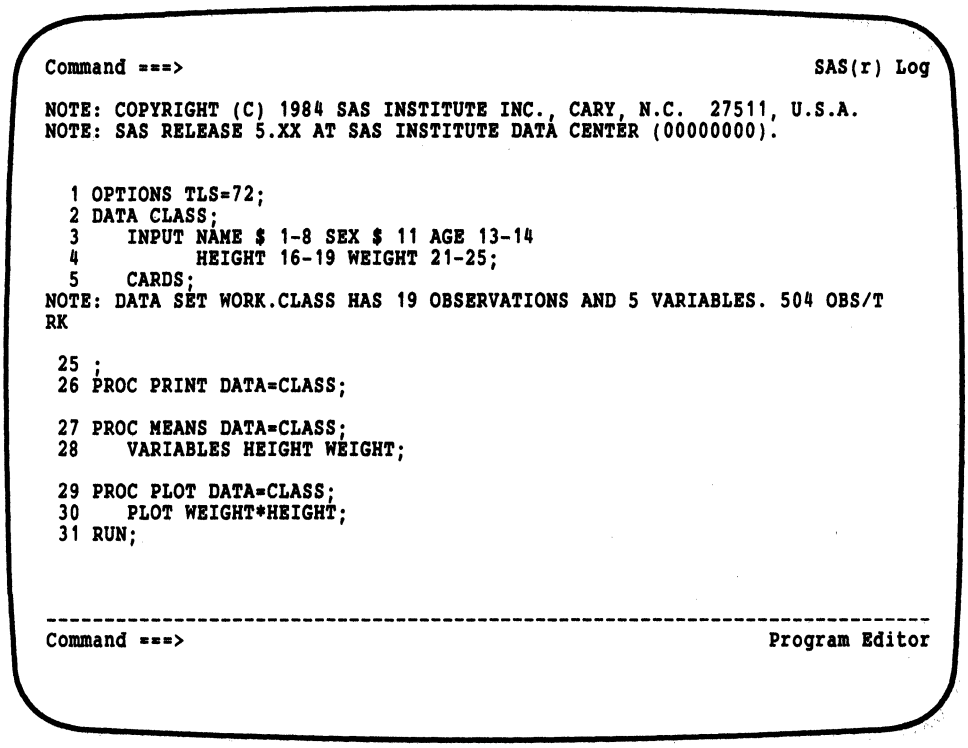
The SAS Display Manager System

Press the END key again to display the output from the third PROC step (PROC PLOT in this example).



The SAS Display Manager System

Press the END key to return to the program editor/log screen. The SAS log is displayed on the log screen.



```
Command ===>                                                    SAS(r) Log

NOTE: COPYRIGHT (C) 1984 SAS INSTITUTE INC., CARY, N.C.  27511, U.S.A.
NOTE: SAS RELEASE 5.XX AT SAS INSTITUTE DATA CENTER (00000000).


  1 OPTIONS TLS=72;
  2 DATA CLASS;
  3    INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
  4          HEIGHT 16-19 WEIGHT 21-25;
  5    CARDS;
NOTE: DATA SET WORK.CLASS HAS 19 OBSERVATIONS AND 5 VARIABLES. 504 OBS/T
RK

 25 ;
 26 PROC PRINT DATA=CLASS;

 27 PROC MEANS DATA=CLASS;
 28    VARIABLES HEIGHT WEIGHT;

 29 PROC PLOT DATA=CLASS;
 30    PLOT WEIGHT*HEIGHT;
 31 RUN;



--------------------------------------------------------------------------------
Command ===>                                                  Program Editor
```

2.3 Minicomputer Environments

Noninteractive Execution

To execute a SAS program in noninteractive mode,

- use an editor in your operating environment to store the SAS statements in a file in your default (or working) directory with a file type (or suffix) of SAS
- specify the filename of the file when you invoke the SAS System.

```
SAS filename
```

The SAS System creates two files in your default directory when the job executes.

filename.LOG	contains the SAS log which is a listing of the SAS statements with informative messages.
filename.LIS	contains output from the SAS procedures.

Note: under the PRIMOS operating environment, the output file has a suffix of LIST, rather than LIS.

Noninteractive Execution

Assume the SAS statements shown earlier are stored in a file in your working directory with the name C02EX1.SAS .

Submit the job by entering:

```
SAS C02EX1
```

The above example would create C02EX1.LOG and C02EX1.LIS (C02EX1.LIST for the PRIMOS operating environment). Use a system command in your operating system to

- produce a hard copy of the content of the files
- display the contents of the files on the terminal.

Note: depending on the operating system, the files generated by the SAS System may have version numbers.

SAS Log: Noninteractive

```
                    S A S   L O G    VMS SAS 5.XX

Copyright (c) 1985 SAS Institute Inc., Cary, N. C. 27511, U. S. A.
NOTE: VMS Version of SAS Release 5.XX at SAS INSTITUTE INC.
NOTE: LICENSED CPUID MODEL = 8600, SERIAL = 00000000.

     1 OPTIONS LS=72 PS=42;
     2 DATA CLASS;
     3    INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
     4          HEIGHT 16-19 WEIGHT 21-25;
     5    CARDS;
NOTE: THE DATA SET WORK.CLASS HAS 19 OBSERVATIONS AND 5 VARIABLES.
NOTE: THE DATA STEP USED 00:00:00.59 CPU SECONDS, 716 PAGEFAULTS
    25 ;
    26 PROC PRINT DATA=CLASS;
NOTE: THE PROCEDURE PRINT USED 00:00:00.24 CPU SECONDS, 265 PAGEFAULTS
NOTE: THE PROCEDURE PRINTED PAGE 1.
    27 PROC MEANS DATA=CLASS;
    28    VARIABLES HEIGHT WEIGHT;
NOTE: THE PROCEDURE MEANS USED 00:00:00.20 CPU SECONDS, 290 PAGEFAULTS
NOTE: THE PROCEDURE PRINTED PAGE 2.
    29 PROC PLOT DATA=CLASS;
    30    PLOT WEIGHT*HEIGHT;
NOTE: THE PROCEDURE PLOT USED 00:00:00.20 CPU SECONDS, 168 PAGEFAULTS
NOTE: THE PROCEDURE PRINTED PAGE 3.
NOTE: SAS INSTITUTE INC., SAS CIRCLE, BOX 8000, CARY, N. C., 27511-8000
```

PROC PRINT Output: Noninteractive

SAS

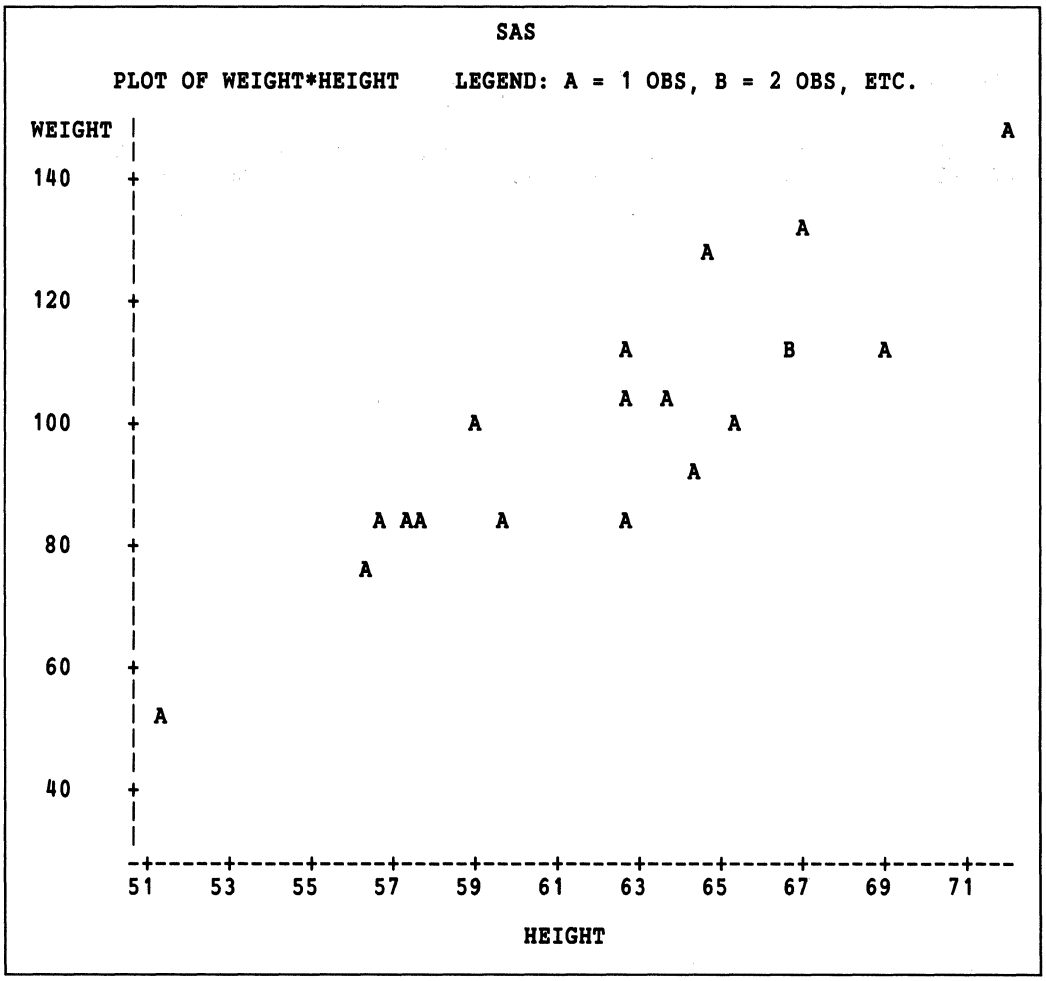
OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0
3	ALFRED	M	14	69.0	112.5
4	WILLIAM	M	15	66.5	112.0
5	JEFFREY	M	13	62.5	84.0
6	RONALD	M	15	67.0	133.0
7	THOMAS	M	11	57.5	85.0
8	PHILIP	M	16	72.0	150.0
9	ROBERT	M	12	64.8	128.0
10	HENRY	M	14	63.5	102.5
11	JANET	F	15	62.5	112.5
12	JOYCE	F	11	51.3	50.5
13	JUDY	F	14	64.3	90.0
14	CAROL	F	14	62.8	102.5
15	JANE	F	12	59.8	84.5
16	LOUISE	F	12	56.3	77.0
17	BARBARA	F	13	65.3	98.0
18	MARY	F	15	66.5	112.0
19	ALICE	F	13	56.5	84.0

PROC MEANS Output: Noninteractive

SAS

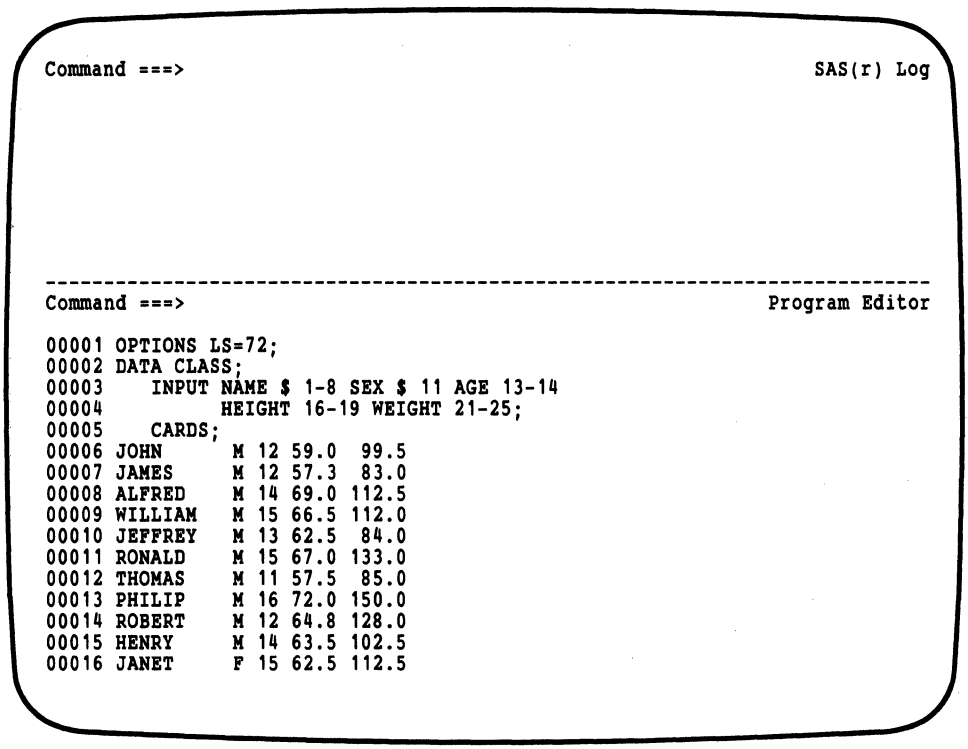
VARIABLE	N	MEAN	STANDARD DEVIATION	MINIMUM VALUE	MAXIMUM VALUE
HEIGHT	19	62.33684211	5.12707525	51.30000000	72.0000000
WEIGHT	19	100.02631579	22.77393349	50.50000000	150.0000000

PROC PLOT Output: Noninteractive



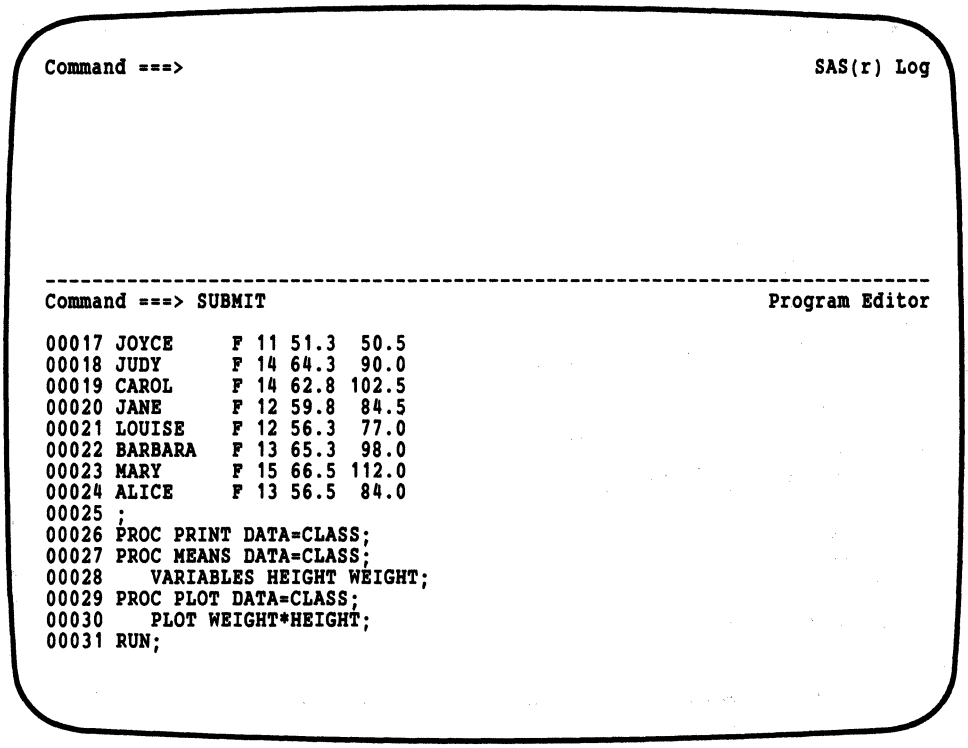
The SAS Display Manager System

Enter your SAS program into the program editor screen.



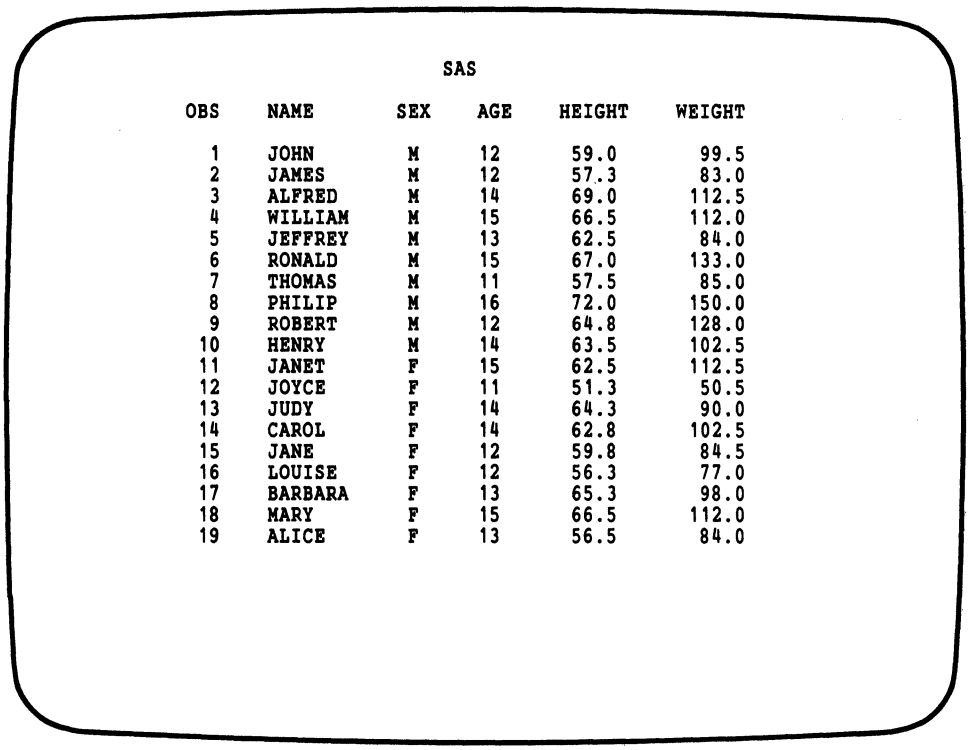
The SAS Display Manager System

Scroll forward to enter the rest of the program and issue the SUBMIT command to execute the program.



The SAS Display Manager System

When the first PROC step (PROC PRINT in this example) has finished executing, the results are displayed on the output screen.

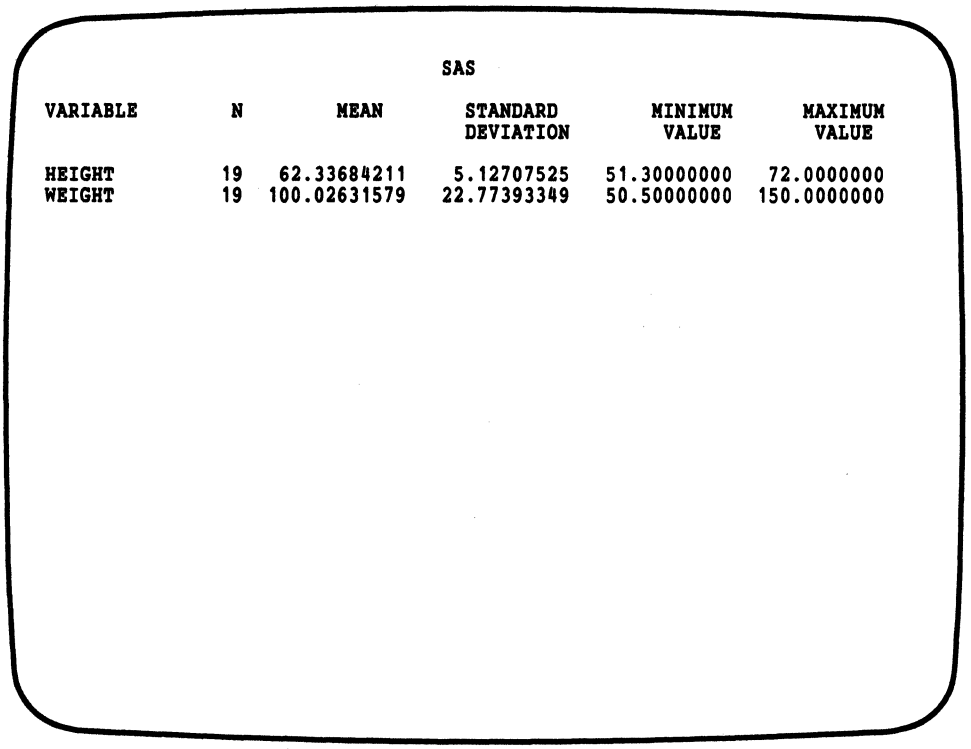


SAS

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0
3	ALFRED	M	14	69.0	112.5
4	WILLIAM	M	15	66.5	112.0
5	JEFFREY	M	13	62.5	84.0
6	RONALD	M	15	67.0	133.0
7	THOMAS	M	11	57.5	85.0
8	PHILIP	M	16	72.0	150.0
9	ROBERT	M	12	64.8	128.0
10	HENRY	M	14	63.5	102.5
11	JANET	F	15	62.5	112.5
12	JOYCE	F	11	51.3	50.5
13	JUDY	F	14	64.3	90.0
14	CAROL	F	14	62.8	102.5
15	JANE	F	12	59.8	84.5
16	LOUISE	F	12	56.3	77.0
17	BARBARA	F	13	65.3	98.0
18	MARY	F	15	66.5	112.0
19	ALICE	F	13	56.5	84.0

The SAS Display Manager System

Press the END key to display the output from the second PROC step (PROC MEANS in this example).

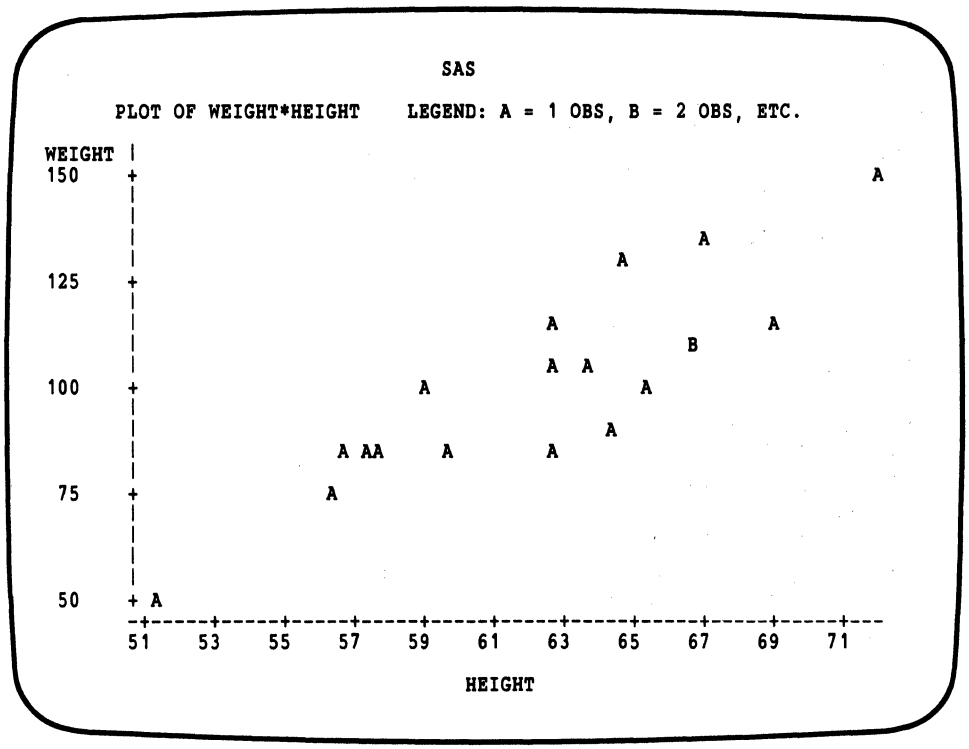


SAS

VARIABLE	N	MEAN	STANDARD DEVIATION	MINIMUM VALUE	MAXIMUM VALUE
HEIGHT	19	62.33684211	5.12707525	51.30000000	72.0000000
WEIGHT	19	100.02631579	22.77393349	50.50000000	150.0000000

The SAS Display Manager System

Press the END key again to display the output from the third PROC step (PROC PLOT in this example).



The SAS Display Manager System

Press the END key to return to the program editor/log screen. The SAS log is displayed on the log screen.

```
Command ===>                                                    SAS(r) Log

Copyright (c) 1985 SAS Institute Inc., Cary, N. C. 27511, U. S. A.
NOTE: VMS Version of SAS Release 5.XX at SAS INSTITUTE INC.
NOTE: LICENSED CPUID MODEL = 8600, SERIAL = 00000000.
NOTE: Enter HELP DMSHELP for help on the SAS Display Manager System.
  1 OPTIONS LS=72;
  2 DATA CLASS;
  3    INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
  4          HEIGHT 16-19 WEIGHT 21-25;
  5    CARDS;
NOTE: THE DATA SET WORK.CLASS HAS 19 OBSERVATIONS AND 5 VARIABLES.
NOTE: THE DATA STEP USED 00:00:00:82 CPU SECONDS, 688 PAGEFAULTS
 25 ;
 26 PROC PRINT DATA=CLASS;
NOTE: THE PROCEDURE PRINT USED 00:00:00:59 CPU SECONDS, 481 PAGEFAULTS
NOTE: THE PROCEDURE PRINTED PAGE 1.
 27 PROC MEANS DATA=CLASS;
 28    VARIABLES HEIGHT WEIGHT;
NOTE: THE PROCEDURE MEANS USED 00:00:00:44 CPU SECONDS, 213 PAGEFAULTS
NOTE: THE PROCEDURE PRINTED PAGE 2.
 29 PROC PLOT DATA=CLASS;
 30    PLOT WEIGHT*HEIGHT;
 31 RUN;
NOTE: THE PROCEDURE PLOT USED 00:00:01:11 CPU SECONDS, 562 PAGEFAULTS
NOTE: THE PROCEDURE PRINTED PAGE 3.
-------------------------------------------------------------------------------
Command ===>                                                  Program Editor
```

2.4 SAS Syntax and SAS Data Sets

Rules for Writing SAS Statements

SAS statements begin with an identifying keyword and end with a semicolon.

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;

data lines

PROC PRINT DATA=CLASS;
PROC MEANS DATA=CLASS;
   VARIABLES HEIGHT WEIGHT;
PROC PLOT DATA=CLASS;
   PLOT WEIGHT*HEIGHT;
```

SAS statements are free-format.

- They can begin anywhere, end anywhere.
- One statement can continue over several lines.
- Several statements can be on a line.
- Blanks—as many as you want—separate fields. Special characters also separate fields.

Note: we recommend that DATA and PROC statements start in column 1 and that other statements be indented.

Structure of SAS Data Sets

A SAS data set is a collection of data values arranged in a rectangular table.

	VARIABLES				
	NAME	**SEX**	**AGE**	**HEIGHT**	**WEIGHT**
observation 1	JOHN	M	12	59.0	99.5
observation 2	JAMES	M	12	57.3	83.0
observation 3	ALFRED	M	14	69.0	112.5
⋮	⋮	⋮	⋮	⋮	⋮
observation 19	ALICE	F	13	56.5	84.0

The columns in the table are called **variables**.

- Variables correspond to fields of data.
- Each variable is given a **name**.
- There are two kinds of variables.

  character variables: NAME, SEX (1-200 long)
  numeric variables: AGE, HEIGHT, WEIGHT (floating)

The rows in the table are called **observations** (or records). There is no limit on the number of observations.

Note: the rectangular structure of a SAS data set implies that every variable must exist for each observation.

Naming SAS Data Sets and Variables

Rules for SAS names:

- 1-8 characters
- start with A-Z or _
- continue with numbers, letters, or underscores.

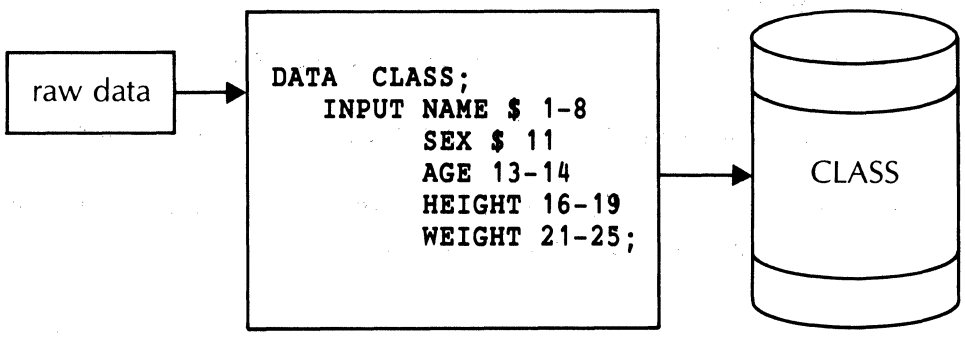
Suggestion: Choose meaningful data set and variable names.

Note: the underscore, '_' , **is not allowed** in SAS data set names for minicomputer environments.

Example:

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;
```

Creating SAS Data Sets



- Data must be in the form of a SAS data set before they can be analyzed by SAS procedures.
- Every SAS data set has a name and is physically stored on disk or tape. In simple jobs, the SAS data sets are stored on temporary space, but they can be stored permanently.
- Many SAS data sets can be created in a single job step.

Processing SAS Data Sets

- Data stored in a SAS data set are always referred to by name. The variables are referred to by name, and the SAS data set is referred to by name.

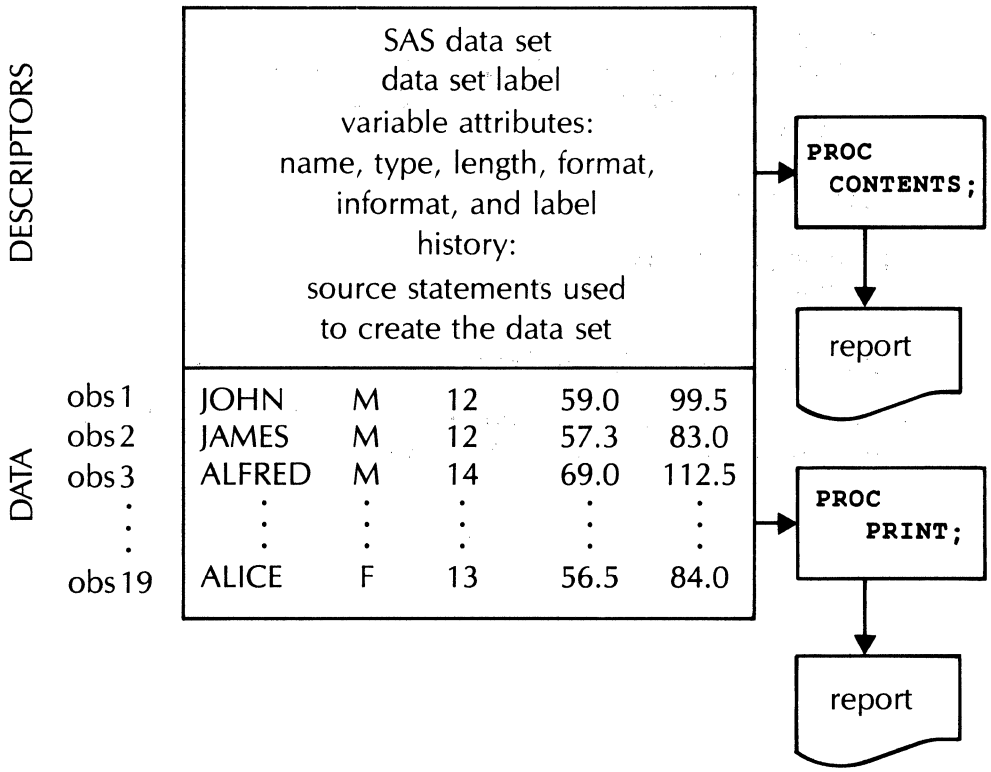
Example:

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;

data lines

PROC MEANS DATA=CLASS;
   VARIABLES HEIGHT WEIGHT;
```

- Many SAS data sets can be processed in a single job step.
- SAS data sets are usually processed in a simple sequential manner, one observation after another.

Documenting SAS Data Sets

- A SAS data set contains, in addition to the data values, descriptors with names and attributes of the variables.



- If data are stored as a SAS data set, all the original physical details of the data can be forgotten. Only the meanings of the variable names must be remembered.
- The PRINT procedure lists the data in a SAS data set.
- The CONTENTS procedure can be used to print the descriptor information in a SAS data set.

Data Portion of a SAS Data Set

```
PROC PRINT DATA=CLASS;
```

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0
3	ALFRED	M	14	69.0	112.5
4	WILLIAM	M	15	66.5	112.0
5	JEFFREY	M	13	62.5	84.0
6	RONALD	M	15	67.0	133.0
7	THOMAS	M	11	57.5	85.0
8	PHILIP	M	16	72.0	150.0
9	ROBERT	M	12	64.8	128.0
10	HENRY	M	14	63.5	102.5
11	JANET	F	15	62.5	112.5
12	JOYCE	F	11	51.3	50.5
13	JUDY	F	14	64.3	90.0
14	CAROL	F	14	62.8	102.5
15	JANE	F	12	59.8	84.5
16	LOUISE	F	12	56.3	77.0
17	BARBARA	F	13	65.3	98.0
18	MARY	F	15	66.5	112.0
19	ALICE	F	13	56.5	84.0

Descriptor Part of a SAS Data Set

Mainframe Environment

```
PROC CONTENTS DATA=CLASS;
```

```
                          CONTENTS PROCEDURE
                      CONTENTS OF SAS MEMBER WORK.CLASS


CREATED BY TSO USERID EDU       ON CPUID 00-0000-000000
AT 13:38 MONDAY, JANUARY 20, 1986   BY SAS RELEASE 5.XX
DSNAME=SYS86020.T133828.RA000.EDU.R0000105   OBSERVATIONS PER TRACK =515
BLKSIZE=19059   LRECL=37   GENERATED BY DATA
NUMBER OF OBSERVATIONS: 19   NUMBER OF VARIABLES: 5
MEMTYPE:   DATA

           ----ALPHABETIC LIST OF VARIABLES AND ATTRIBUTES-----
       # VARIABLE TYPE   LENGTH POSITION FORMAT       INFORMAT     LABEL
       3 AGE      NUM         8       13
       4 HEIGHT   NUM         8       21
       1 NAME     CHAR        8        4
       2 SEX      CHAR        1       12
       5 WEIGHT   NUM         8       29



------------------------------ SOURCE RECORDS --------------------------
| DATA CLASS;
|    INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
|          HEIGHT 16-19 WEIGHT 21-25;
|    CARDS;
------------------------------------------------------------------------
```

Descriptor Part of a SAS Data Set

Minicomputer Environment

```
PROC CONTENTS DATA=CLASS;
```

```
                                SAS

                          CONTENTS PROCEDURE
                    CONTENTS OF SAS MEMBER WORK.CLASS

NUMBER OF OBSERVATIONS: 19              NUMBER OF VARIABLES: 5
MEMTYPE:             DATA

          ----ALPHABETIC LIST OF VARIABLES AND ATTRIBUTES-----
       # VARIABLE TYPE   LENGTH POSITION FORMAT       INFORMAT     LABEL
       3 AGE      NUM         8        9
       4 HEIGHT   NUM         8       17
       1 NAME     CHAR        8        0
       2 SEX      CHAR        1        8
       5 WEIGHT   NUM         8       25
```

2.5 Exercise

Writing a SAS Program

A branch office in Chicago has stored its monthly revenue data on cards. Write a SAS job that reads the data listed below and solves the following problems.

- Set the line size to 72 and the page size to 40.
- Create a SAS data set.
- Print the resulting SAS data set.
- Compute the average monthly revenue.

	1	2	3	4	5	6	7	8	9	10	11	12
1	J	A	N		3	6	5	1	.	2	3	
2	F	E	B		2	8	1	3	.	1	2	
3	M	A	R		3	8	1	4	.	4	1	
4	A	P	R		3	4	7	1	.	6	6	
5	M	A	Y		3	2	4	3	.	4	2	
6	J	U	N		3	9	6	0	.	8	6	
7	J	U	L		4	1	0	9	.	3	7	
8	A	U	G		3	7	6	9	.	1	7	
9	S	E	P		4	0	8	7	.	5	6	
10	O	C	T		4	0	1	3	.	7	7	
11	N	O	V		3	9	6	3	.	2	4	
12	D	E	C		4	1	2	5	.	9	7	

Answer

- Set the line size and page size.
  LS=72 PS=40
- Name the data set.
  CHIC
- What variables do we input?
  MONTH, REVENUE
- Ready for the data.
- Enter the data.
- Print the data set.
  PROC PRINT DATA = CHIC
- Find the average monthly revenue.

SAS Log: Mainframe Environment

```
          SAS(R) LOG   OS SAS 5.XX      MVS/XA TSO USER TSOUSER

NOTE: COPYRIGHT (C) 1984 SAS INSTITUTE INC., CARY, N.C.  27511,
U.S.A.
NOTE: SAS RELEASE 5.XX AT SAS INSTITUTE DATA CENTER (00000000)

1          OPTIONS LS=72 PS=40;
2          DATA CHICAGO;
3             INPUT MONTH $ 1-3
4                   REVENUE 5-11;
5             CARDS;

NOTE: DATA SET WORK.CHICAGO HAS 12 OBSERVATIONS AND 2 VARIABLES. 1271 OB
S/TRK
NOTE: THE DATA STATEMENT USED 0.05 SECONDS AND 1032K.

18         ;
19         PROC PRINT DATA=CHICAGO;
NOTE: THE PROCEDURE PRINT USED 0.09 SECONDS AND 1220K
      AND PRINTED PAGE 1.

20         PROC MEANS DATA=CHICAGO;
21            VARIABLES REVENUE;
NOTE: THE PROCEDURE MEANS USED 0.09 SECONDS AND 1240K
      AND PRINTED PAGE 2.
NOTE: SAS USED 1240K MEMORY.

NOTE: SAS INSTITUTE INC.
      SAS CIRCLE
      PO BOX 8000
      CARY, N.C. 27511-8000
```

SAS Log: Minicomputer Environment

```
                    S A S   L O G    VMS SAS 5.XX

Copyright (c) 1985 SAS Institute Inc., Cary, N. C. 27511, U. S. A.
NOTE: VMS Version of SAS Release 5.XX at SAS INSTITUTE INC.
NOTE: LICENSED CPUID MODEL = 8600, SERIAL = 00000000.

     1 OPTIONS LS=72 PS=40;
     2 DATA CHICAGO;
     3    INPUT MONTH $ 1-3
     4          REVENUE 5-11;
     5    CARDS;
NOTE: THE DATA SET WORK.CHICAGO HAS 12 OBSERVATIONS AND 2 VARIABLES.
NOTE: THE DATA STEP USED 00:00:00.49 CPU SECONDS, 708 PAGEFAULTS
    18 ;
    19 PROC PRINT DATA=CHICAGO;
NOTE: THE PROCEDURE PRINT USED 00:00:00.21 CPU SECONDS, 264 PAGEFAULTS
NOTE: THE PROCEDURE PRINTED PAGE 1.
    20 PROC MEANS DATA=CHICAGO;
    21    VARIABLES REVENUE;
NOTE: THE PROCEDURE MEANS USED 00:00:00.17 CPU SECONDS, 243 PAGEFAULTS
NOTE: THE PROCEDURE PRINTED PAGE 2.
NOTE: SAS INSTITUTE INC., SAS CIRCLE, BOX 8000, CARY, N. C., 27511-8000
```

Procedure Output

PROC PRINT Output

OBS	MONTH	REVENUE
1	JAN	3651.23
2	FEB	2813.12
3	MAR	3814.41
4	APR	3471.66
5	MAY	3243.42
6	JUN	3960.86
7	JUL	4109.37
8	AUG	3769.17
9	SEP	4087.56
10	OCT	4013.77
11	NOV	3963.24
12	DEC	4125.97

PROC MEANS Output

VARIABLE	N	MEAN	STANDARD DEVIATION	MINIMUM VALUE	MAXIMUM VALUE
REVENUE	12	3751.9816667	401.13701511	2813.1200000	4125.970000

3. Reading and Processing Raw Data

3.1 Creating SAS Data Sets

3.2 Multiple Records per Observation

3.3 Exercises

3.1 Creating SAS Data Sets

The DATA Statement

The two major functions of the DATA statement are to

- signal the beginning of the DATA step
- name the data sets being created in the DATA step.

General form of the DATA statement:

DATA *SASdataset(s)*;

Naming Temporary SAS Data Sets

- Name supplied by user:

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;

  data lines
```

```
NOTE: DATA SET WORK.CLASS HAS 19 OBSERVATIONS AND 5 VARIABLES. 5
15 OBS/TRK
```

- Name omitted by user (SAS System supplies the name):

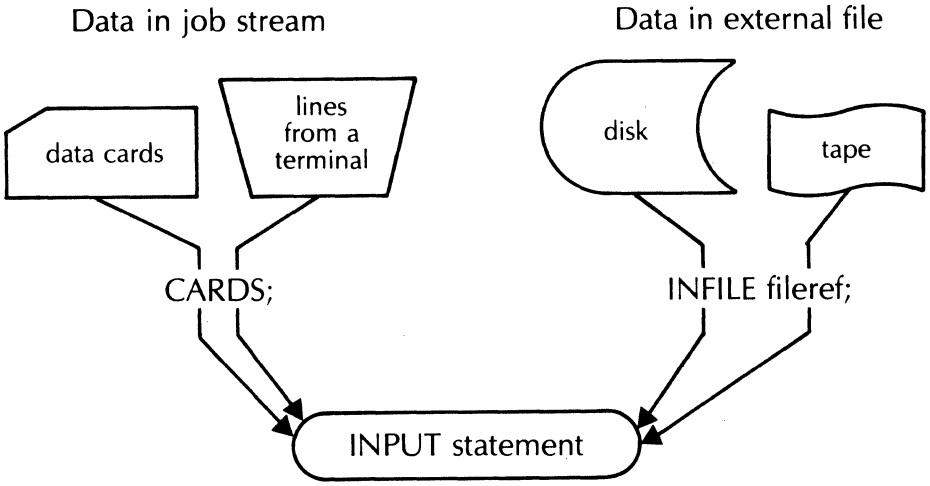
```
DATA;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;

  data lines
```

```
NOTE: DATA SET WORK.DATA1 HAS 19 OBSERVATIONS AND 5 VARIABLES. 5
15 OBS/TRK
```

The first time the data set name is omitted in a SAS job, the name DATA1 is used. The second time the data set name is omitted, the name DATA2 is used, and so on.

Inputting Raw Data

The SAS System can handle data in virtually any form, from almost any input file.



Note: regardless of where the data are stored, the same INPUT statement is used.

Functions of the CARDS, INFILE and INPUT Statements

- Reading the data

  INPUT statement
  reads raw data lines

  assigns names to the SAS variables that correspond to the fields

  has three modes: COLUMN, LIST, and FORMATTED.

- Pointing to the data file

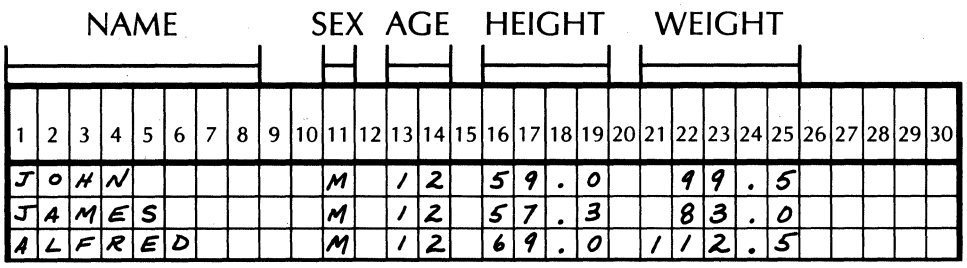
  CARDS statement
  indicates that data records follow immediately.

  INFILE statement
  points to an external file where the raw data are stored.

Column Input

Specify where to find the values on the input record by column position.

INPUT *variable startcol-endcol;*




```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;

data lines
```

Restrictions:

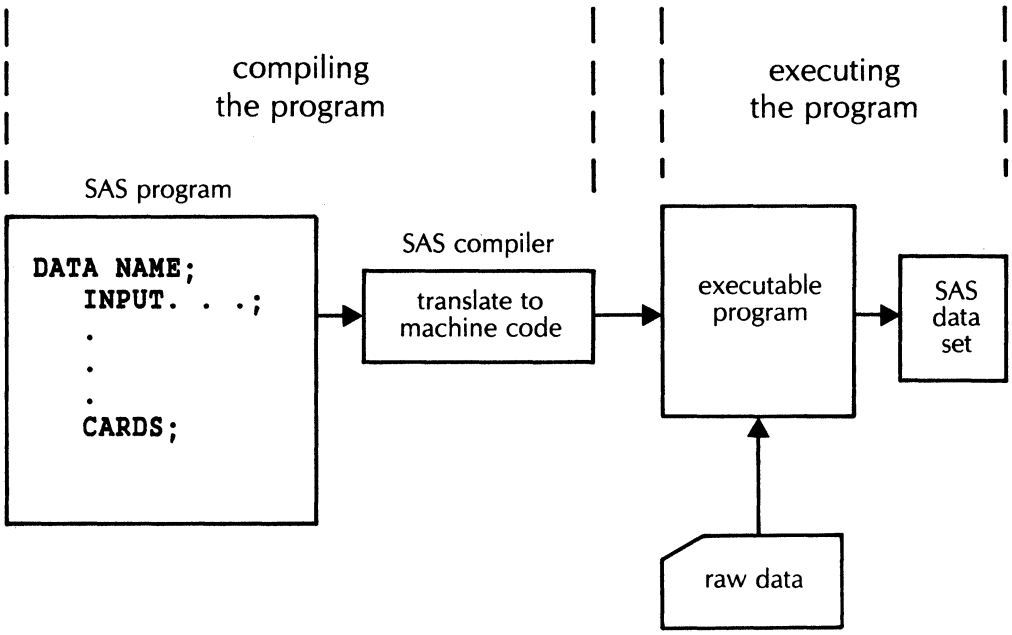
- The field positions for the variables are fixed.
- A $ is used to indicate character variables.

Program Flow of the DATA Step

Looking behind the Scenes

The DATA step involves writing a compact SAS program to process data. The SAS System processes the program in two steps.

- It compiles the program.
- It executes the program.

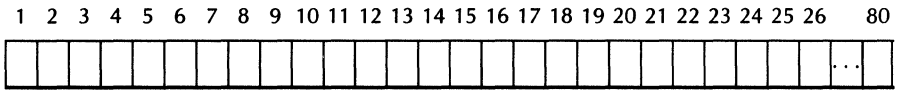



```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;

data lines
```

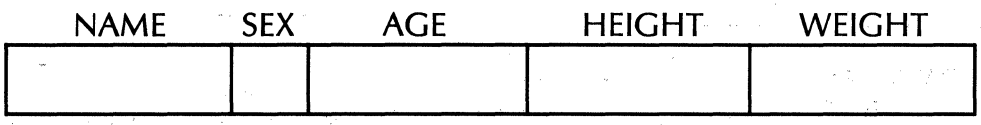
Compiling the DATA Step

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;
```

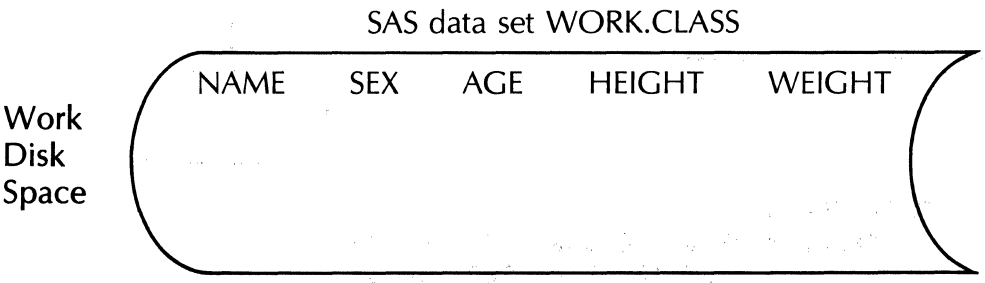
Input Buffer—temporary storage for an input record



Program Data Vector—temporary storage for a SAS observation



The descriptive information is written to the SAS data set (mainframe environments).

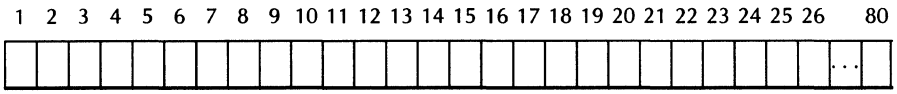


Executing the DATA Step

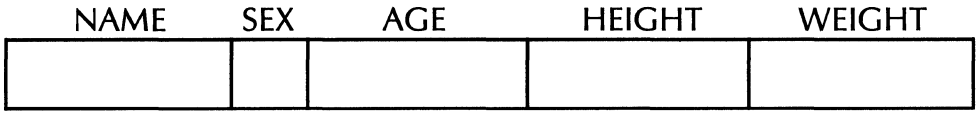
- The program data vector is initialized to missing before each execution of the DATA step.
- The DATA step is executed once for each observation in the data set.

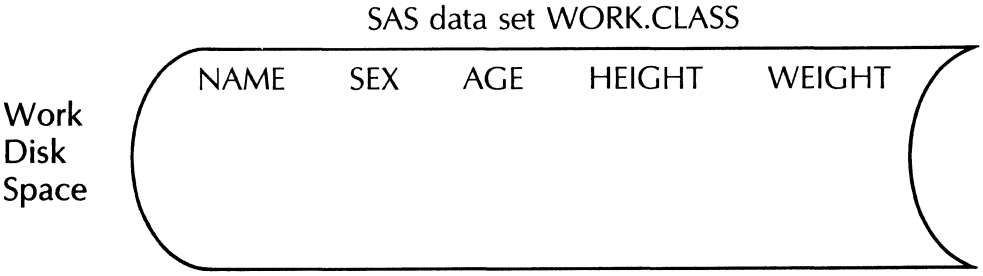
```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;
JOHN       M 12 59.0  99.5
JAMES      M 12 57.3  83.0
ALFRED     M 14 69.0 112.5
```

Input Buffer



Program Data Vector





Column Input Features

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;
```

- Data fields can be read in any order.

```
INPUT WEIGHT 21-25 AGE 13-14 NAME $ 1-8
      SEX $ 11 HEIGHT 16-19;
```

- Blank fields are read as missing.
- Embedded blanks are allowed in character data values.

```
"JOHN SMITH"
```

- Character data values can range from 1 to 200 characters in length.

```
INPUT COMPANY $ 1-200;
```

- Fields or parts of fields can be reread.

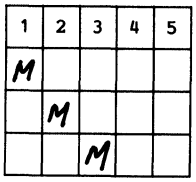
```
INPUT NAME $ 1-8 INITIAL $ 1;
```

Reading Data Values with Column Input

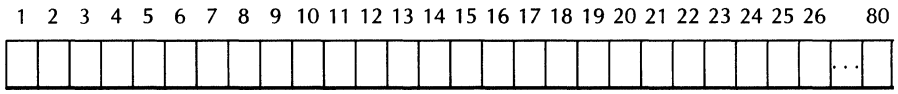
Character Values

- Character data values in a field are left aligned.
- The number of bytes used to store character data values is determined by the field width.

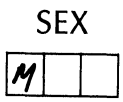
```
INPUT SEX $ 1-3;
```



Input Buffer



Program Data Vector



Advantage: When making comparisons, specify:

```
IF SEX='M' THEN DELETE;
```

instead of:

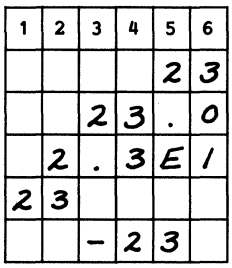
```
IF SEX='M  ' OR
   SEX=' M ' OR
   SEX='  M' THEN DELETE;
```

Reading Data Values with Column Input

Numeric Values

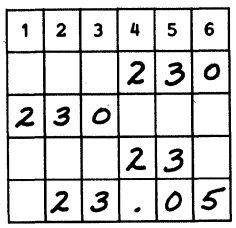
- Numeric values can occur anywhere in the field.
- The sign and decimal or decimal exponent can be specified.

```
INPUT X 1-6;
```



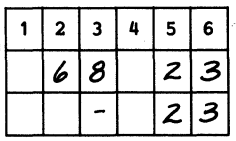
decimal parameter

```
INPUT X 1-6 .1;
```



Embedded blanks are not allowed in a numeric field.

```
INPUT X 1-6;
```



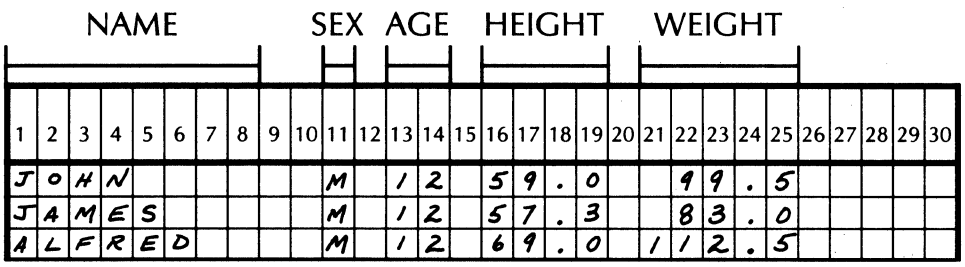
invalid data
invalid data

List Input

List the variables in the order they appear in the input data.

INPUT *list of variables;*

```
INPUT NAME $ SEX $ AGE HEIGHT WEIGHT;
```



The INPUT statement scans over blanks until it finds a character, then reads in the field up to the next blank.

Rules and restrictions:

- A $ is used to indicate character variables.
- Every field must be specified in order.
- Fields must be separated with blanks.
- Data values for character variables are restricted:

  no embedded blanks as in JOHN SMITH
  default length of 8 characters.

- Blank fields cause the matching of variable names and values to get out of sync.

Why List Input Is Needed

Example: The raw data for the students in the class were stored with one blank between each field instead of being stored in fixed fields. Read the data.

```
DATA CLASS;
   INPUT NAME $ SEX $ AGE HEIGHT WEIGHT;
   CARDS;
JOHN M 12 59.0 99.5
JAMES M 12 57.3 83.0
ALFRED M 14 69.0 112.5
WILLIAM M 15 66.5 112.0
JEFFREY M 13 62.5  84.0
RONALD M 15 67.0 133.0
THOMAS M 11 57.5  85.0
PHILIP M 16 72.0 150.0
ROBERT M 12 64.8 128.0
HENRY M 14 63.5 102.5
JANET F 15 62.5 112.5
JOYCE F 11 51.3  50.5
JUDY F 14 64.3  90.0
CAROL F 14 62.8 102.5
JANE F 12 59.8  84.5
LOUISE F 12 56.3  77.0
BARBARA F 13 65.3  98.0
MARY F 15 66.5 112.0
ALICE F 13 56.5  84.0
;
```

Why List Input Is Needed

```
PROC PRINT DATA=CLASS;
```

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0
3	ALFRED	M	14	69.0	112.5
4	WILLIAM	M	15	66.5	112.0
5	JEFFREY	M	13	62.5	84.0
6	RONALD	M	15	67.0	133.0
7	THOMAS	M	11	57.5	85.0
8	PHILIP	M	16	72.0	150.0
9	ROBERT	M	12	64.8	128.0
10	HENRY	M	14	63.5	102.5
11	JANET	F	15	62.5	112.5
12	JOYCE	F	11	51.3	50.5
13	JUDY	F	14	64.3	90.0
14	CAROL	F	14	62.8	102.5
15	JANE	F	12	59.8	84.5
16	LOUISE	F	12	56.3	77.0
17	BARBARA	F	13	65.3	98.0
18	MARY	F	15	66.5	112.0
19	ALICE	F	13	56.5	84.0

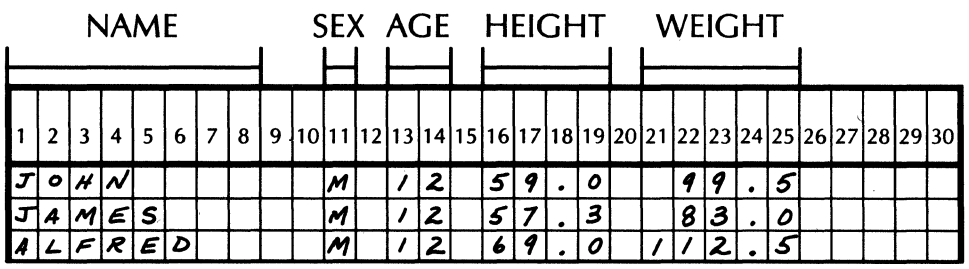
Formatted Input

Specify the starting location and field widths (similar to FORTRAN and PL/I). Move an input "pointer" to the starting position of the field, then specify the variable and an informat.

INPUT *pointer control variable informat*;

informats:	***w.***	numeric width
	w.d	numeric with decimal
	$w.	character
pointer controls:	***@n***	go to column *n*
	+n	move the pointer *n* positions

```
INPUT NAME $8. @11 SEX $1. +1 AGE 2.
      +1 HEIGHT 4. +1 WEIGHT 5.;
```



With formatted input, you can read data in nonstandard numeric or character formats.

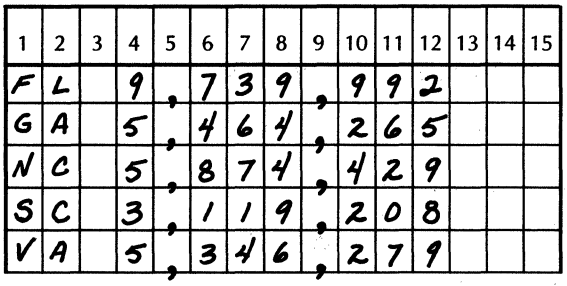
Selected Informats

w.	standard numeric
w.d	standard numeric with decimal
$w.	standard character
$CHAR*w.*	characters with blanks
HEX*w.*	numeric hexadecimal
$HEX*w.*	character hexadecimal
IB*w.d*	integer binary
PIB*w.d*	positive integer binary
PD*w.d*	packed decimal
PK*w.*	unsigned packed decimal
RB*w.d*	real binary (floating point)
ZD*w.d*	zoned decimal
ZDB*w.d*	zoned decimal with blanks
COMMA*w.d*	commas in numbers
E*w.*	scientific notation
BZ*w.d*	blanks are zeros
$VARYING*w.*	varying-length character values

See the *SAS User's Guide: Basics* for additional information on SAS informats.

Formatted Input Example

Example: The postal codes and 1980 populations for five states in the southeast are shown in the table below. Read the data and print the SAS data set.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F	L		9	,	7	3	9	,	9	9	2			
G	A		5	,	4	6	4	,	2	6	5			
N	C		5	,	8	7	4	,	4	2	9			
S	C		3	,	1	1	9	,	2	0	8			
V	A		5	,	3	4	6	,	2	7	9			

```
DATA POPS;
   INPUT STATE $2. +1 POP COMMA9.;
   CARDS;
FL 9,739,992
GA 5,464,265
NC 5,874,429
SC 3,119,208
VA 5,346,279
;
PROC PRINT DATA=POPS;
```

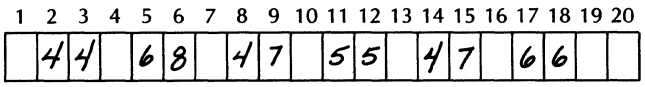
OBS	STATE	POP
1	FL	9739992
2	GA	5464265
3	NC	5874429
4	SC	3119208
5	VA	5346279

Informat Lists and Numbered Variable Names

Consider the following INPUT statement that reads gross sales (in thousands) for a company over 6 consecutive months.

```
INPUT JAN 3. FEB 3. MAR 3. APR 3. MAY 3. JUN 3.;
```

Input Buffer



Special features allow the INPUT statement to be written with less coding:

- Variables and informats can be grouped separately with parentheses.

```
INPUT (JAN FEB MAR APR MAY JUN) (3. 3. 3. 3. 3. 3.);
```

- The informat list is recycled to satisfy the variable list.

```
INPUT (JAN FEB MAR APR MAY JUN) (3.);
```

- Numbered variable names can be used in abbreviated form to refer to several related variables.

```
INPUT (MONTH1 MONTH2 MONTH3 MONTH4 MONTH5 MONTH6) (3.);
```

can be rewritten as:

```
INPUT (MONTH1-MONTH6) (3.);
```

Mixing Input Styles

Example: Mix the three INPUT styles (list, column, and formatted) in one INPUT statement.

```
DATA CLASS;
   INPUT NAME $ @11 SEX $1. AGE 13-14
         HEIGHT @21 WEIGHT 5.;
   CARDS;
JOHN       M 12 59.0  99.5
JAMES      M 12 57.3  83.0
ALFRED     M 14 69.0 112.5
     .
     .
     .
PROC PRINT DATA=CLASS;
```

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0
3	ALFRED	M	14	69.0	112.5
4	WILLIAM	M	15	66.5	112.0
5	JEFFREY	M	13	62.5	84.0
6	RONALD	M	15	67.0	133.0
7	THOMAS	M	11	57.5	85.0
8	PHILIP	M	16	72.0	150.0
9	ROBERT	M	12	64.8	128.0
10	HENRY	M	14	63.5	102.5
11	JANET	F	15	62.5	112.5
12	JOYCE	F	11	51.3	50.5
13	JUDY	F	14	64.3	90.0
14	CAROL	F	14	62.8	102.5
15	JANE	F	12	59.8	84.5
16	LOUISE	F	12	56.3	77.0
17	BARBARA	F	13	65.3	98.0
18	MARY	F	15	66.5	112.0
19	ALICE	F	13	56.5	84.0

3.2 Multiple Records per Observation

Ways to Read Multiple Records

There are several ways to read an observation that continues over multiple input records.

Multiple records per observation can be processed with

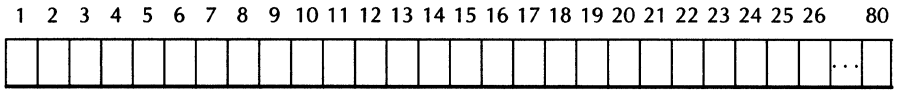
- multiple INPUT statements
- the / pointer control
- the #*n* pointer control.

Multiple INPUT Statements

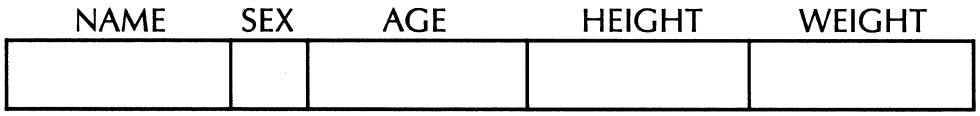
Each time an INPUT statement is executed, a new record is loaded into the input buffer.

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11;
   INPUT AGE 4-5;
   INPUT HEIGHT 1-5 WEIGHT 6-10;
   CARDS;
JOHN      M
   12
 59.0 99.5
      .
      .
      .
```

Input Buffer



Program Data Vector



Note: this method defines a one-line input buffer.

Multiple INPUT Statements

Example: Use the multiple INPUT statement method to read the data for JOHN and JAMES.

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11;
   INPUT AGE 4-5;
   INPUT HEIGHT 1-5 WEIGHT 6-10;
   CARDS;
JOHN      M
   12
 59.0 99.5
JAMES     M
   12
 57.3 83.0
;
PROC PRINT DATA=CLASS;
```

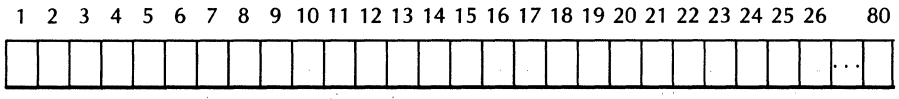
OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0

The / Pointer Control

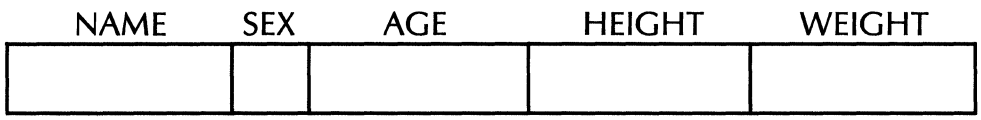
Each time a / is encountered on an INPUT statement, a new record is loaded into the input buffer.

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 / AGE 4-5 /
         HEIGHT 1-5 WEIGHT 6-10;
   CARDS;
JOHN       M
   12
 59.0 99.5
   .
   .
   .
```

Input Buffer



Program Data Vector



Note: this method defines a one-line input buffer.

The / Pointer Control

Example: Use the / pointer control in the INPUT statement to read the data for JOHN and JAMES.

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 / AGE 4-5 /
         HEIGHT 1-5 WEIGHT 6-10;
   CARDS;
JOHN       M
   12
 59.0 99.5
JAMES      M
   12
 57.3 83.0
;
PROC PRINT DATA=CLASS;
```

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0

The #n Pointer Control

The #*n* pointer control is used to advance to the first column of the *n*th record in a group of records.

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 #2 AGE 4-5
         #3 HEIGHT 1-5 WEIGHT 6-10;
   CARDS;
JOHN       M
   12
 59.0 99.5
   .
   .
   .
```

Note: the highest *n* value following the # pointer control defines the size of the group and the number of lines in the input buffer.

Input Buffer (3 lines)

	1	2	3	4	5	6	7	8	9	10	11
#1	J	O	H	N							M
#2				1	2						
#3		5	9	.	0		9	9	.	5	

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT

The #n Pointer Control

Example: Use the *#n* method to read the data for JOHN and JAMES.

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 #2 AGE 4-5
         #3 HEIGHT 1-5 WEIGHT 6-10;
   CARDS;
JOHN       M
   12
 59.0 99.5
JAMES      M
   12
 57.3 83.0
;
PROC PRINT DATA=CLASS;
```

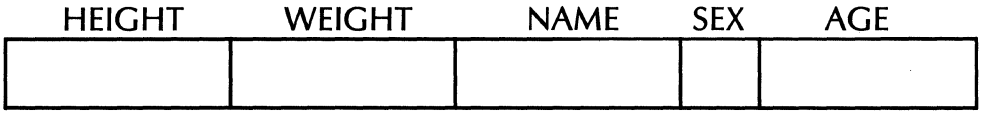
OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0

The #n Pointer Control

The multiple-line input buffer allows the records in a group to be read in any order:

```
INPUT #3 HEIGHT 1-5 WEIGHT 6-10 #1
      NAME $ 1-8 SEX $11 #2 AGE 4-5;
```

Program Data Vector



Example: Use the *#n* method to read the records in a different order.

```
DATA CLASS;
   INPUT #3 HEIGHT 1-5 WEIGHT 6-10 #1
         NAME $ 1-8 SEX $11 #2 AGE 4-5;
   CARDS;
JOHN       M
   12
 59.0 99.5
JAMES      M
   12
 57.3 83.0
;
PROC PRINT DATA=CLASS;
```

OBS	HEIGHT	WEIGHT	NAME	SEX	AGE
1	59.0	99.5	JOHN	M	12
2	57.3	83.0	JAMES	M	12

3.3 Exercises

3.1 Given the following two raw data records:

```
Column      1234567890
            452 857
            763 943
              X   Y
```

what will be the values of the variables X and Y in the SAS data sets created by the following DATA steps?

a.
```
DATA; INPUT X Y;
  CARDS;
```

b.
```
DATA; INPUT X 1-3 Y 5-7;
  CARDS;
```

c.
```
DATA; INPUT X 1-2 Y 6-7;
  CARDS;
```

d.
```
DATA; INPUT X $ Y $;
  CARDS;
```

e.
```
DATA; INPUT X $ 1-7 Y $ 2-5;
  CARDS;
```

f.
```
DATA; INPUT X 3. Y 3.;
  CARDS;
```

g.
```
DATA; INPUT X 3.2 @6 Y 3.;
  CARDS;
```

h.
```
DATA; INPUT X/Y;
  CARDS;
```

SAS Data Sets

X	Y
452 763	857 943
452 763	857 943
4.5 7.6	57 43
"452" "763"	"857" "943"
" 452 □ 857" 763 □ 943	
452 457	■57 ■94
4.52 7.63	57□ 43□
452	763

Exercises

3.2 A researcher prepared a summary of geographical information and stored it using this record format.

Variable Name	Field Location	Variable Type
STATE	1-3	character
COUNTY	5-12	character
SQMILE	14-19	numeric
REGION	23-25	numeric
TRACT	29-31	numeric
CODE	34-35	character
RAINFALL	40-46	numeric
TEMP	50-54	numeric
TEMPTYPE	59	character

Using this summary, code the INPUT statement using

a. list input

b. column input

c. formatted input.

Exercises

3.3 Cards were punched from the following coding sheet and contain five data values for each observation. The data values in order from left to right on each data line represent the variables CODE, REVENUE, NUMBER, FLIGHT, and FUEL, respectively.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A	A	B		1	6	4	.	1		2	0	1		4	6	8		1	0										
A	C		1	9	.	2		1	9	6		2	3	1			1	7											
		A	B	A		1	6		1	5		1	9	8		5													

Prepare SAS statements to place the data into a SAS data set named AIRLINE.

Exercises

3.4* A survey of attitudes toward nuclear power plants produced data with a record format described in the table below.

Variable Name	Record Number	Field Location	Variable Type
REGION	1	1-20	character
PLANTS	1	21-25	numeric
PROXIMIT	1	26-30	numeric
LOCATION	1	31-40	character
ACCIDENT	2	26-30	numeric
ENERGY	2	31-35	numeric
FAVOR	2	36	character
QUEST1	3	1	character
QUEST2	3	2	character
QUEST3	3	3	character
QUEST4	3	4	character

The data are stored on cards.

Prepare SAS statements for reading the data into a SAS data set named NUCLEAR.

* See Section 3.2.

4. Listing and Sorting Data

4.1 Introduction to the PROC Step

4.2 Listing Data Values

4.3 Sorting SAS Data Sets

4.4 Exercises

4.1 Introduction to the PROC Step

Invoking SAS Procedures

The **PROC** (or **PROCEDURE**) statement is used to invoke a SAS procedure.

SAS procedures are computer programs that

- read SAS data sets
- compute statistics
- print results
- create SAS data sets.

General form of the PROC statement:

PROC *program options*;

Examples:

```
PROC MEANS;

PROC MEANS SUM MAXDEC=2 DATA=CLASS;
```

Note: the DATA= option specifies the SAS data set to be processed. If omitted, the procedure will process the most recently created SAS data set.

Selected Statements Used with Procedures

VARIABLES Statement

Specifies the variables in the selected SAS data set to be processed by the procedure.

General form of the VARIABLES statement:

VARIABLES *variable list*;

or

VAR *variable list*;

BY Statement

Allows by-group (subgroup) processing.

General form of the BY statement:

BY *variable list*;

Note: the SAS data set to be processed with a BY statement must be grouped by the variables in the BY statement.

Selected Statements Used with Procedures

TITLE Statement

Defines text to be printed at the top of the output pages.

Up to 10 titles can be specified.

General form of the TITLE statement:

TITLE*n* *'title'*;

FOOTNOTE Statement

Defines text to be printed at the bottom of the output pages.

Up to 10 footnotes can be specified.

General form of the FOOTNOTE statement:

FOOTNOTE*n* *'footnote'*;

Note: specifying TITLE*n* (FOOTNOTE*n*) defines a new TITLE*n* (FOOTNOTE*n*) and suppresses all titles (footnotes) with a larger *n* value.

Data for Upcoming Examples

The data described below are used in a series of examples in the remainder of this chapter.

Payroll information for a small company has been stored on cards with the following record format:

Field Description	Field Location	Type of Data
DEPARTMENT	1-3	numeric
NAME	5-12	character
EMPLOYEE NUMBER	14-18	numeric
SEX	20	character
NET PAY	22-28	numeric
GROSS PAY	30-36	numeric

Create a SAS data set from the data.

```
DATA PAYROLL;
   INPUT DEPT 1-3 NAME $ 5-12 NUMBER 14-18
         SEX $ 20 NETPAY 22-28 GROSSPAY 30-36;
   CARDS;

data lines
```

4.2 Listing Data Values

The PRINT Procedure

The PRINT procedure prints the data values in a SAS data set.

Features:

- automatic formatting
- columns labeled with variable names
- special handling of control breaks
- printing summaries
- optimization of page space
- special BY/ID formatting

The PRINT Procedure

General form of the PROC PRINT statement:

PROC PRINT *options*;

Selected options:

DATA=	specifies the SAS data set to be processed.
DOUBLE	double-spaces the output.
LABEL	uses variable labels for column headings.
SPLIT=	indicates how to split labels.
NOOBS	suppresses OBS column in the output.
N	requests that the number of observations be printed at the end of the data set, or at the end of each BY group, if a BY statement is used.

Selected statements used with PROC PRINT:

BY *variable list*;
VAR *variable list*;
ID *variable list*;
PAGEBY *byvariable*;
SUM *variable list*;

The PRINT Procedure

Example: List the data in the PAYROLL data set using the default features of PROC PRINT.

```
PROC PRINT DATA=PAYROLL;
```

OBS	DEPT	NAME	NUMBER	SEX	NETPAY	GROSSPAY
1	915	LARSON	11357	F	265.47	383.92
2	915	RYAN	10961	M	291.56	399.20
3	915	WOOD	8113	F	238.17	372.24
4	915	MOORE	12649	M	557.18	728.12
5	915	HELMS	3658	M	479.26	701.26
6	916	SHIRES	10052	F	487.12	729.22
7	916	RICHARDS	5781	M	319.27	472.84
8	916	SMITH	6237	F	207.09	345.88
9	916	MURPHY	6927	M	272.66	385.11
10	917	HERZOG	6433	M	692.57	1045.26
11	917	TALL	11931	M	355.19	492.26
12	917	HIGGINS	11658	M	777.50	1235.46
13	917	DONNER	10554	F	669.06	972.29
14	917	EVANS	7716	M	224.36	310.40
15	917	POWELL	8123	F	789.39	1271.54

The PRINT Procedure

Example: Rearrange the order in which the variables are printed.

Do not print the observation number.

Double-space the output.

Put a title at the top of the listing.

```
PROC PRINT DOUBLE NOOBS;
   VAR NAME NUMBER DEPT SEX NETPAY GROSSPAY;
   TITLE 'LISTING OF EMPLOYEE PAYROLL FILE';
```

The PRINT Procedure

LISTING OF EMPLOYEE PAYROLL FILE

NAME	NUMBER	DEPT	SEX	NETPAY	GROSSPAY
LARSON	11357	915	F	265.47	383.92
RYAN	10961	915	M	291.56	399.20
WOOD	8113	915	F	238.17	372.24
MOORE	12649	915	M	557.18	728.12
HELMS	3658	915	M	479.26	701.26
SHIRES	10052	916	F	487.12	729.22
RICHARDS	5781	916	M	319.27	472.84
SMITH	6237	916	F	207.09	345.88
MURPHY	6927	916	M	272.66	385.11
HERZOG	6433	917	M	692.57	1045.26
TALL	11931	917	M	355.19	492.26
HIGGINS	11658	917	M	777.50	1235.46
DONNER	10554	917	F	669.06	972.29
EVANS	7716	917	M	224.36	310.40
POWELL	8123	917	F	789.39	1271.54

The PRINT Procedure

Example: Print only the variables NUMBER, NETPAY, and GROSSPAY.

Compute the totals for NETPAY and GROSSPAY.

Use an appropriate title.

```
PROC PRINT;
   VAR NUMBER NETPAY GROSSPAY;
   SUM NETPAY GROSSPAY;
   TITLE 'NET PAY AND GROSS PAY TOTALS';
```

NET PAY AND GROSS PAY TOTALS

OBS	NUMBER	NETPAY	GROSSPAY
1	11357	265.47	383.92
2	10961	291.56	399.20
3	8113	238.17	372.24
4	12649	557.18	728.12
5	3658	479.26	701.26
6	10052	487.12	729.22
7	5781	319.27	472.84
8	6237	207.09	345.88
9	6927	272.66	385.11
10	6433	692.57	1045.26
11	11931	355.19	492.26
12	11658	777.50	1235.46
13	10554	669.06	972.29
14	7716	224.36	310.40
15	8123	789.39	1271.54
		6625.85	9845.00

4.3 Sorting SAS Data Sets

The SORT Procedure

The SORT procedure can rearrange the observations in a SAS data set or create a new SAS data set containing the rearranged observations.

The SORT procedure

- can sort on multiple fields
- can sort in ascending or descending order
- does not generate printed output
- uses the ASCII collating sequence to sort character variables in minicomputer environments
- uses the EBCDIC collating sequence to sort character variables in mainframe environments
- treats a missing value as the smallest possible value.

Note: see the *SAS User's Guide: Basics* for the ASCII and EBCDIC collating sequences.

The SORT Procedure

General form of the PROC SORT statement:

PROC SORT *options*;

Selected options:

DATA=*SASdataset*
names the input data set.

OUT=*SASdataset*
names the output data set.

Statements used with PROC SORT:

BY *variable list*;
BY DESCENDING *variable*;

The SORT Procedure

Example: Sort the PAYROLL data set alphabetically by NAME.

```
PROC SORT DATA=PAYROLL;
   BY NAME;
PROC PRINT;
   TITLE 'PAYROLL FILE';
   FOOTNOTE 'LISTED IN ALPHABETICAL ORDER';
```

PAYROLL FILE

OBS	DEPT	NAME	NUMBER	SEX	NETPAY	GROSSPAY
1	917	DONNER	10554	F	669.06	972.29
2	917	EVANS	7716	M	224.36	310.40
3	915	HELMS	3658	M	479.26	701.26
4	917	HERZOG	6433	M	692.57	1045.26
5	917	HIGGINS	11658	M	777.50	1235.46
6	915	LARSON	11357	F	265.47	383.92
7	915	MOORE	12649	M	557.18	728.12
8	916	MURPHY	6927	M	272.66	385.11
9	917	POWELL	8123	F	789.39	1271.54
10	916	RICHARDS	5781	M	319.27	472.84
11	915	RYAN	10961	M	291.56	399.20
12	916	SHIRES	10052	F	487.12	729.22
13	916	SMITH	6237	F	207.09	345.88
14	917	TALL	11931	M	355.19	492.26
15	915	WOOD	8113	F	238.17	372.24

LISTED IN ALPHABETICAL ORDER

The SORT Procedure

Example: Create a new SAS data set by rearranging the PAYROLL data set in order of SEX and GROSSPAY (in descending order) within SEX.

```
PROC SORT DATA=PAYROLL OUT=SORTPAY;
   BY SEX DESCENDING GROSSPAY;
PROC PRINT;
   TITLE 'PAYROLL FILE';
   FOOTNOTE 'SORTED BY SEX AND DESCENDING GROSS PAY';
```

PAYROLL FILE

OBS	DEPT	NAME	NUMBER	SEX	NETPAY	GROSSPAY
1	917	POWELL	8123	F	789.39	1271.54
2	917	DONNER	10554	F	669.06	972.29
3	916	SHIRES	10052	F	487.12	729.22
4	915	LARSON	11357	F	265.47	383.92
5	915	WOOD	8113	F	238.17	372.24
6	916	SMITH	6237	F	207.09	345.88
7	917	HIGGINS	11658	M	777.50	1235.46
8	917	HERZOG	6433	M	692.57	1045.26
9	915	MOORE	12649	M	557.18	728.12
10	915	HELMS	3658	M	479.26	701.26
11	917	TALL	11931	M	355.19	492.26
12	916	RICHARDS	5781	M	319.27	472.84
13	915	RYAN	10961	M	291.56	399.20
14	916	MURPHY	6927	M	272.66	385.11
15	917	EVANS	7716	M	224.36	310.40

SORTED BY SEX AND DESCENDING GROSS PAY

BY-group Processing

When a BY statement is used with a procedure,

- the procedure processes each BY group separately
- the data must be grouped by the BY-group variables.

Example: Print the PAYROLL data set with the employees grouped by department.

```
PROC SORT DATA=PAYROLL;
   BY DEPT;
PROC PRINT;
   BY DEPT;
   TITLE 'PAYROLL FILE';
   FOOTNOTE 'GROUPED BY DEPARTMENT';
```

BY-group Processing

PAYROLL FILE

DEPT=915

OBS	NAME	NUMBER	SEX	NETPAY	GROSSPAY
1	HELMS	3658	M	479.26	701.26
2	LARSON	11357	F	265.47	383.92
3	MOORE	12649	M	557.18	728.12
4	RYAN	10961	M	291.56	399.20
5	WOOD	8113	F	238.17	372.24

DEPT=916

OBS	NAME	NUMBER	SEX	NETPAY	GROSSPAY
6	MURPHY	6927	M	272.66	385.11
7	RICHARDS	5781	M	319.27	472.84
8	SHIRES	10052	F	487.12	729.22
9	SMITH	6237	F	207.09	345.88

DEPT=917

OBS	NAME	NUMBER	SEX	NETPAY	GROSSPAY
10	DONNER	10554	F	669.06	972.29
11	EVANS	7716	M	224.36	310.40
12	HERZOG	6433	M	692.57	1045.26
13	HIGGINS	11658	M	777.50	1235.46
14	POWELL	8123	F	789.39	1271.54
15	TALL	11931	M	355.19	492.26

GROUPED BY DEPARTMENT

4.4 Exercises

4.1 A small department store collects selected information on each of its sales and records the data on cards. The record layout and 16 sample records are shown below.

Field Description	Start Position	Field Width	Variable Type
SALESPERSON	1	5	character
DEPARTMENT	6	8	character
COST OF ITEM	14	5	numeric

```
Record                                          Field
Number      |...+....1....+....2  <------       Position

     1      SMITHCLOTHING19.28
     2      SMITHTOYS     6.74
     3      JONESTOYS     3.72
     4      ASHLYTOYS    15.83
     5      JONESCLOTHING21.92
     6      SMITHTOYS     6.97
     7      JONESCLOTHING11.58
     8      SMITHCLOTHING12.76
     9      SMITHCLOTHING15.49
    10      ASHLYTOYS     5.45
    11      ASHLYCLOTHING15.90
    12      ASHLYTOYS    17.93
    13      JONESCLOTHING 9.04
    14      SMITHTOYS    19.29
    15      SMITHCLOTHING12.67
    16      ASHLYCLOTHING16.65
```

Exercises

4.1 Write a complete SAS job that will solve all of the problems listed below.

a. Prepare a SAS data set named SALES from the raw data on the preceding page.

The data set SALES should contain the following variables:

CLERK - salesperson

DEPT - department

COST - cost of item sold

b. Print the data set.

c. Print only the variables CLERK and COST in the order listed and suppress the observation number.

d. Obtain a listing of the data set such that the clerks are listed in alphabetical order and the observations are arranged from largest cost to smallest cost for each clerk.

e. Repeat part (d) except put each clerk on a separate page.

f. Print the data set grouped by clerk, display the total sales for each clerk, and display the total of all sales.

Exercises

4.2 A small town in North Carolina stores its water and sewer billing data on cards. The record layout is shown below.

Field Description	Field Location	Variable Type
ACCOUNT NUMBER	1-8	character
PREVIOUS READING	9-14	numeric
PRESENT READING	15-20	numeric
MONTH OF READING	21-23	character
YEAR OF READING	24-27	numeric
COST OF WATER	28-32	numeric

a. Create a SAS data set named WATER with the following variables:

Variable Name	Field Description	Variable Type
ACCOUNT	ACCOUNT NUMBER	character
PAST	PREVIOUS READING	numeric
PRESENT	PRESENT READING	numeric
MONTH	MONTH OF READING	character
YEAR	YEAR OF READING	numeric
COST	COST OF WATER	numeric

Exercises

4.2 **b.** Print the data set with the observations arranged by account number. Suppress the observation number.

c. Print only the account number and cost for each customer. Double-space the output and suppress the observation number.

d. Print only the year, month, and cost of the water for each customer. Suppress the observation number; show the number of observations in the data set at the end of the listing; and compute the total cost of the water used by all residents.

5. Data Transformations

5.1 Creating Variables and Editing Values

5.2 Conditional Execution of SAS Statements

5.3 Lengths of Character Variables

5.4 Exercises

5.1 Creating Variables and Editing Values

Assignment Statements

Assignment statements are used to create new variables and to modify values of existing variables.

General form of an assignment statement:

variable=*expression*;

The SAS System evaluates the expression and assigns the resulting value to the variable.

Assignment Statements

Example: Read three variables (YEAR, REVENUE, and EXPENSES) into a SAS data set.

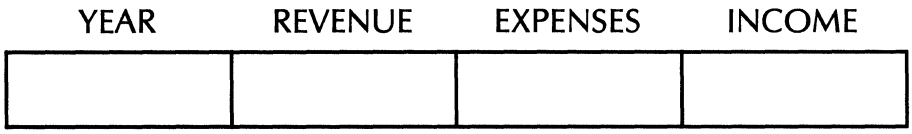
Create a variable named INCOME, which is the difference between REVENUE and EXPENSES.

Change the values of YEAR from 2 digits to 4 digits.

```
DATA PROFITS;
   INPUT YEAR REVENUE EXPENSES;
   INCOME=REVENUE-EXPENSES;
   YEAR=YEAR+1900;
   CARDS;
80 5650 1050
81 6280 1140
;
PROC PRINT;
```

OBS	YEAR	REVENUE	EXPENSES	INCOME
1	1980	5650	1050	4600
2	1981	6280	1140	5140

Program Data Vector



Note: any variable defined by an assignment statement is included in the program data vector.

Types of Expressions

- simple arithmetic operations: + − * / **

```
SUM = X + Y;        addition
DIF = X − Y;        subtraction
TWICE = X*2;        multiplication
HALF = X / 2;       division
CUBIC = X**3;       exponentiation
Y = − X;            change the sign
```

- constants

```
N=0;                numeric constant
SEX='FEMALE';       character constant
```

- complex expressions

priority of evaluation ( ) ** * / + −

```
A = X + Y + Z;      left to right
A = X + Y*Z;        operator precedence
A = X / Y / Z;      left to right
A = X / (Y / Z);    parenthetical
```

- functions

variable=functionname(*argument1, argument2, . . .*);

```
S = SQRT(X);
A= ABS(X);
Z= ABS(SQRT(X)−2);
```

SAS Functions

SAS functions fall into the following categories:

- arithmetic functions
- truncation functions
- mathematical functions
- trigonometric and hyperbolic functions
- probability functions
- sample statistic functions
- random number functions
- character functions
- date and time functions
- state and ZIP code functions
- system functions
- special functions.

SAS Functions

Truncation Functions

ROUND rounds values to the nearest round-off unit.

CEIL returns the smallest integer greater than or equal to the argument.

INT yields the integer portion of the argument.

Example: Compare the results of the ROUND, CEIL, and INT functions.

```
DATA CHGNUM;
   INPUT X;
   TENTHS=ROUND(X,.1);
   OVER=CEIL(X);
   INTEGER=INT(X);
   CARDS;
326.56
 98.2
1401.73
;
PROC PRINT DATA=CHGNUM;
```

OBS	X	TENTHS	OVER	INTEGER
1	326.56	326.6	327	326
2	98.20	98.2	99	98
3	1401.73	1401.7	1402	1401

Note: see the *SAS User's Guide: Basics* for further discussion.

SAS Functions

Selected functions that compute sample statistics:

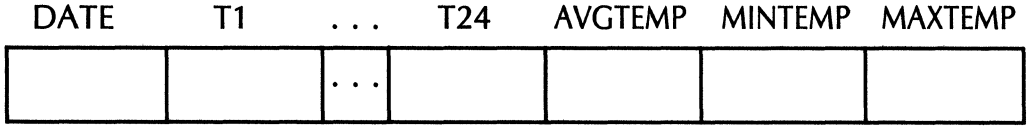
SUM	sum
MEAN	arithmetic mean
VAR	variance
MIN	minimum value
MAX	maximum value
STD	standard deviation

Example: The temperature at a specific location is recorded every hour on the hour for several days. Each record in a file represents one day and contains the date and the 24 recorded temperatures for that date.

Create a SAS data set that contains the date, the 24 hourly temperatures, the average temperature, the minimum temperature, and the maximum temperature for each day.

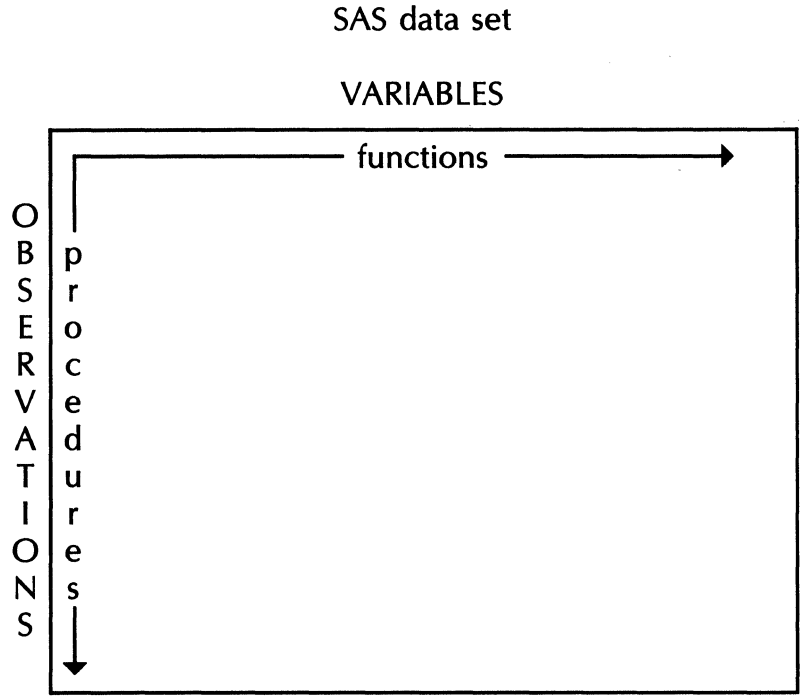
```
DATA TEMP;
   INPUT DATE $ 1-7 @11 (T1-T24) (2.);
   AVGTEMP=MEAN(OF T1-T24);
   MINTEMP=MIN(OF T1-T24);
   MAXTEMP=MAX(OF T1-T24);
   CARDS;

data lines
```

Program Data Vector



Sample Statistics: Functions vs. Procedures

- The sample statistics functions produce statistics for each observation (row) in the SAS data set (functions operate across rows).
- Procedures produce sample statistics for variables (columns) in the SAS data set (procedures operate down columns).



Accumulate Totals

The RETAIN Statement

All variables in the program data vector are normally reset to missing before each execution of the DATA step.

Use a RETAIN statement to

- retain variable values from the previous execution of the DATA step
- give initial values to the variables.

General form of the RETAIN statement:

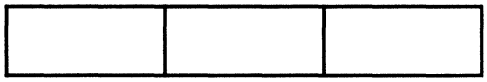
RETAIN *variables initial value . . .*;

Accumulate Totals

Example: Use RETAIN and assignment statements to accumulate totals and count observations.

```
DATA ADD;
   RETAIN COUNT 0 TOTAL 0;
   INPUT SCORE;
   COUNT=COUNT+1;
   TOTAL=TOTAL+SCORE;
   CARDS;
10
 5
 3
 7
 .
 6
 4
;
PROC PRINT;
```

Program Data Vector



OBS	COUNT	TOTAL	SCORE
1	1	10	10
2	2	15	5
3	3	18	3
4	4	25	7
5	5	.	.
6	6	.	6
7	7	.	4

Note: if any score is missing, the value of TOTAL is set to missing and remains missing for all subsequent observations.

Accumulate Totals with the SUM Function

Example: Use the SUM function to avoid the problem of summing variables that may have missing values.

```
DATA ADD;
   RETAIN COUNT 0 TOTAL 0;
   INPUT SCORE;
   COUNT=COUNT+1;
   TOTAL=SUM(TOTAL,SCORE);
   CARDS;
10
 5
 3
 7
 .
 6
 4
;
PROC PRINT;
```

Program Data Vector

OBS	COUNT	TOTAL	SCORE
1	1	10	10
2	2	15	5
3	3	18	3
4	4	25	7
5	5	25	.
6	6	31	6
7	7	35	4

Accumulate Totals with the Sum Statement

The sum statement is a special assignment statement that accumulates values from one observation to the next.

General form of the sum statement:

variable + *expression;*

This *expression* is evaluated, and the result is added to the current value of the *variable*.

The *variable* must have a valid SAS variable name. This variable will contain the accumulated value.

The sum statement

- retains the values of the created variable
- treats a missing value as zero.

Accumulate Totals with the Sum Statement

Example: Use the sum statement to accumulate totals and count observations.

```
DATA ADD;
   INPUT SCORE;
   COUNT+1;
   TOTAL+SCORE;
   CARDS;
10
 5
 3
 7
 .
 6
 4
;
PROC PRINT;
```

Program Data Vector

SCORE	COUNT	TOTAL

OBS	SCORE	COUNT	TOTAL
1	10	1	10
2	5	2	15
3	3	3	18
4	7	4	25
5	.	5	25
6	6	6	31
7	4	7	35

5.2 Conditional Execution of SAS Statements

IF-THEN and ELSE Statements

Use the IF-THEN statement when you want to execute a SAS statement conditional on some expression.

General form of IF-THEN and ELSE statements:

IF *expression* **THEN** *statement;*
ELSE *statement;*

Comparison Operators

There are eight operators available to make comparisons simple.

LT	<	less than
GT	>	greater than
EQ	=	equal to
LE	<=	less than or equal to
GE	>=	greater than or equal to
NE	¬=	not equal to
NL	¬<	not less than
NG	¬>	not greater than

Example: Create a SAS variable that indicates a failing grade when a test score is less than 68.

```
DATA QUIZ;
   INPUT SCORE;
   IF SCORE < 68 THEN GRADE='FAIL';
   ELSE GRADE='PASS';
CARDS;
```

Note: the NL and NG operators are not available in minicomputer environments.

The ^ is used in place of the ¬ in minicomputer environments.

Comparison Operators

Numeric Comparison

Example: Recode a response on a questionnaire that was coded:

1=GOOD 2=FAIR 3=POOR.

```
DATA RECODE;
   INPUT CODE;
   IF CODE=1 THEN RESPONSE='GOOD';
   IF CODE=2 THEN RESPONSE='FAIR';
   IF CODE=3 THEN RESPONSE='POOR';
CARDS;
```

For efficiency, use ELSE statements:

```
DATA RECODE;
   INPUT CODE;
   IF CODE=1 THEN RESPONSE='GOOD';
   ELSE IF CODE=2 THEN RESPONSE='FAIR';
   ELSE IF CODE=3 THEN RESPONSE='POOR';
CARDS;
```

What effect would a missing value for the variable CODE have in the two examples above?

Comparison Operators

Character Comparison

Example: Recode the character variable SEX on the CLASS data set.

```
DATA CLASS;
   INPUT NAME $ SEX $ AGE HEIGHT WEIGHT;
   IF SEX='M' THEN SEX='MALE';
   ELSE SEX='FEMALE';
CARDS;
```

What effect would a missing value for the variable SEX have in the example above?

Logical Operators

OR	|	or, either
AND	&	and, both
NOT	¬	not, negation

OR operator

if either comparison operator is true, then the result of the logical operator is equal to 1 or true.

```
IF STATE='NC' OR STATE='SC'
   THEN REGION='SOUTHEAST';
```

AND operator

if both comparison operators are true simultaneously, then the result of the logical operator is equal to 1 or true.

```
IF STATE='NC' AND CITY='RALEIGH'
   THEN SALESREP='HARVEY LESTER';
```

The statement

```
IF 80<=SCORE<=90 THEN GRADE='B';
```

is equivalent to the statement

```
IF 80<=SCORE & SCORE<=90 THEN GRADE='B';
```

NOT operator

if the comparison operator is false, then the result of the logical operator is equal to 1 or true.

```
IF ¬(STATUS='MARRIED') THEN STATUS='SINGLE';
```

DO and END Statements

Execution of a DO statement specifies that all statements between the DO statement and its matching END statement are to be executed.

DO and END statements can be used to execute a group of statements when a condition is met.

IF *expression* **THEN**
  DO;
    executable statements
  END;

DO and END Statements

Example: The variables NAME, DEPTNO, COM, and SALARY are recorded for a group of employees. If the department number is 201, then define the department name to be SALES and set gross pay equal to salary plus commission. Otherwise, the department name is ADMIN and the gross pay is simply equal to salary.

```
DATA EMPLOY;
   INPUT NAME $ 1-8 DEPTNO 10-12
         COM 14-17 SALARY 19-23;
   IF DEPTNO=201 THEN
      DO;
         DEPT='SALES';
         GROSSPAY=COM+SALARY;
      END;
   ELSE
      DO;
         DEPT='ADMIN';
         GROSSPAY=SALARY;
      END;
   CARDS;
JOHNSON  201 1500 18000
MOSSER   101      21000
LARKIN   101      24000
GARRETT  201 4800 18000
;
PROC PRINT;
```

DO and END Statements

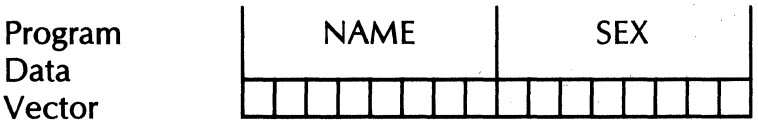
OBS	NAME	DEPTNO	COM	SALARY	DEPT	GROSSPAY
1	JOHNSON	201	1500	18000	SALES	19500
2	MOSSER	101	.	21000	ADMIN	21000
3	LARKIN	101	.	24000	ADMIN	24000
4	GARRETT	201	4800	18000	SALES	22800

5.3 Lengths of Character Variables

Length Defined with an INPUT Statement

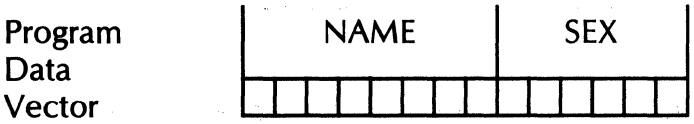
- List input: default length=8 bytes

```
INPUT NAME $ SEX $;
```



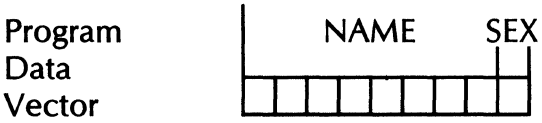
- Column input: length=the width of the input field

```
INPUT NAME $ 1-8 SEX $ 10-15;
```



- Formatted input: length=the width of the input field

```
INPUT NAME $8. SEX $1.;
```



Length Defined with an Assignment Statement

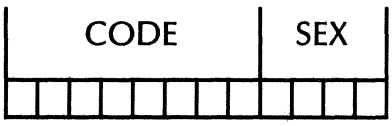
The length of a character variable defined by an assignment statement is equal to the number of characters in the first value listed for the variable.

Example: Recode the values for a variable that was coded:

1=MALE 2=FEMALE.

```
DATA RECODE;
   INPUT CODE;
   IF CODE=1 THEN SEX='MALE';
   ELSE SEX='FEMALE';
CARDS;
```

Program Data Vector



Note: the value FEMALE would be truncated to FEMA in this example.

Length Defined with a LENGTH Statement

You can use the LENGTH statement to define the length of a variable.

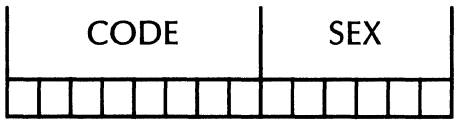
General form of the LENGTH statement:

LENGTH *variable list* **$** *length;*

Example: Solve the truncation problem with a LENGTH statement.

```
DATA RECODE;
   INPUT CODE;
   LENGTH SEX $ 6;
   IF CODE=1 THEN SEX='MALE';
   ELSE SEX='FEMALE';
CARDS;
```

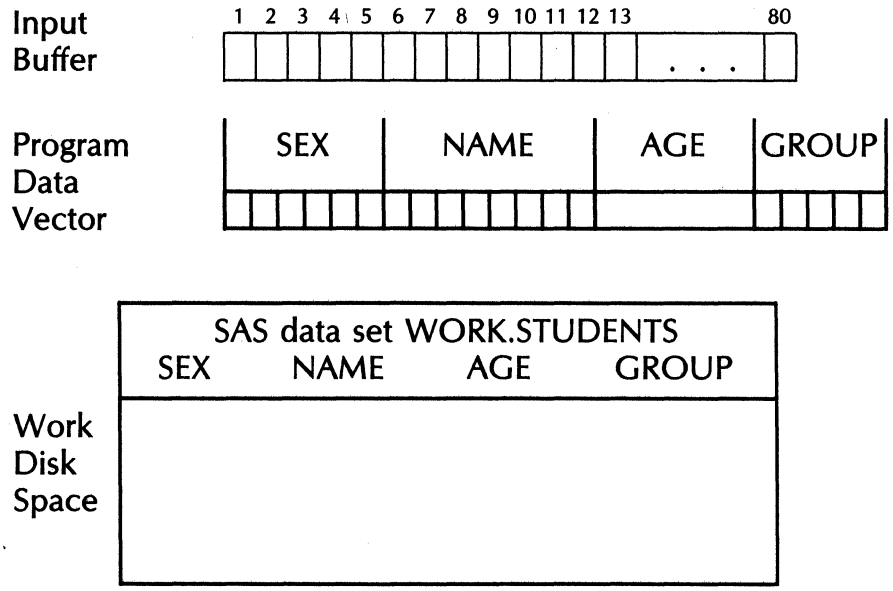
Program Data Vector



Defining Lengths of Character Variables

Example: Conditionally edit the values of the variable SEX and assign values to the variable GROUP based on the values of the variable AGE.

```
DATA STUDENTS;
   LENGTH SEX $ 6;
   INPUT NAME $ SEX $ 10 AGE 12-13;
   IF SEX='M' THEN SEX='MALE';
   ELSE SEX='FEMALE';
   IF AGE < 13 THEN GROUP='CHILD';
   ELSE GROUP='TEEN';
   CARDS;
JEFF      M 14
KAREN     F  9
SHARON    F 17
TIMOTHY   M 16
;
PROC CONTENTS;
PROC PRINT;
```



PROC CONTENTS Output

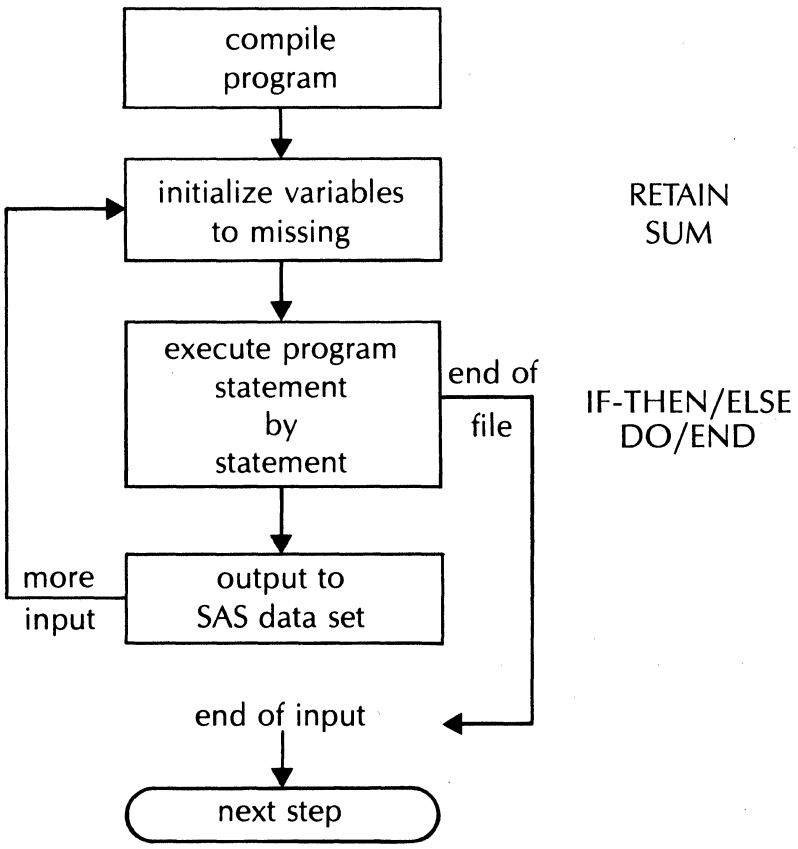
```
                          CONTENTS PROCEDURE
                    CONTENTS OF SAS MEMBER WORK.STUDENTS


CREATED BY TSO USERID EDU       ON CPUID 00-0000-000000
AT 14:39 MONDAY, FEBRUARY 3, 1986   BY SAS RELEASE 5.XX
DSNAME=SYS86034.T143944.RA000.EDU.R0000048
OBSERVATIONS PER TRACK =615   BLKSIZE=19069   LRECL=31
GENERATED BY DATA
NUMBER OF OBSERVATIONS: 4   NUMBER OF VARIABLES: 4
MEMTYPE:   DATA

          ----ALPHABETIC LIST OF VARIABLES AND ATTRIBUTES-----
       # VARIABLE TYPE   LENGTH POSITION FORMAT       INFORMAT     LABEL
       3 AGE      NUM         8       18
       4 GROUP    CHAR        5       26
       2 NAME     CHAR        8       10
       1 SEX      CHAR        6        4


------------------------------ SOURCE RECORDS -------------------------
| DATA STUDENTS;
|    LENGTH SEX $ 6;
|    INPUT NAME $ SEX $ 10 AGE 12-13;
|    IF SEX='M' THEN SEX='MALE';
|    ELSE SEX='FEMALE';
|    IF AGE < 13 THEN GROUP='CHILD';
|    ELSE GROUP='TEEN';
|    CARDS;
-----------------------------------------------------------------------
```

PROC PRINT Output

OBS	SEX	NAME	AGE	GROUP
1	MALE	JEFF	14	TEEN
2	FEMALE	KAREN	9	CHILD
3	FEMALE	SHARON	17	TEEN
4	MALE	TIMOTHY	16	TEEN

Review of DATA Step Logic

- The statements in a DATA step form a program.



- The SAS System compiles the program into machine code and then executes it.
- The SAS System executes each program statement in sequence until the end of the program.
- The SAS System repeats the program execution for each observation in the input data.

5.4 Exercises

5.1 The National Weather Service keeps records of temperature and precipitation data for each state. Data collected for NC and SC over a period of one year have been stored on cards with a record format described in the table below.

Variable Name	Field Location	Variable Type
STATE	1-2	character
MONTH	3-5	character
RAIN	6-10	numeric
MAXTEMP	11-15	numeric
MINTEMP	16-20	numeric
MEANTEMP	21-25	numeric

The rainfall data are recorded in inches, and the temperature data are recorded in degrees Fahrenheit. Given the above information, write a complete SAS DATA step to solve parts (a), (b), and (c) below.

a. Create a SAS data set that contains all of the input variables listed above.

b. Create a variable that contains the difference between the minimum and maximum temperatures.

c. Create the two variables described on the next page.

d. Print the data set.

Exercises

5.1

Variable	Description
PRECIP	a character variable with 2 possible values:
DRY	months in which the rainfall for a state is less than 2 inches
WET	all other months.
TEMP	a character variable with 3 possible values:
HOT	months in which the MEANTEMP for a state exceeds 80 degrees F
MILD	months in which the MEANTEMP for a state is greater than 40 degrees F but less than or equal to 80 degrees F
COLD	months in which the MEANTEMP for a state is less than or equal to 40 degrees F.

5.2 Write one DATA step that solves all the parts to problem 5.1 except express all temperature data in degrees Celsius and print the data set.

Note: C=5*(F−32)/9 where C = degrees Celsius
F = degrees Fahrenheit

Exercises

5.3 Jalopy Rent-A-Car computes its bills by recording the following information for each car rental:

- Total number of days (rounded to tenths of a day) the customer kept the car (DAYS)
- Daily charge rate for the car (DAYRATE)
- Miles driven (MILES)
- Mileage charge in cents per mile (MILERATE)
- Discount rate allowed for the customer (DISCRATE)
- Complete collision insurance: 1=YES/0=NO (INSURE).

The rental information is stored on data cards. The values from left to right on each card represent the variables DAYS, DAYRATE, MILES, MILERATE, DISCRATE and INSURE, respectively (assume there is at least one blank between each field). The bill is computed by using the information above and by taking into consideration the following factors:

Exercises

5.3

- The rental charge (RENTCOST) is equal to the number of days the customer kept the car multiplied by the daily charge rate.
- The minimum rental charge (RENTCOST) is the rental charge for one day.
- The mileage cost (MILECOST) is equal to the number of miles driven multiplied by the mileage rate.
- Complete collision insurance costs (NSURCOST) $6.00 per day. For the insurance cost, a fraction of a day is treated as a whole day.
- The customer discount (DISCOUNT) is based on the DISCRATE and is only applied to the mileage and rental costs.
- The subtotal (SUBTOTAL) is equal to the rental charge plus the mileage cost plus the insurance cost minus the discount.
- 6% sales tax (TAX) is added to the bill.
- The total cost (TOTAL) is the subtotal plus the tax.

Write a SAS DATA step that will compute the bill and create a SAS data set with the following variables: DAYS, DAYRATE, MILES, MILERATE, DISCRATE, INSURE, RENTCOST, MILECOST, NSURCOST, DISCOUNT, SUBTOTAL, TAX, and TOTAL. Print the data set.

6. Errors and Missing Values in SAS Jobs

6.1 Syntax Errors

6.2 Data Errors

6.3 Missing Values

6.4 The HELP Facility

6.5 Exercises

6.1 Syntax Errors

Detecting Syntax Errors

When errors like misspelling SAS keywords, forgetting semicolons, and choosing invalid options are made, the SAS supervisor

- prints the word ERROR, an error number, or both depending upon the operating system
- identifies the location of the error
- prints a message explaining the error at the end of the step.

Note: syntax errors are detected when the step is compiled.

A Program with Syntax Errors

Example: Choose an invalid SAS variable name when creating the SAS data set and misspell a SAS keyword in the same job.

```
OPTIONS LS=72 PS=46;
DATA CLASS;
   INPUT FIRSTNAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;
JOHN       M 12 59.0  99.5
JAMES      M 12 57.3  83.0
ALFRED     M 14 69.0 112.5
WILLIAM    M 15 66.5 112.0
JEFFREY    M 13 62.5  84.0
RONALD     M 15 67.0 133.0
THOMAS     M 11 57.5  85.0
PHILIP     M 16 72.0 150.0
ROBERT     M 12 64.8 128.0
HENRY      M 14 63.5 102.5
JANET      F 15 62.5 112.5
JOYCE      F 11 51.3  50.5
JUDY       F 14 64.3  90.0
CAROL      F 14 62.8 102.5
JANE       F 12 59.8  84.5
LOUISE     F 12 56.3  77.0
BARBARA    F 13 65.3  98.0
MARY       F 15 66.5 112.0
ALICE      F 13 56.5  84.0
;
PROC CONTENTS DATA=CLASS;
PROC PRINT DSTA=CLASS;
```

SAS Log: Mainframe Environments

```
          SAS(R) LOG   OS SAS 5.XX      MVS/XA TSO USER TSOUSER

NOTE: COPYRIGHT (C) 1984 SAS INSTITUTE INC., CARY, N.C.  27511,
U.S.A.
NOTE: SAS RELEASE 5.XX AT SAS INSTITUTE DATA CENTER (00000000)

1          OPTIONS LS=72 PS=46;
2          DATA CLASS;
3             INPUT FIRSTNAME $ 1-8 SEX $ 11 AGE 13-14
                   ---------
                   103
4                  HEIGHT 16-19 WEIGHT 21-25;
5             CARDS;


ERROR 103: A NAME CANNOT HAVE MORE THAN 8 CHARACTERS.

NOTE: SAS STOPPED PROCESSING THIS STEP BECAUSE OF ERRORS.
NOTE: SAS SET OPTION OBS=0 AND WILL CONTINUE TO CHECK STATEMENTS.
      THIS MAY CAUSE NOTE: NO OBSERVATIONS IN DATA SET.
NOTE: DATA SET WORK.CLASS HAS 0 OBSERVATIONS AND 5 VARIABLES. 515 OBS/TR
K
NOTE: THE DATA STATEMENT USED 0.04 SECONDS AND 896K.

25         ;
26         PROC CONTENTS DATA=CLASS;
NOTE: THE PROCEDURE CONTENTS USED 0.08 SECONDS AND 1048K.

27         PROC PRINT DSTA=CLASS;
                      ----
                      1

ERROR 1: SYNTAX ERROR: EXPECTING ONE OF THE FOLLOWING:
      D, DATA, DEBUG, DOUBLE, L, LABEL, N, NOOBS, PAGE, R, ROUND, S,
      SPLIT, U, UNIFORM.

NOTE: SAS STOPPED PROCESSING THIS STEP BECAUSE OF CANCEL/QUIT.
NOTE: THE PROCEDURE PRINT USED 0.07 SECONDS AND 996K.
NOTE: SAS USED 1048K MEMORY.

ERROR: ERRORS ON PAGES 1.

NOTE: SAS INSTITUTE INC.
      SAS CIRCLE
      PO BOX 8000
      CARY, N.C. 27511-8000
```

SAS Log: Minicomputer Environments

```
                    S A S   L O G    VMS SAS 5.XX

Copyright (c) 1985 SAS Institute Inc., Cary, N. C. 27511, U. S. A.
NOTE: VMS Version of SAS Release 5.XX at SAS INSTITUTE INC.
NOTE: LICENSED CPUID MODEL = 8600, SERIAL = 00000000.

     1 OPTIONS LS=72 PS=46;
     2 DATA CLASS;
     3    INPUT FIRSTNAME $ 1-8 SEX $ 11 AGE 13-14
                ----------
                103
103 A NAME CANNOT HAVE MORE THAN 8 CHARACTERS.
     4          HEIGHT 16-19 WEIGHT 21-25;
     5    CARDS;
NOTE: SAS STOPPED PROCESSING THIS STEP BECAUSE OF ERRORS.
NOTE: THE DATA STEP USED 00:00:00.23 CPU SECONDS, 211 PAGEFAULTS
    25 ;
    26 PROC CONTENTS DATA=CLASS;
ERROR: INPUT DATA SET NOT FOUND.
NOTE: THE PROCEDURE CONTENTS USED 00:00:00.16 CPU SECONDS, 279
      PAGEFAULTS
ERROR: SYNTAX ERROR AT WORD DSTA
       AT LINE 27 AT COLUMN 12
ERROR: EXPECTING ONE OF THE FOLLOWING:D, DATA, DOUBLE, L, LABEL, N,
       , NOOBS, PAGE, R, ROUND, S, SPLIT, U, UNIFORM.
    27 PROC PRINT DSTA=CLASS;
NOTE: THE PROCEDURE PRINT USED 00:00:00.05 CPU SECONDS, 38 PAGEFAULTS
NOTE: SAS INSTITUTE INC., SAS CIRCLE, BOX 8000, CARY, N. C., 27511-8000
```

A Program with Syntax Errors

Example: Correct the original two errors and misspell a SAS variable name when invoking a procedure.

```
OPTIONS LS=72 PS=46;
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;
JOHN       M 12 59.0  99.5
JAMES      M 12 57.3  83.0
ALFRED     M 14 69.0 112.5
WILLIAM    M 15 66.5 112.0
JEFFREY    M 13 62.5  84.0
RONALD     M 15 67.0 133.0
THOMAS     M 11 57.5  85.0
PHILIP     M 16 72.0 150.0
ROBERT     M 12 64.8 128.0
HENRY      M 14 63.5 102.5
JANET      F 15 62.5 112.5
JOYCE      F 11 51.3  50.5
JUDY       F 14 64.3  90.0
CAROL      F 14 62.8 102.5
JANE       F 12 59.8  84.5
LOUISE     F 12 56.3  77.0
BARBARA    F 13 65.3  98.0
MARY       F 15 66.5 112.0
ALICE      F 13 56.5  84.0
;
PROC CONTENTS DATA=CLASS;
PROC PRINT DATA=CLASS;
   VARIABLES NAME HEIGTH WEIGHT;
```

SAS Log: Mainframe Environments

```
          SAS(R) LOG   OS SAS 5.XX      MVS/XA TSO USER TSOUSER

NOTE: COPYRIGHT (C) 1984 SAS INSTITUTE INC., CARY, N.C.  27511,
U.S.A.
NOTE: SAS RELEASE 5.XX AT SAS INSTITUTE DATA CENTER (00000000)

1          OPTIONS LS=72 PS=46;
2          DATA CLASS;
3             INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
4                   HEIGHT 16-19 WEIGHT 21-25;
5             CARDS;

NOTE: DATA SET WORK.CLASS HAS 19 OBSERVATIONS AND 5 VARIABLES. 515 OBS/T
RK
NOTE: THE DATA STATEMENT USED 0.05 SECONDS AND 896K.

25         ;
26         PROC CONTENTS DATA=CLASS;
NOTE: THE PROCEDURE CONTENTS USED 0.11 SECONDS AND 1048K
      AND PRINTED PAGE 1.

27         PROC PRINT DATA=CLASS;
28
ERROR: VARIABLE HEIGTH NOT FOUND.
28            VARIABLES NAME HEIGTH WEIGHT;
NOTE: SAS STOPPED PROCESSING THIS STEP BECAUSE OF ERRORS.
NOTE: THE PROCEDURE PRINT USED 0.08 SECONDS AND 1032K.
NOTE: SAS USED 1048K MEMORY.

ERROR: ERRORS ON PAGES 1.

NOTE: SAS INSTITUTE INC.
      SAS CIRCLE
      PO BOX 8000
      CARY, N.C. 27511-8000
```

Procedure Output: Mainframe Environments

```
                        CONTENTS PROCEDURE
                   CONTENTS OF SAS MEMBER WORK.CLASS


CREATED BY TSO USERID EDU       ON CPUID 00-0000-000000
AT 9:16 FRIDAY, FEBRUARY 14, 1986   BY SAS RELEASE 5.XX
DSNAME=SYS86045.T091643.RA000.EDU.R0000052
OBSERVATIONS PER TRACK =515   BLKSIZE=19059   LRECL=37
GENERATED BY DATA
NUMBER OF OBSERVATIONS: 19   NUMBER OF VARIABLES: 5
MEMTYPE:   DATA

          ----ALPHABETIC LIST OF VARIABLES AND ATTRIBUTES-----
       # VARIABLE TYPE   LENGTH POSITION FORMAT       INFORMAT     LABEL
       3 AGE      NUM         8       13
       4 HEIGHT   NUM         8       21
       1 NAME     CHAR        8        4
       2 SEX      CHAR        1       12
       5 WEIGHT   NUM         8       29


------------------------------ SOURCE RECORDS --------------------------
| DATA CLASS;
|    INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
|          HEIGHT 16-19 WEIGHT 21-25;
|    CARDS;
------------------------------------------------------------------------
```

SAS Log: Minicomputer Environments

```
                    S A S   L O G    VMS SAS 5.XX

Copyright (c) 1985 SAS Institute Inc., Cary, N. C. 27511, U. S. A.
NOTE: VMS Version of SAS Release 5.XX at SAS INSTITUTE INC.
NOTE: LICENSED CPUID MODEL = 8600, SERIAL = 00000000.

     1 OPTIONS LS=72 PS=46;
     2 DATA CLASS;
     3    INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
     4          HEIGHT 16-19 WEIGHT 21-25;
     5    CARDS;
NOTE: THE DATA SET WORK.CLASS HAS 19 OBSERVATIONS AND 5 VARIABLES.
NOTE: THE DATA STEP USED 00:00:00.53 CPU SECONDS, 715 PAGEFAULTS
    25 ;
    26 PROC CONTENTS DATA=CLASS;
NOTE: THE PROCEDURE CONTENTS USED 00:00:00.23 CPU SECONDS, 244
      PAGEFAULTS
NOTE: THE PROCEDURE PRINTED PAGE 1.
    27 PROC PRINT DATA=CLASS;
    28    VARIABLES NAME HEIGTH WEIGHT;
ERROR: VARIABLE HEIGTH NOT FOUND.
NOTE: THE PROCEDURE PRINT USED 00:00:00.08 CPU SECONDS, 80 PAGEFAULTS
NOTE: SAS INSTITUTE INC., SAS CIRCLE, BOX 8000, CARY, N. C., 27511-8000
```

Procedure Output: Minicomputer Environments

```
                                  SAS

                          CONTENTS PROCEDURE
                    CONTENTS OF SAS MEMBER WORK.CLASS

NUMBER OF OBSERVATIONS: 19                NUMBER OF VARIABLES: 5
MEMTYPE:                DATA

          ----ALPHABETIC LIST OF VARIABLES AND ATTRIBUTES-----
        # VARIABLE TYPE   LENGTH POSITION FORMAT       INFORMAT     LABEL
        3 AGE      NUM         8        9
        4 HEIGHT   NUM         8       17
        1 NAME     CHAR        8        0
        2 SEX      CHAR        1        8
        5 WEIGHT   NUM         8       25
```

6.2 Data Errors

Detecting Data Errors

Errors in the data occur when

- invalid data are found in a field
- illegal arguments are used in functions
- impossible mathematical operations are requested.

Note: errors in the data are detected when the step is executed.

When errors in the data are encountered, the SAS System

- prints a note that describes the error
- lists the values currently stored in the input buffer
- lists the values currently stored in the program data vector
- continues executing.

Automatic Variables

During every DATA step, the SAS supervisor automatically creates two variables, _N_ and _ERROR_, and places the variables in the program data vector.

_N_ counts the number of times the DATA step has begun executing.

ERROR signals the occurrence of an error in the data during an execution of the DATA step

equals 1 when a data error is encountered

equals 0 when no data error is encountered.

Note: these automatic variables are **not** written to the SAS data set. However, they can be used in programming statements since they are in the program data vector.

A Program with Data Errors

Example: Divide by zero and read character data in a numeric field.

```
OPTIONS LS=72 PS=46;
DATA PRICES;
   INPUT PARTNO 1-4 UNITS 6-7 COST 9-12;
   UNITCOST=COST/UNITS;
   CARDS;
1466 12 1200
1801  0  350
283A  5  125
3044  3  240
;
PROC PRINT;
```

SAS Log: Mainframe Environments

```
          SAS(R) LOG   OS SAS 5.XX      MVS/XA TSO USER TSOUSER

NOTE: COPYRIGHT (C) 1984 SAS INSTITUTE INC., CARY, N.C.  27511,
U.S.A.
NOTE: SAS RELEASE 5.XX AT SAS INSTITUTE DATA CENTER (00000000)

1          OPTIONS LS=72 PS=46;
2          DATA PRICES;
3             INPUT PARTNO 1-4 UNITS 6-7 COST 9-12;
4             UNITCOST=COST/UNITS;
5             CARDS;

NOTE: DIVISION BY ZERO AT LINE 4 COLUMN 17.

RULE:     ----+----1----+----2----+----3----+----4----+----5----+----6

7          1801  0  350
PARTNO=1801 UNITS=0 COST=350 UNITCOST=. _ERROR_=1 _N_=2
NOTE: INVALID DATA FOR PARTNO IN LINE 8 1-4.    3:17
8          283A  5  125
PARTNO=. UNITS=5 COST=125 UNITCOST=25 _ERROR_=1 _N_=3
NOTE: MATHEMATICAL OPERATIONS COULD NOT BE PERFORMED AT THE
      FOLLOWING PLACES. THE RESULT OF THESE OPERATIONS HAVE
      BEEN SET TO MISSING VALUES.
      EACH PLACE IS GIVEN BY: (NUMBER OF TIMES) AT (LINE):(COLUMN).

      1 AT 4:13

NOTE: DATA SET WORK.PRICES HAS 4 OBSERVATIONS AND 4 VARIABLES. 529 OBS/T
RK
NOTE: THE DATA STATEMENT USED 0.05 SECONDS AND 804K.

10         ;

11         PROC PRINT;
NOTE: THE PROCEDURE PRINT USED 0.08 SECONDS AND 796K
      AND PRINTED PAGE 1.
NOTE: SAS USED 804K MEMORY.

NOTE: SAS INSTITUTE INC.
      SAS CIRCLE
      PO BOX 8000
      CARY, N.C. 27511-8000
```

SAS Log: Minicomputer Environments

```
                    S A S   L O G    VMS SAS 5.XX

Copyright (c) 1985 SAS Institute Inc., Cary, N. C. 27511, U. S. A.
NOTE: VMS Version of SAS Release 5.XX at SAS INSTITUTE INC.
NOTE: LICENSED CPUID MODEL = 8600, SERIAL = 00000000.

     1 OPTIONS LS=72 PS=46;
     2 DATA PRICES;
     3    INPUT PARTNO 1-4 UNITS 6-7 COST 9-12;
     4    UNITCOST=COST/UNITS;
     5    CARDS;
NOTE: DIVISION BY ZERO AT LINE 4 COLUMN 23.

RULER:    ----+----1----+----2----+----3----+----4----+----5----+----6
  7       1801  0  350
PARTNO=1801 UNITS=0 COST=350 UNITCOST=. _ERROR_=1 _N_=2
NOTE: INVALID DATA FOR PARTNO AT LINE 8 COLUMN 1-4.
  8       283A  5  125
PARTNO=. UNITS=5 COST=125 UNITCOST=. _ERROR_=1 _N_=3
NOTE: THE DATA SET WORK.PRICES HAS 4 OBSERVATIONS AND 4 VARIABLES.
NOTE: MATHEMATICAL OPERATIONS COULD NOT BE PERFORMED AT THE FOLLOWING
      PLACES.
      THE RESULTS OF THESE OPERATIONS HAVE BEEN SET TO MISSING VALUES.
      EACH PLACE IS GIVEN BY: (NUMBER OF TIMES) AT (LINE):(COLUMN).
      1 AT 4:23.
NOTE: THE DATA STEP USED 00:00:00.62 CPU SECONDS, 809 PAGEFAULTS
    10 ;
    11 PROC PRINT;
NOTE: THE PROCEDURE PRINT USED 00:00:00.16 CPU SECONDS, 270 PAGEFAULTS
NOTE: THE PROCEDURE PRINTED PAGE 1.
NOTE: SAS INSTITUTE INC., SAS CIRCLE, BOX 8000, CARY, N. C., 27511-8000
```

Procedure Output

OBS	PARTNO	UNITS	COST	UNITCOST
1	1466	12	1200	100
2	1801	0	350	.
3	.	5	125	25
4	3044	3	240	80

6.3 Missing Values

Identifying Missing Values

Most collections of data include missing values. For example, data from a survey might contain a missing value because a respondent failed to answer a given question.

Missing Values for Numeric Variables

represented by: blank field or single dot

check for missing:

```
IF RESPONSE=. THEN VALUE='missing';
```

Missing Values for Character Variables

represented by: blank field or single dot

check for missing:

```
IF VALUE=' ' THEN VALUE='missing';
```

Note: SAS procedures treat missing values in various ways, usually excluding missing data from analysis (see the *SAS User's Guide: Basics*).

How Missing Values Are Processed

Example: A variable is set to missing if the input field is blank or occupied by a single period.

```
DATA;
   INPUT ID 1-2 AGE 4-5;
   CARDS;
.  42
11
;
PROC PRINT;
```

OBS	ID	AGE
1	.	42
2	11	.

Example: Variables start out missing and remain missing if not assigned a value.

```
DATA;
   INPUT CODE;
   IF CODE=1 THEN RESPONSE='AGREE';
   CARDS;
1
2
;
PROC PRINT;
```

OBS	CODE	RESPONSE
1	1	AGREE
2	2	

How Missing Values Are Processed

Example: Variables can be assigned missing values.

```
DATA;
   INPUT CODE;
   IF CODE=9 THEN CODE=.;
   CARDS;
1
9
;
PROC PRINT;
```

OBS	CODE
1	1
2	.

Example: Missing values propagate through arithmetic expressions.

```
DATA;
   INPUT CHECKING SAVINGS;
   TOTAL1=CHECKING+SAVINGS;
   TOTAL2=SUM(CHECKING,SAVINGS);
   CARDS;
100 2000
300  .
;
PROC PRINT;
```

OBS	CHECKING	SAVINGS	TOTAL1	TOTAL2
1	100	2000	2100	2100
2	300	.	.	300

How Missing Values Are Processed

Example: Missing values compare as minus infinity.

```
DATA;
   INPUT PAYMENT DUE;
   IF PAYMENT<DUE THEN STATUS='PAST DUE';
   ELSE STATUS='PAID';
   CARDS;
10  20
25  25
.   10
.    0
;
PROC PRINT;
```

OBS	PAYMENT	DUE	STATUS
1	10	20	PAST DUE
2	25	25	PAID
3	.	10	PAST DUE
4	.	0	PAST DUE

6.4 The HELP Facility

The HELP Statement

You can use the HELP statement to obtain additional information about SAS procedures, statements, and other features as well as the status of SAS products at your site.

General form of the HELP statement:

HELP [*keyword*] [*/option*];

where

keyword specifies the name of a SAS procedure, statement, or other feature about which you want to obtain additional information.

HELP statement option:

FS/NOFS specifies whether HELP information is displayed in full-screen format on a display terminal.

Note: to obtain a list of special keywords available on the HELP statement, specify

```
HELP;
```

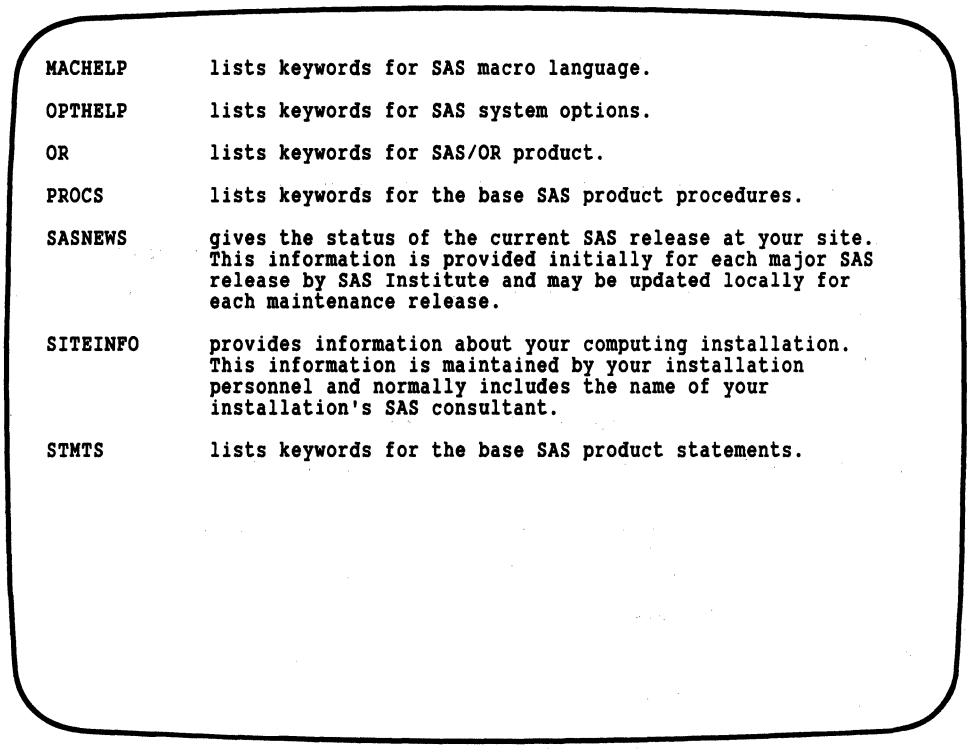
The HELP Statement

Example: Obtain a list of special HELP statement keywords.

```
HELP;
```

```
Help files
Operating systems: All                Scroll forward for more information

To see a list of the available help keywords, see the following help
files:

Keyword        Notes
-------        -----

AF             lists keywords for SAS/AF product.

BASEMISC       lists keywords for the base SAS product miscellaneous
               topics.

DMS            lists keywords for SAS display manager commands.

ETS            lists keywords for SAS/ETS product.

FSP            lists keywords for SAS/FSP product.

FUNCHELP       lists keywords for the base SAS product functions.

GRAPH          lists keywords for SAS/GRAPH product.

IML            lists keywords for SAS/IML product.

IMS            lists keywords for SAS/IMS-DL/I product.
```

The HELP Statement



```
MACHELP      lists keywords for SAS macro language.

OPTHELP      lists keywords for SAS system options.

OR           lists keywords for SAS/OR product.

PROCS        lists keywords for the base SAS product procedures.

SASNEWS      gives the status of the current SAS release at your site.
             This information is provided initially for each major SAS
             release by SAS Institute and may be updated locally for
             each maintenance release.

SITEINFO     provides information about your computing installation.
             This information is maintained by your installation
             personnel and normally includes the name of your
             installation's SAS consultant.

STMTS        lists keywords for the base SAS product statements.
```

The HELP Statement

Example: Obtain information on the COMMENT statement.

```
HELP COMMENT;
```

```
Comment Statement (Use anywhere)     SAS UG: BASICS
Operating systems: All               Scroll forward for more information

The comment statement can be used anywhere in your SAS job to document
the purpose of the job, to explain any unusual segments of the program,
or to describe the steps in a complex program or calculation. The form
of the comment statement is

  *message;

message
explains or documents the job. The message can be any length, although
it cannot contain semicolons.

You can also write comment statements as

  COMMENT message;

You can use any number of comment statements in a job. The comment
statement must end in a semicolon.  Macro variable references and macro
calls are not expanded in SAS comments.  This statement is a comment:

   *THIS CODE FINDS THE NUMBER IN THE BY GROUP;

Comments of the form

  /*message*/

can also be used. They can appear within SAS statements anywhere a
single blank can appear and can contain semicolons. Do not nest comments
of this type, and do not begin them in column 1. (SAS may interpret the
symbols /* in columns 1 and 2 as a request to end the SAS job or
session.)  Comments within a TITLE or FOOTNOTE statement are printed
with the text.  For example, this statement is a valid SAS statement:

   PROC SORT /* SORT THE DATA SET */;

Comments within a title or footnote not surrounded by quotes are
replaced with blanks when the title is printed.
```

The HELP Statement

Example: Use the HELP statement to find out who supports SAS software at your installation.

```
HELP SITEINFO;
```

```
SITEINFO
SAS Licensed Installation Information:

  Site Name  : ABCDWXYZ Corporation Anytown Regional Data Center

       Number: 38471645

Please provide the above information with all documentation
sent to SAS Institute when reporting a suspected problem.

Installation Name and Address:

  Name       : ABCDWXYZ Corporation
  Address    : Anytown Regional Data Center
               97531 Main Street
               PO Box 32760
               Anytown, ST 65536
  Telephone  : 999/234-5678 X1234

SAS Installation Representative (this is the individual to
which SAS Institute directs all communication about products
installed at this site):

  Name       : Name Of Representative
  Telephone  : 999/234-5678 X1234

Requests for problem determination assistance should be
directed to the following individual and/or group:

  Name       : User Services Section
  Telephone  : 999/234-5678 X1234

SAS Institute Program Products (IPPs) Installed at this Site:

  Institute Pro- ------------Licensed Systems----------------
   gram Product  System Name  CPU Type/Model  Serial  Op. System
  ------------  -----------  --------------  ------  ----------
  OS SAS 82.2    A158         3158-3          000001  MVS/SP 1.3
  ------------  ---------------------------------------------
  SAS/GRAPH      A158
  SAS/ETS        A158
  ---------------------------------------------------------------
```

6.5 Exercises

6.1 Given the following three raw data records:

Column 1234567

```
 4   0
10  20
 6   .
```

what will be the values of the variables X, Y and Z in the SAS data sets created by the following DATA steps?

a.
```
DATA; INPUT X Y;
    Z=X/Y;
    CARDS;
```

b.
```
DATA; INPUT X Y;
    Z=X+Y;
    CARDS;
```

c.
```
DATA; INPUT X Y;
    Z=MEAN(X,Y);
    CARDS;
```

d.
```
DATA; INPUT X Y;
    IF X<Y THEN Z='LOSS';
    CARDS;
```

e.
```
DATA; INPUT X Y;
    Z=X-Y;
    IF Z<0 THEN Z=0;
    CARDS;
```

SAS Data Sets

X	Y	Z

7. Summarizing Data

7.1 Computing Descriptive Statistics

7.2 Frequency Counts

7.3 Exercises

7.1 Computing Descriptive Statistics

The MEANS Procedure

The MEANS procedure produces simple univariate descriptive statistics for numeric variables.

Features:

- selected univariate statistics
- special BY-group processing: using a BY statement will cause PROC MEANS to calculate descriptive statistics separately for groups of observations
- optional printed output
- optional output data set of summary statistics

The MEANS Procedure

General form of the PROC MEANS statement:

PROC MEANS *options*;

Options:

DATA=*SASdataset*
names the data set to be analyzed.

NOPRINT
does not print output.

MAXDEC=*n*
uses *n* decimal places to print output.

Specific statistics may be requested:

N	**RANGE**	**CV**	**MAX**
NMISS	**SUM**	**SKEWNESS**	**CSS**
MEAN	**VAR**	**KURTOSIS**	**STDERR**
STD	**USS**	**T**	**PRT**
MIN			

Selected statements used with PROC MEANS:

BY *variable list*;
VAR *variable list*;
OUTPUT OUT=*SASdataset option=variable list. . .*;

Data for PROC MEANS Examples

The PAYROLL data set created in the DATA step below is used for a series of examples in the remainder of this section.

```
DATA PAYROLL;
   INPUT DEPT 1-3 NAME $ 5-12 NUMBER 14-18
         SEX $ 20 NETPAY 22-28 GROSSPAY 30-36;
   CARDS;
914 LARSON   11357 F  265.47  383.92
914 RYAN     10961 M  291.56  399.20
914 JOHNSON  10546 M  435.23  618.72
914 THOMPSON 11584 M  221.09  297.56
915 WOOD      8113 F  238.17  372.24
915 TAYLOR    2943 F  287.22  401.72
915 MOORE    12649 M  557.18  728.12
915 HELMS     3658 M  479.26  701.26
916 SHIRES   10052 F  487.12  729.22
916 RICHARDS  5781 M  319.27  472.84
916 SMITH     6237 F  207.09  345.88
916 MURPHY    6927 M  272.66  385.11
916 FAIRLEY   8846 M  521.66  834.80
917 HERZOG    6433 M  692.57 1045.26
917 TALL     11931 M  355.19  492.26
917 HIGGINS  11658 M  777.50 1235.46
917 DOOB     10554 F  669.06  972.29
918 EPERT     7716 M  224.36  310.40
918 CLANCY    6648 M  245.37  347.12
918 POWELL    8123 F  789.39 1271.54
;
```

Default PROC MEANS Output

Example: Invoke the MEANS procedure to generate default statistics on all numeric variables in the PAYROLL data set.

```
PROC MEANS;
   TITLE 'SIMPLE STATISTICS';
   TITLE2 'ON ALL NUMERIC VARIABLES';
```

SIMPLE STATISTICS
ON ALL NUMERIC VARIABLES

VARIABLE	N	MEAN	STANDARD DEVIATION	MINIMUM VALUE	MAXIMUM VALUE
DEPT	20	915.9000000	1.3726655	914.0000000	918.00000
NUMBER	20	8635.8500000	2820.7494203	2943.0000000	12649.00000
NETPAY	20	416.8210000	194.4725352	207.0900000	789.39000
GROSSPAY	20	617.2460000	311.2411710	297.5600000	1271.54000

Selected PROC MEANS Options

Example: Request specific statistics for selected variables and print the results to 2 decimal places.

```
PROC MEANS MEAN MIN MAX MAXDEC=2;
   VAR NETPAY GROSSPAY;
   TITLE 'SIMPLE STATISTICS';
   TITLE2 'ON SALARY DATA';
```

SIMPLE STATISTICS
ON SALARY DATA

VARIABLE	MEAN	MINIMUM VALUE	MAXIMUM VALUE
NETPAY	416.82	207.09	789.39
GROSSPAY	617.25	297.56	1271.54

BY-group Processing with PROC MEANS

Example: Use the MEANS procedure to generate descriptive statistics on the PAYROLL data set for males and females separately.

```
PROC SORT;
   BY SEX;
PROC MEANS MAXDEC=2;
   BY SEX;
   VAR NETPAY GROSSPAY;
```

SIMPLE STATISTICS
ON SALARY DATA

VARIABLE	N	MEAN	STANDARD DEVIATION	MINIMUM VALUE	MAXIMUM VALUE
SEX=F					
NETPAY	7	420.50	231.98	207.09	789.39
GROSSPAY	7	639.54	364.64	345.88	1271.54
SEX=M					
NETPAY	13	414.84	181.55	221.09	777.50
GROSSPAY	13	605.24	294.03	297.56	1235.46

Storing PROC MEANS Statistics

Example: Create a new SAS data set containing summary statistics by SEX.

Suppress the PROC MEANS printout.

Print the SAS data set.

```
PROC MEANS NOPRINT;
   BY SEX;
   VAR NETPAY GROSSPAY;
   OUTPUT OUT=SUMMARY N=NUMBER MEAN=AVGNET AVGGROSS;
PROC PRINT DATA=SUMMARY;
```

SIMPLE STATISTICS
ON SALARY DATA

OBS	SEX	NUMBER	AVGNET	AVGGROSS
1	F	7	420.503	639.544
2	M	13	414.838	605.239

7.2 Frequency Counts

The FREQ Procedure

The FREQ procedure counts values and produces one-way to *n*-way frequency tables (crosstabulations).

Features:

- distribution of variable values
- combined frequency for two or more variables
- weighted frequency
- measures of association and test statistics for two-way tables
- optional output data set

The FREQ Procedure

General form of the PROC FREQ statement:

PROC FREQ DATA=*SASdataset***;**

You can use the TABLES statement to specify the type of frequency table you want to produce.

General form of the TABLES statement:

TABLES *requests/options***;**

one-way	TABLES SEX;
two-way	TABLES SEX*INCOME;
three-way	TABLES GROUP*SEX*INCOME;

Selected TABLES statement options:

EXPECTED	**NOFREQ**	**NOCUM**	**ALL**
DEVIATION	**NOPERCENT**	**MISSING**	**LIST**
CELLCHI2	**NOROW**	**NOPRINT**	**SPARSE**
CHISQ	**NOCOL**	**OUT=***SASdataset*	

Data for PROC FREQ Examples

The CENSUS data set created in the DATA step below is used in a series of examples in the remainder of this section.

```
DATA CENSUS;
   INPUT AGEGROUP $ 1-8 INCOME 10-14
         SEX $ 16-21 STATUS $ 23-29;
   IF INCOME<=5000 THEN INCOME=0;
   ELSE IF 5000<INCOME<=10000 THEN INCOME=5000;
   ELSE IF 10000<INCOME<=20000 THEN INCOME=10000;
   ELSE IF INCOME>20000 THEN INCOME=20000;
   CARDS;
0 TO 12      0 MALE   SINGLE
0 TO 12      0 MALE   SINGLE
0 TO 12      0 MALE   SINGLE
0 TO 12      0 MALE   SINGLE
0 TO 12      0 MALE   SINGLE
      .
      .
```

The variables and their values are:

INCOME	SEX	STATUS	AGEGROUP
0	MALE	MARRIED	0 TO 12
5000	FEMALE	SINGLE	13 TO 19
10000			20 TO 39
20000			40 TO 99

Note: the CENSUS data set contains 193 observations.

One-way Tables

Example: Generate one-way frequency tables on all variables.

```
PROC FREQ;
```

AGEGROUP	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
0 TO 12	37	19.2	37	19.2
13 TO 19	42	21.8	79	40.9
20 TO 39	38	19.7	117	60.6
40 TO 99	76	39.4	193	100.0

INCOME	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
0	111	57.5	111	57.5
5000	45	23.3	156	80.8
10000	26	13.5	182	94.3
20000	11	5.7	193	100.0

SEX	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
FEMALE	99	51.3	99	51.3
MALE	94	48.7	193	100.0

STATUS	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
MARRIED	85	44.0	85	44.0
SINGLE	108	56.0	193	100.0

One-way Tables

Example: Generate one-way frequency tables on a subset of the variables and suppress the cumulative statistics.

```
PROC FREQ;
   TABLES AGEGROUP SEX STATUS/NOCUM;
```

AGEGROUP	FREQUENCY	PERCENT
0 TO 12	37	19.2
13 TO 19	42	21.8
20 TO 39	38	19.7
40 TO 99	76	39.4

SEX	FREQUENCY	PERCENT
FEMALE	99	51.3
MALE	94	48.7

STATUS	FREQUENCY	PERCENT
MARRIED	85	44.0
SINGLE	108	56.0

Two-way Tables

Example: Generate a two-way table of AGEGROUP by SEX.

```
PROC FREQ;
   TABLES AGEGROUP*SEX;
```

TABLE OF AGEGROUP BY SEX

AGEGROUP / FREQUENCY / PERCENT / ROW PCT / COL PCT	FEMALE	MALE	TOTAL
0 TO 12	21	16	37
	10.88	8.29	19.17
	56.76	43.24	
	21.21	17.02	
13 TO 19	20	22	42
	10.36	11.40	21.76
	47.62	52.38	
	20.20	23.40	
20 TO 39	17	21	38
	8.81	10.88	19.69
	44.74	55.26	
	17.17	22.34	
40 TO 99	41	35	76
	21.24	18.13	39.38
	53.95	46.05	
	41.41	37.23	
TOTAL	99	94	193
	51.30	48.70	100.00

Two-way Tables

Example: Generate a two-way table of STATUS by INCOME and suppress all percentages.

```
PROC FREQ;
   TABLES STATUS*INCOME/NOCOL NOROW NOPERCENT;
```

TABLE OF STATUS BY INCOME

STATUS	INCOME				
FREQUENCY	0	5000	10000	20000	TOTAL
MARRIED	36	26	17	6	85
SINGLE	75	19	9	5	108
TOTAL	111	45	26	11	193

Multiway Tables

Example: Generate a three-way table of SEX by INCOME by STATUS.

```
PROC FREQ;
   TABLES SEX*INCOME*STATUS;
```

TABLE 1 OF INCOME BY STATUS
CONTROLLING FOR SEX=FEMALE

INCOME / STATUS	MARRIED	SINGLE	TOTAL
0	33	41	74
FREQUENCY	33	41	74
PERCENT	33.33	41.41	74.75
ROW PCT	44.59	55.41	
COL PCT	64.71	85.42	
5000	15	5	20
FREQUENCY	15	5	20
PERCENT	15.15	5.05	20.20
ROW PCT	75.00	25.00	
COL PCT	29.41	10.42	
10000	2	0	2
FREQUENCY	2	0	2
PERCENT	2.02	0.00	2.02
ROW PCT	100.00	0.00	
COL PCT	3.92	0.00	
20000	1	2	3
FREQUENCY	1	2	3
PERCENT	1.01	2.02	3.03
ROW PCT	33.33	66.67	
COL PCT	1.96	4.17	
TOTAL	51	48	99
	51.52	48.48	100.00

Multiway Tables

TABLE 2 OF INCOME BY STATUS
CONTROLLING FOR SEX=MALE

INCOME / STATUS	MARRIED	SINGLE	TOTAL
0	3	34	37
	3.19	36.17	39.36
	8.11	91.89	
	8.82	56.67	
5000	11	14	25
	11.70	14.89	26.60
	44.00	56.00	
	32.35	23.33	
10000	15	9	24
	15.96	9.57	25.53
	62.50	37.50	
	44.12	15.00	
20000	5	3	8
	5.32	3.19	8.51
	62.50	37.50	
	14.71	5.00	
TOTAL	34	60	94
	36.17	63.83	100.00

FREQUENCY
PERCENT
ROW PCT
COL PCT

Multiway Tables

Example: Generate a three-way table and use the LIST option.

```
PROC FREQ;
   TABLES SEX*INCOME*STATUS/LIST;
```

SEX	INCOME	STATUS	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
FEMALE	0	MARRIED	33	17.1	33	17.1
FEMALE	0	SINGLE	41	21.2	74	38.3
FEMALE	5000	MARRIED	15	7.8	89	46.1
FEMALE	5000	SINGLE	5	2.6	94	48.7
FEMALE	10000	MARRIED	2	1.0	96	49.7
FEMALE	20000	MARRIED	1	0.5	97	50.3
FEMALE	20000	SINGLE	2	1.0	99	51.3
MALE	0	MARRIED	3	1.6	102	52.8
MALE	0	SINGLE	34	17.6	136	70.5
MALE	5000	MARRIED	11	5.7	147	76.2
MALE	5000	SINGLE	14	7.3	161	83.4
MALE	10000	MARRIED	15	7.8	176	91.2
MALE	10000	SINGLE	9	4.7	185	95.9
MALE	20000	MARRIED	5	2.6	190	98.4
MALE	20000	SINGLE	3	1.6	193	100.0

Storing Frequency Tables

Example: Suppress the normal output from PROC FREQ, store the results in a SAS data set, and print the SAS data set.

```
PROC FREQ;
   TABLES SEX*INCOME/NOPRINT OUT=COUNTS;
PROC PRINT DATA=COUNTS;
```

OBS	SEX	INCOME	COUNT	PERCENT
1	FEMALE	0	74	38.3420
2	FEMALE	5000	20	10.3627
3	FEMALE	10000	2	1.0363
4	FEMALE	20000	3	1.5544
5	MALE	0	37	19.1710
6	MALE	5000	25	12.9534
7	MALE	10000	24	12.4352
8	MALE	20000	8	4.1451

7.3 Exercises

7.1 A small department store collects selected information on each of its sales and records the data on cards. The record layout and 16 sample records are shown below.

Field Description	Field Location	Variable Type
SALESPERSON	1-5	character
DEPARTMENT	6-13	character
COST OF ITEM	14-18	numeric

```
Record                                          Field
Number      1...+....1....+....2  <----------   Position

     1      SMITHCLOTHING19.28
     2      SMITHTOYS     6.74
     3      JONESTOYS     3.72
     4      ASHLYTOYS    15.83
     5      JONESCLOTHING21.92
     6      SMITHTOYS     6.97
     7      JONESCLOTHING11.58
     8      SMITHCLOTHING12.76
     9      SMITHCLOTHING15.49
    10      ASHLYTOYS     5.45
    11      ASHLYCLOTHING15.90
    12      ASHLYTOYS    17.93
    13      JONESCLOTHING 9.04
    14      SMITHTOYS    19.29
    15      SMITHCLOTHING12.67
    16      ASHLYCLOTHING16.65
```

Exercises

7.1 Write a complete SAS job that will solve all of the problems listed below.

a. Prepare a SAS data set named SALES from the raw data on the preceding page.

The data set SALES should contain the following variables:

CLERK - salesperson

DEPT - department

COST - cost of item sold

COM - commission

where

COM= 7.5% of cost for toy sales
COM= 5.0% of cost for clothing sales.

b. Print the data set.

c. Produce a frequency table that displays the number of sales made by each clerk.

d. Produce a two-way frequency table that displays the number of sales made by each clerk within each department.

e. Compute the average, minimum, and maximum cost for all sales.

f. For each clerk, determine the number of sales made, the average cost of the sales, and the total cost of the sales.

Exercises

7.2 A small town in North Carolina stores its water and sewer billing data on cards. The record layout is shown below.

Field Description	Field Location	Variable Type
ACCOUNT NUMBER	1-8	character
PREVIOUS READING	9-14	numeric
PRESENT READING	15-20	numeric
MONTH OF READING	21-23	character
YEAR OF READING	24-27	numeric

a. Create a SAS data set named WATER with the variables shown on the next page.

Exercises

7.2

Variable Name	Description	Variable Type
ACCOUNT	ACCOUNT NUMBER	character
PAST	PREVIOUS READING	numeric
PRESENT	PRESENT READING	numeric
MONTH	MONTH OF READING	character
YEAR	YEAR OF READING	numeric
GALLONS	GALLONS OF WATER USED (present-past=gallons)	numeric
WATER	COST OF WATER (rate=$.58 per 100 gal)	numeric
SEWER	COST FOR SEWER SERVICE (rate=80% of water cost)	numeric
TAX	4% TAX ON WATER & SEWER	numeric
BILL	TOTAL BILL (water+sewer+tax)	numeric

b. Print the data set with the observations arranged by account number. Suppress the observation number.

c. Compute the average water cost, sewer cost, and bill for the town residents.

d. Compute the average, minimum, and maximum water cost, sewer cost, and bill by month.

e. Compute the total sewer cost, water cost, and bill for each resident by year.

Exercises

7.3 The Realtor's Association in a particular city keeps a data file of houses that are on the market. The record layout is described below.

Field Description	Field Location	Variable Type
ADDRESS	1-25	character
STYLE	26-33	character
SQUARE FEET	34-37	numeric
BEDROOMS	38	numeric
BATHS	39	numeric
AGE	40-41	numeric
FIREPLACE (1=YES,0=NO)	42	numeric
BASEMENT (1=YES,0=NO)	43	numeric
GARAGE (1=YES,0=NO)	44	numeric
ASSUMPTION (1=YES,0=NO)	45	numeric
PRICE	46-51	numeric

Write a complete SAS job that solves the problems on the next page.

Exercises

7.3 **a.** Prepare a SAS data set named HOUSES from the raw data file described on the previous page.

The data set HOUSES should contain the following variables:

ADDRESS	- street address of the house
STYLE	- style of house (RANCH, SPLIT, or TWOSTORY)
SQFEET	- heated square feet in the house
BR	- number of bedrooms
BATHS	- number of bathrooms
AGE	- age of house in years
FIRE	- is there a fireplace?
BASE	- is there a basement?
GARAGE	- is there a garage?
ASSUME	- is the loan assumable?
PRICE	- asking price of the house.

b. Print the data set.

c. Obtain a listing that displays the houses in order from least expensive to most expensive.

d. Produce a frequency table that displays the number of houses available in each style.

e. Produce a two-way frequency table that displays the number of houses available in each of the bedroom number-bathroom number combinations.

Exercises

7.3 **f.** Compute the average, minimum, and maximum cost for all houses on the market.

g. Determine the number of houses available and the average cost for each style of house.

7.4 The H&L Insurance Company records the following information on every policy it sells.

Variable Name	Field Description	Field Location	Variable Type
POLICY	policy number	1-10	character
CUSTOMER	customer name	11-35	character
AGENT	agent number	36-40	numeric
BASEPREM	base premium	41-47	numeric
PLAN	payment plan 1=yearly 2=quarterly 3=monthly	48	numeric
TYPE	type of insurance LIFE AUTO HEALTH	49-54	character

a. Create a SAS data set named INSURE that contains all of the variables listed above plus the additional variables described on the next page.

Exercises

7.4

Variable Name	Variable Type	Variable Description
YEARPREM	numeric	The yearly premium paid by the customer. The yearly premium equals the base premium for customers on the yearly payment plan, but the company increases the base premium 5% for customers on the quarterly payment plan and 10% for customers on the monthly payment plan.
PAYMENT	numeric	The amount the customer must pay during each pay period. This will be a yearly, quarterly, or monthly payment depending upon the payment plan.
FEE	numeric	The yearly increase in premium for each customer due to the payment plan.

b. Compute the total base premium, the total yearly premium, and the total gain achieved by offering the customers different payment plans.

Exercises

7.4 **c.** Determine the number of policies sold and the total yearly premium for each type of insurance.

d. Compute the total base premium and total yearly premium by agent.

e. Produce a frequency table that displays the number of customers on each payment plan.

f. Construct a crosstabulation that displays the number of sales by each agent for each type of insurance.

8. Reading and Writing SAS Data Sets

8.1 Writing SAS Data Sets

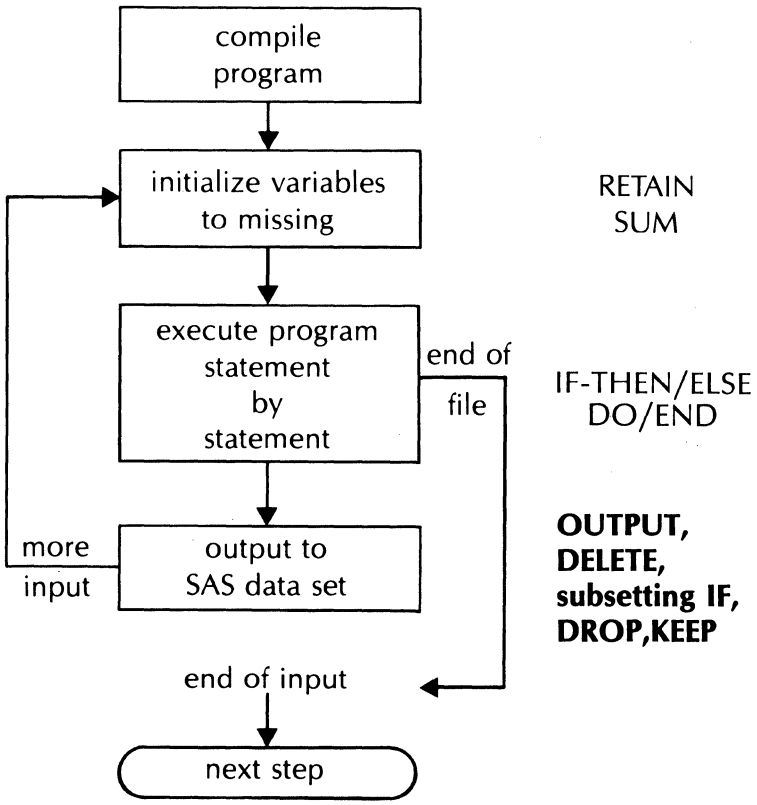
8.2 Reading a SAS Data Set

8.3 Exercises

8.1 Writing SAS Data Sets

Review of Program Logic

- The statements in a DATA step form a program.



- The SAS System compiles the program into machine code and then executes it.
- The SAS System executes each program statement in sequence until the end of the program.
- The SAS System repeats the program execution for each observation in the input data.

Deleting Observations

A DELETE or subsetting IF statement can be used for information retrieval (obtain a subset of the observations).

General form of the DELETE statement:

DELETE;

or

IF *expression* **THEN DELETE;**

The DELETE statement stops the processing of an observation. Control is immediately returned to the beginning of the DATA step without writing the values in the program data vector to the output SAS data set.

Example: Get rid of invalid data.

```
DATA PRODUCT;
   INPUT DEPT UNITS COST;
   IF UNITS<=0 | COST<=0 THEN DELETE;
   UNITCOST=COST/UNITS;
   CARDS;
17   10   525.00
42   50    -6.00
31    5   100.00
24    0     3.00
 5    1    15.00
;
PROC PRINT;
```

OBS	DEPT	UNITS	COST	UNITCOST
1	17	10	525	52.5
2	31	5	100	20.0
3	5	1	15	15.0

Selecting Observations

General form of the subsetting IF statement:

IF *expression*;

The subsetting IF statement is equivalent to:

IF ¬*(expression)* **THEN DELETE;**

The subsetting IF statement indicates which observations to include in the output SAS data set. The statement works like a gate; it allows an observation to pass when the expression is true.

Example: Take a subset of the data.

```
DATA HISTORY;
   INPUT YEAR REVENUE;
   IF YEAR<1970 THEN DELETE;
   CARDS;


DATA HISTORY;
   INPUT YEAR REVENUE;
   IF YEAR>=1970;
   CARDS;
```

Selecting Observations

Example: Take a subset of the data based on the values of a character variable.

```
DATA FORD;
   INPUT MAKE $ YEAR COST;
   IF MAKE='FORD';
   CARDS;
CHEVY 76 2895
FORD  69  700
FORD  72 1750
CHEVY 69 1000
OLDS  73 2995
FORD  79 7100
OLDS  73 1095
FORD  72 1800
;
PROC PRINT;
```

OBS	MAKE	YEAR	COST
1	FORD	69	700
2	FORD	72	1750
3	FORD	79	7100
4	FORD	72	1800

Writing Observations to SAS Data Sets

The OUTPUT statement controls when the values in the program data vector are written to the output SAS data set.

General form of the OUTPUT statement:

OUTPUT;

or

OUTPUT *SASdataset(s)*;

Note: if an OUTPUT statement appears in the DATA step, there is no automatic output at the end of the step.

When an OUTPUT statement is executed, the SAS System immediately outputs the current program data vector values to a SAS data set.

Execution of the OUTPUT statement does not return control to the beginning of the DATA step.

OUTPUT statements can be used to

- create two or more SAS observations from each line of input data
- create multiple SAS data sets from one input file in one DATA step
- create a SAS data set without any input data.

Creating Multiple SAS Data Sets

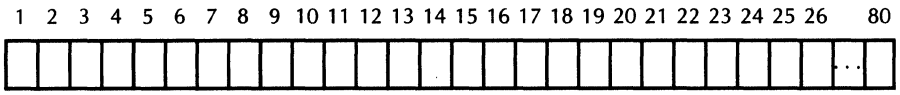
Example: Each record in the input file contains the name, sex, and number of years of education for a subject.

Create one SAS data set that contains subjects with 12 or fewer years of education and another data set that contains subjects with more than 12 years of education.

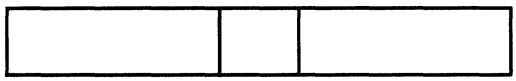
Creating Multiple SAS Data Sets

```
DATA HISCHOOL COLLEGE;
   INPUT NAME $ 1-8 SEX $ 10 YRSEDUC 12-13;
   IF YRSEDUC <= 12 THEN OUTPUT HISCHOOL;
   IF YRSEDUC > 12 THEN OUTPUT COLLEGE;
   CARDS;
KATHRYN  F 16
GEORGE   M 12
WILLIAM  M 18
JENNIFER F 12
CYNTHIA  F 16
;
```

Input Buffer



Program Data Vector



SAS data set HISCHOOL

NAME	SEX	YRSEDUC
GEORGE	M	12
JENNIFER	F	12

SAS data set COLLEGE

NAME	SEX	YRSEDUC
KATHRYN	F	16
WILLIAM	M	18
CYNTHIA	F	16

Selecting Variables

DROP and KEEP statements and data set options control the processing of variables in SAS data sets.

General form of the DROP and KEEP statements:

DROP *variables*;
KEEP *variables*;

General form of the DROP= and KEEP= data set options:

SASdataset(**DROP=***variables*)
SASdataset(**KEEP=***variables*)

DROP and KEEP

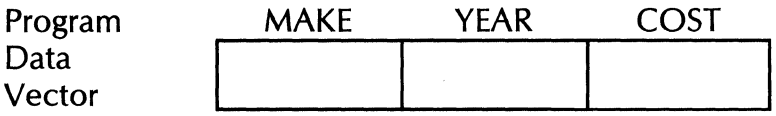
- statements and options are valid in the DATA step and control which variables are stored in the output SAS data set
- options are also valid in the PROC step and are in effect only for the duration of the step
- statements and options are non-executable — they supply information to the SAS System when the step is compiled
- statements can be placed anywhere in a DATA step.

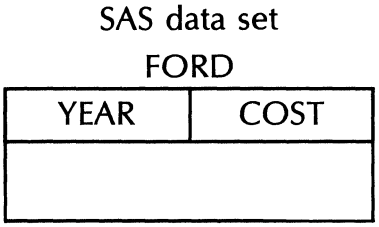
DROP and KEEP Statements

Example: Select only observations where MAKE=FORD.

Drop the variable MAKE from the data set.

```
DATA FORD;
   INPUT MAKE $ YEAR COST;
   DROP MAKE;
   IF MAKE='FORD';
   CARDS;
CHEVY 76 2895
FORD  69  700
FORD  72 1750
CHEVY 69 1000
OLDS  73 2995
FORD  79 7100
OLDS  73 1095
FORD  72 1800
;
```





Write a KEEP statement that is equivalent to the DROP statement in the DATA step above.

DROP and KEEP Statements

Example: Print the data set and identify the make of the selected cars in a title.

```
PROC PRINT;
   TITLE 'LISTING OF AVAILABLE FORDS';
```

```
                    LISTING OF AVAILABLE FORDS

                      OBS    YEAR    COST

                       1      69      700
                       2      72     1750
                       3      79     7100
                       4      72     1800
```

DROP= and KEEP= Data Set Options

Example: Each record in the input file contains the name, sex, and number of years of education for a subject.

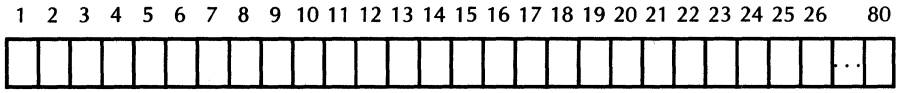
Create one SAS data set (HISCHOOL) that contains subjects with 12 or fewer years of education and another data set (COLLEGE) that contains subjects with more than 12 years of education.

Use the DROP= data set option to drop the variable containing the years of education from the HISCHOOL data set only.

DROP= and KEEP= Data Set Options

```
DATA HISCHOOL(DROP=YRSEDUC) COLLEGE;
   INPUT NAME $ 1-8 SEX $ 10 YRSEDUC 12-13;
   IF YRSEDUC <= 12 THEN OUTPUT HISCHOOL;
   ELSE OUTPUT COLLEGE;
   CARDS;
KATHRYN  F 16
GEORGE   M 12
WILLIAM  M 18
JENNIFER F 12
CYNTHIA  F 16
;
```

Input Buffer



Program Data Vector

NAME	SEX	YRSEDUC

SAS data set HISCHOOL

NAME	SEX
GEORGE	M
JENNIFER	F

SAS data set COLLEGE

NAME	SEX	YRSEDUC
KATHRYN	F	16
WILLIAM	M	18
CYNTHIA	F	16

Write a KEEP= option that is equivalent to the DROP= option in the DATA step above.

8.2 Reading a SAS Data Set

The SET Statement

The SET statement is used to transfer data from an existing SAS data set to the program data vector.

General form of the SET statement:

SET *SASdataset*;

- All variables in the existing SAS data set are automatically passed through the program data vector to the new SAS data set (unless otherwise directed with programming statements).
- All observations in the existing SAS data set are automatically passed through the program data vector to the new SAS data set (unless otherwise directed with programming statements).
- New variables can be created with assignment statements.

Note: **DO NOT** use an INPUT statement to read a SAS data set.

Data for Upcoming Examples

The CLASS data set is used in a series of examples in the remainder of this section.

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;
JOHN       M 12 59.0  99.5
JAMES      M 12 57.3  83.0
  .
  .
  .
PROC PRINT DATA=CLASS;
```

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0
3	ALFRED	M	14	69.0	112.5
4	WILLIAM	M	15	66.5	112.0
5	JEFFREY	M	13	62.5	84.0
6	RONALD	M	15	67.0	133.0
7	THOMAS	M	11	57.5	85.0
8	PHILIP	M	16	72.0	150.0
9	ROBERT	M	12	64.8	128.0
10	HENRY	M	14	63.5	102.5
11	JANET	F	15	62.5	112.5
12	JOYCE	F	11	51.3	50.5
13	JUDY	F	14	64.3	90.0
14	CAROL	F	14	62.8	102.5
15	JANE	F	12	59.8	84.5
16	LOUISE	F	12	56.3	77.0
17	BARBARA	F	13	65.3	98.0
18	MARY	F	15	66.5	112.0
19	ALICE	F	13	56.5	84.0

The SET Statement

Example: Create another SAS data set that contains all the CLASS data set information plus a variable that contains the ratio between height and weight for each student.

Print the new SAS data set.

```
DATA NEWCLASS;
   SET CLASS;
   RATIO=HEIGHT/WEIGHT;
PROC PRINT DATA=NEWCLASS;
```

Compiling the DATA Step

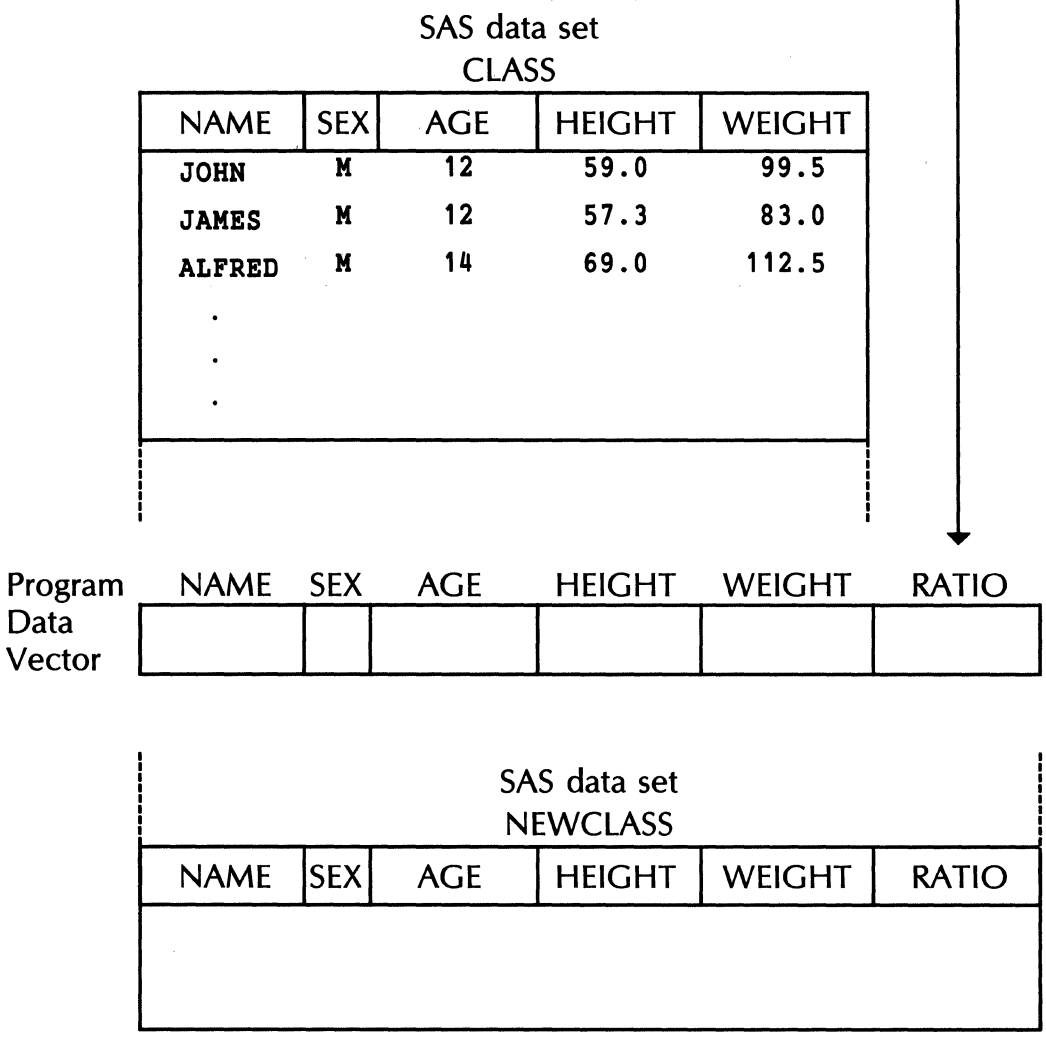
At compile time,

- the SET statement reads the descriptor portion of the existing SAS data set
- a program data vector is created that contains all of the variables found in the existing SAS data set plus any new variables created with assignment statements
- the descriptor portion of the new SAS data set is written.

Note: in minicomputer environments, the descriptor portion is not written to the new SAS data set until the execution phase begins.

Compiling the DATA Step

```
DATA NEWCLASS;
   SET CLASS;
   RATIO=HEIGHT/WEIGHT;
```



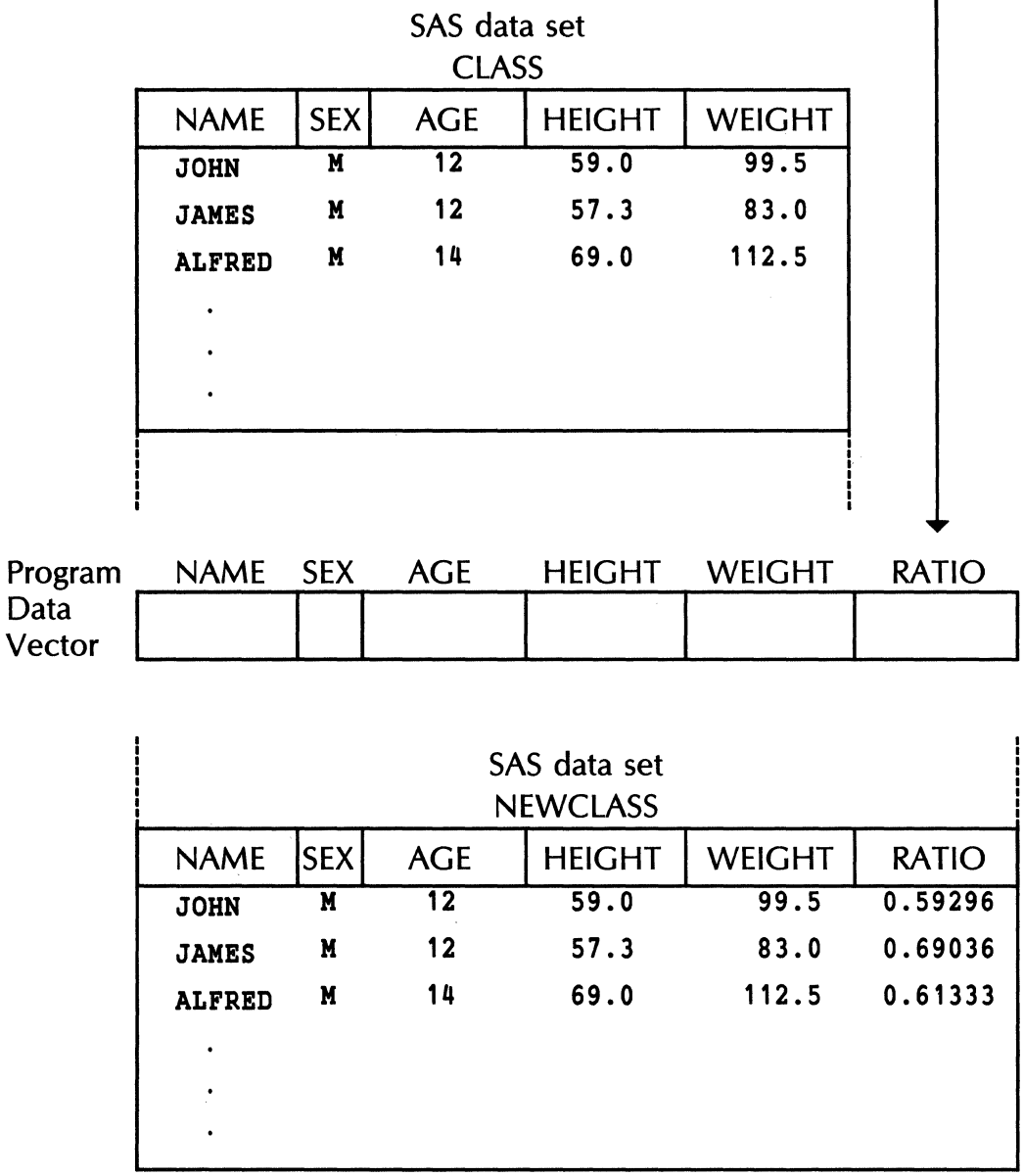
Executing the DATA Step

At execution time,

- the SET statement is executed once for each observation in the existing SAS data set
- each time the SET statement is executed, it reads an observation from the existing SAS data set and writes the current observation in the program data vector
- any program statements in the DATA step are executed for the current observation
- the values in the program data vector are written to the new SAS data set after the last executable statement in the DATA step (or when an OUTPUT statement is executed).

Executing the DATA Step

```
DATA NEWCLASS;
   SET CLASS;
   RATIO=HEIGHT/WEIGHT;
```



Print the Data Set

```
PROC PRINT DATA=NEWCLASS;
```

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT	RATIO
1	JOHN	M	12	59.0	99.5	0.59296
2	JAMES	M	12	57.3	83.0	0.69036
3	ALFRED	M	14	69.0	112.5	0.61333
4	WILLIAM	M	15	66.5	112.0	0.59375
5	JEFFREY	M	13	62.5	84.0	0.74405
6	RONALD	M	15	67.0	133.0	0.50376
7	THOMAS	M	11	57.5	85.0	0.67647
8	PHILIP	M	16	72.0	150.0	0.48000
9	ROBERT	M	12	64.8	128.0	0.50625
10	HENRY	M	14	63.5	102.5	0.61951
11	JANET	F	15	62.5	112.5	0.55556
12	JOYCE	F	11	51.3	50.5	1.01584
13	JUDY	F	14	64.3	90.0	0.71444
14	CAROL	F	14	62.8	102.5	0.61268
15	JANE	F	12	59.8	84.5	0.70769
16	LOUISE	F	12	56.3	77.0	0.73117
17	BARBARA	F	13	65.3	98.0	0.66633
18	MARY	F	15	66.5	112.0	0.59375
19	ALICE	F	13	56.5	84.0	0.67262

The SET Statement

Example: Use the CLASS data set to create another SAS data set that contains only students who are at least 13 years old.

```
DATA TEENS;
   SET CLASS;
   IF AGE >= 13;
PROC PRINT;
```

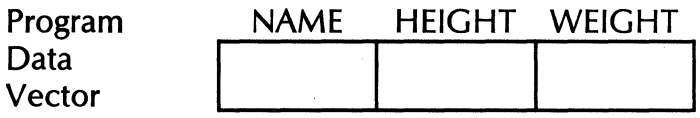
OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	ALFRED	M	14	69.0	112.5
2	WILLIAM	M	15	66.5	112.0
3	JEFFREY	M	13	62.5	84.0
4	RONALD	M	15	67.0	133.0
5	PHILIP	M	16	72.0	150.0
6	HENRY	M	14	63.5	102.5
7	JANET	F	15	62.5	112.5
8	JUDY	F	14	64.3	90.0
9	CAROL	F	14	62.8	102.5
10	BARBARA	F	13	65.3	98.0
11	MARY	F	15	66.5	112.0
12	ALICE	F	13	56.5	84.0

The SET Statement

Example: Use the CLASS data set to create a SAS data set named CONVERT.

Drop the variables AGE and SEX from the CONVERT data set, change the units for HEIGHT from inches to centimeters, and change the units for WEIGHT from pounds to kilograms.

```
DATA CONVERT;
   SET CLASS(DROP=AGE SEX);
   HEIGHT=2.54*HEIGHT;
   WEIGHT=.454*WEIGHT;
```



The SET Statement

```
PROC PRINT DATA=CONVERT;
```

OBS	NAME	HEIGHT	WEIGHT
1	JOHN	149.860	45.173
2	JAMES	145.542	37.682
3	ALFRED	175.260	51.075
4	WILLIAM	168.910	50.848
5	JEFFREY	158.750	38.136
6	RONALD	170.180	60.382
7	THOMAS	146.050	38.590
8	PHILIP	182.880	68.100
9	ROBERT	164.592	58.112
10	HENRY	161.290	46.535
11	JANET	158.750	51.075
12	JOYCE	130.302	22.927
13	JUDY	163.322	40.860
14	CAROL	159.512	46.535
15	JANE	151.892	38.363
16	LOUISE	143.002	34.958
17	BARBARA	165.862	44.492
18	MARY	168.910	50.848
19	ALICE	143.510	38.136

8.3 Exercises

8.1 The Realtor's Association in a particular city keeps a data file of houses that are on the market. The record layout is described below.

Field Description	Field Location	Variable Type
ADDRESS	1-25	character
STYLE	26-33	character
SQUARE FEET	34-37	numeric
BEDROOMS	38	numeric
BATHS	39	numeric
AGE	40-41	numeric
FIREPLACE (1=YES, 0=NO)	42	numeric
BASEMENT (1=YES, 0=NO)	43	numeric
GARAGE (1=YES, 0=NO)	44	numeric
ASSUMPTION (1=YES, 0=NO)	45	numeric
PRICE	46-51	numeric

Write a complete SAS job that will solve all of the problems listed on the next page.

Exercises

8.1 **a.** Prepare a SAS data set named HOUSES from the raw data file described above. Assume the data are in the job stream.

The data set HOUSES should contain the following variables:

ADDRESS	- street address of the house
STYLE	- style of house (RANCH, SPLIT, or TWOSTORY)
SQFEET	- heated square feet in the house
BR	- number of bedrooms
BATHS	- number of bathrooms
AGE	- age of house in years
FIRE	- is there a fireplace?
BASE	- is there a basement?
GARAGE	- is there a garage?
ASSUME	- is the loan assumable?
PRICE	- asking price of the house.

b. Print the data set.

c. Use the HOUSES data set to create another SAS data set that contains only houses that cost $70,000 or less. Print the data set.

d. Use the HOUSES data set to create a different data set for each style of house using only one DATA step. Print each data set with an appropriate title.

Exercises

8.1 **e.** One realtor has three buyers who need listings of houses that contain the features they want. Buyer 1 wants a 3-bedroom 2-bath house, Buyer 2 wants a 4-bedroom 2-bath house for under $90,000, and Buyer 3 wants a 3-bedroom 2-bath house with a fireplace. Use the HOUSES data set to create the three appropriate DATA sets in one DATA step and print each one with an appropriate title.

f. Use one DATA step to create two more SAS data sets from Buyer 3's data set. One data set should contain houses with a garage and the other should contain houses without a garage. Include only the variables ADDRESS and PRICE in the two data sets. Print each data set with an appropriate title.

g. Use one DATA step to create two more SAS data sets from Buyer 2's data set. The first data set should contain houses that cost less than $75,000 and should not include the variables FIRE, BASE, BR, BATHS, and GARAGE. The second data set should contain all other houses in Buyer 2's data set and should not include the variables BR, BATHS, AGE, BASE, and STYLE. Print each data set with an appropriate title.

Exercises

8.2 The Independent Telephone Company in a small town in NC keeps its telephone directory information in a raw data file with the following record format.

Field Description	Field Position	Type of Data
telephone number	1-7	numeric
customer name	8-35	character
street address	36-55	character
ZIP code	56-60	numeric

a. Use the information above to create a SAS data set named DIRECT that contains the variables listed below. Assume the data are in the job stream. Print the data set.

Variable Name	Variable Description
NUMBER	telephone number
EXCHANGE	first three digits of NUMBER (862, 863, or 864)
NAME	customer name
ADDRESS	street address
ZIP	ZIP code (28007 or 28008)

Exercises

8.2 **b.** All of the customers with ZIP=28007 live inside the city limits, and those with ZIP=28008 live outside the city limits. The 862 exchange is for customers inside the city limits and the other two exchanges are for customers outside the city limits. Create a SAS data set that contains customers with incorrect exchanges. Print the data set.

c. Use one DATA step to create one SAS data set for each exchange value. Print each data set with an appropriate title.

8.3 Use the raw data described below to solve the following problems.

Field Description	Field Location	Variable Type
SOCIAL SECURITY NUMBER	1-9	numeric
START DATE OF EMPLOYMENT	10-15	character
ANNUAL SALARY	16-20	numeric
DEPARTMENT NUMBER (DEPT 121, 131, OR 141)	21-23	numeric

Exercises

8.3 **a.** Create a SAS data set named EMPLOYEE from the raw data. The data set should contain the following variables:

SSN	- social security number
START	- start date of employment
SALARY	- annual salary
DEPT	- department number

b. Use the EMPLOYEE data set to create a SAS data set named DEPT121. The data set DEPT121 should only contain information for department 121 employees on the variables SSN, SALARY, and DEPT.

c. Use the SAS data set EMPLOYEE to create a separate SAS data set for each department. Include all four variables in each data set.

d. Use the raw data to create a separate SAS data set for each department and a SAS data set that contains all the departments. The individual department data sets should only contain the variables SSN and SALARY, and the data set that contains all the departments should contain the variables SSN, DEPT, and SALARY.

9. Formatting SAS Output

9.1 Descriptive Variable Labels

9.2 Using SAS Formats

9.3 Creating Formats

9.4 Exercises

9.1 Descriptive Variable Labels

The LABEL Statement

The LABEL statement is used to assign descriptive labels to variables.

General form of the LABEL statement:

LABEL *variable*=*'label'*. . .;

- The LABEL statement associates descriptive labels of up to 40 characters with variable names.
- Many SAS procedures use the variable labels for printing.
- Labels can be stored in the descriptor portion of a SAS data set.
- Labels can be assigned to variables for the duration of a PROC step.

The LABEL Statement in a PROC Step

Example: Use a LABEL statement with the MEANS procedure.

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;

data lines

PROC MEANS N MEAN DATA=CLASS;
   VAR HEIGHT WEIGHT;
   LABEL HEIGHT='HEIGHT IN INCHES'
         WEIGHT='WEIGHT IN POUNDS';
   TITLE 'LABEL STATEMENT IN A PROC STEP';
```

```
                LABEL STATEMENT IN A PROC STEP

 VARIABLE   LABEL                      N          MEAN

 HEIGHT     HEIGHT IN INCHES          19    62.33684211
 WEIGHT     WEIGHT IN POUNDS          19   100.02631579
```

The LABEL Statement in a DATA Step

Example: Use a LABEL statement in a DATA step and process the data set with the MEANS procedure.

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   LABEL HEIGHT='HEIGHT IN INCHES'
         WEIGHT='WEIGHT IN POUNDS';
   CARDS;

data lines

PROC MEANS N MEAN DATA=CLASS;
   VAR HEIGHT WEIGHT;
   TITLE 'LABEL STATEMENT IN A DATA STEP';
```

```
                LABEL STATEMENT IN A DATA STEP

 VARIABLE   LABEL                      N          MEAN

 HEIGHT     HEIGHT IN INCHES          19    62.33684211
 WEIGHT     WEIGHT IN POUNDS          19   100.02631579
```

Note: the labels in this example are stored in the descriptor portion of the data set.

9.2 Using SAS Formats

The FORMAT Statement

The FORMAT statement is used to specify output formats for data values.

General form of the FORMAT statement:

FORMAT *variable list format . . .;*

- The FORMAT statement associates output formats with variables.
- Formats can be stored in the descriptor portion of a SAS data set.
- Formats can be temporarily defined for the duration of a PROC step.

The FORMAT Statement

Selected SAS formats:

BESTw.	SAS chooses best notation
COMMAw.d	commas in numbers (12,683.14)
DOLLARw.d	dollar sign, commas ($12,682.14)
Zw.d	print leading zeros (0038)
ROMANw.	Roman numerals (XIV)
FRACTw.	fractions (3/4)
SSNw.	social security numbers (123-45-6789)
WORDSw.	number in words (TWENTY-THREE)

The FORMAT Statement in a PROC Step

Example: Use a FORMAT statement with the PRINT procedure.

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   CARDS;

data lines

PROC PRINT DATA=CLASS;
   FORMAT NAME $4. AGE ROMAN4. HEIGHT WORDS40.
          WEIGHT 4.;
   TITLE 'FORMAT STATEMENT IN A PROC STEP';
```

The FORMAT Statement in a PROC Step

FORMAT STATEMENT IN A PROC STEP

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	XII	FIFTY-NINE	100
2	JAME	M	XII	FIFTY-SEVEN AND THIRTY HUNDREDTHS	83
3	ALFR	M	XIV	SIXTY-NINE	113
4	WILL	M	XV	SIXTY-SIX AND FIFTY HUNDREDTHS	112
5	JEFF	M	XIII	SIXTY-TWO AND FIFTY HUNDREDTHS	84
6	RONA	M	XV	SIXTY-SEVEN	133
7	THOM	M	XI	FIFTY-SEVEN AND FIFTY HUNDREDTHS	85
8	PHIL	M	XVI	SEVENTY-TWO	150
9	ROBE	M	XII	SIXTY-FOUR AND EIGHTY HUNDREDTHS	128
10	HENR	M	XIV	SIXTY-THREE AND FIFTY HUNDREDTHS	103
11	JANE	F	XV	SIXTY-TWO AND FIFTY HUNDREDTHS	113
12	JOYC	F	XI	FIFTY-ONE AND THIRTY HUNDREDTHS	51
13	JUDY	F	XIV	SIXTY-FOUR AND THIRTY HUNDREDTHS	90
14	CARO	F	XIV	SIXTY-TWO AND EIGHTY HUNDREDTHS	103
15	JANE	F	XII	FIFTY-NINE AND EIGHTY HUNDREDTHS	85
16	LOUI	F	XII	FIFTY-SIX AND THIRTY HUNDREDTHS	77
17	BARB	F	XIII	SIXTY-FIVE AND THIRTY HUNDREDTHS	98
18	MARY	F	XV	SIXTY-SIX AND FIFTY HUNDREDTHS	112
19	ALIC	F	XIII	FIFTY-SIX AND FIFTY HUNDREDTHS	84

The FORMAT Statement in a DATA Step

Example: Use the FORMAT statement in a DATA step and process the data set with the PRINT procedure.

```
DATA CLASS;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   FORMAT NAME $4. AGE ROMAN4. HEIGHT WORDS40.
          WEIGHT 4.;
   CARDS;

data lines

PROC PRINT;
   TITLE 'FORMAT STATEMENT IN A DATA STEP';
```

Note: the formats in this example are stored in the descriptor portion of the data set.

The FORMAT Statement in a DATA Step

FORMAT STATEMENT IN A DATA STEP

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	XII	FIFTY-NINE	100
2	JAME	M	XII	FIFTY-SEVEN AND THIRTY HUNDREDTHS	83
3	ALFR	M	XIV	SIXTY-NINE	113
4	WILL	M	XV	SIXTY-SIX AND FIFTY HUNDREDTHS	112
5	JEFF	M	XIII	SIXTY-TWO AND FIFTY HUNDREDTHS	84
6	RONA	M	XV	SIXTY-SEVEN	133
7	THOM	M	XI	FIFTY-SEVEN AND FIFTY HUNDREDTHS	85
8	PHIL	M	XVI	SEVENTY-TWO	150
9	ROBE	M	XII	SIXTY-FOUR AND EIGHTY HUNDREDTHS	128
10	HENR	M	XIV	SIXTY-THREE AND FIFTY HUNDREDTHS	103
11	JANE	F	XV	SIXTY-TWO AND FIFTY HUNDREDTHS	113
12	JOYC	F	XI	FIFTY-ONE AND THIRTY HUNDREDTHS	51
13	JUDY	F	XIV	SIXTY-FOUR AND THIRTY HUNDREDTHS	90
14	CARO	F	XIV	SIXTY-TWO AND EIGHTY HUNDREDTHS	103
15	JANE	F	XII	FIFTY-NINE AND EIGHTY HUNDREDTHS	85
16	LOUI	F	XII	FIFTY-SIX AND THIRTY HUNDREDTHS	77
17	BARB	F	XIII	SIXTY-FIVE AND THIRTY HUNDREDTHS	98
18	MARY	F	XV	SIXTY-SIX AND FIFTY HUNDREDTHS	112
19	ALIC	F	XIII	FIFTY-SIX AND FIFTY HUNDREDTHS	84

9.3 Creating Formats

The FORMAT Procedure

The FORMAT procedure is used to create user-defined formats.

General form of the PROC FORMAT statement:

PROC FORMAT *options*;

Statements used with PROC FORMAT:

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

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

The FORMAT Procedure

Example: Responses for a 5-item questionnaire were recorded as 1 or 2, where 1=AGREE and 2=DISAGREE.

Recode the responses using the labels 'AGREE' and 'DISAGREE' for the values 1 and 2, respectively.

```
PROC FORMAT;
   VALUE ITEMFMT 1='AGREE'
                 2='DISAGREE';
DATA QUESTION;
   INPUT ID ITEM1-ITEM5;
   CARDS;
1  1  2  1  2  2
2  2  2  2  1  1
3  1  1  2  2  1
;
```

The FORMAT Procedure

Example: Print the data as they were recorded.

```
PROC PRINT;
   TITLE 'QUESTIONNAIRE DATA';
```

QUESTIONNAIRE DATA

OBS	ID	ITEM1	ITEM2	ITEM3	ITEM4	ITEM5
1	1	1	2	1	2	2
2	2	2	2	2	1	1
3	3	1	1	2	2	1

Example: Print the data with the created format.

```
PROC PRINT;
   FORMAT ITEM1-ITEM5 ITEMFMT.;
   TITLE 'FORMATTED QUESTIONNAIRE DATA';
```

FORMATTED QUESTIONNAIRE DATA

OBS	ID	ITEM1	ITEM2	ITEM3	ITEM4	ITEM5
1	1	AGREE	DISAGREE	AGREE	DISAGREE	DISAGREE
2	2	DISAGREE	DISAGREE	DISAGREE	AGREE	AGREE
3	3	AGREE	AGREE	DISAGREE	DISAGREE	AGREE

The VALUE Statement

Format names

- must be 8 characters or fewer in length
- must begin with a letter or underscore for numeric variables
- must begin with a dollar sign for character variables
- cannot end with a number
- must be unique.

The VALUE Statement

Labels can be assigned to

- single numbers

```
VALUE Q 1='AGREE' 2='DISAGREE';
```

- ranges of numbers

```
VALUE AGEFMT  LOW-<0='MISCODED'
                0-12='CHILD'
               13-19='TEEN'
             20-HIGH='ADULT';
```

- several numbers that are not in a range

```
VALUE SEXFMT     1='FEMALE'
                 2='MALE'
             0,3-9='MISCODED';
```

- character values and ranges of character values.

```
VALUE $GRADE     'A'='GOOD'
             'B'-'D'='FAIR'
                 'E'='POOR'
             'I','U'='SEE INSTRUCTOR'
              OTHER ='MISCODED';
```

Note: character values must be enclosed in single quotes.

Labels must be

- 40 characters or fewer in length
- enclosed in single quotes.

The VALUE Statement

Example: Create formats for the SEX, HEIGHT, and WEIGHT variables in the CLASS data set.

```
DATA CLASS;
   INPUT NAME $ SEX $ AGE HEIGHT WEIGHT;
   CARDS;

data lines

PROC FORMAT;
   VALUE HTFMT    LOW-<55.0='SHORT'
                  55.0-66.0='AVERAGE'
                 66.0<-HIGH='TALL';
   VALUE WTFMT    LOW-<75.0='SKINNY'
                 75.0-125.0='AVERAGE'
                125.0<-HIGH='FAT';
   VALUE $SEXFMT 'M'='MALE'
                 'F'='FEMALE'
              OTHER ='MISCODED';
```

The VALUE Statement

Example: Use the formats with PROC PRINT.

```
PROC PRINT;
   FORMAT HEIGHT HTFMT. WEIGHT WTFMT. SEX $SEXFMT.;
```

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	MALE	12	AVERAGE	AVERAGE
2	JAMES	MALE	12	AVERAGE	AVERAGE
3	ALFRED	MALE	14	TALL	AVERAGE
4	WILLIAM	MALE	15	TALL	AVERAGE
5	JEFFREY	MALE	13	AVERAGE	AVERAGE
6	RONALD	MALE	15	TALL	FAT
7	THOMAS	MALE	11	AVERAGE	AVERAGE
8	PHILIP	MALE	16	TALL	FAT
9	ROBERT	MALE	12	AVERAGE	FAT
10	HENRY	MALE	14	AVERAGE	AVERAGE
11	JANET	FEMALE	15	AVERAGE	AVERAGE
12	JOYCE	FEMALE	11	SHORT	SKINNY
13	JUDY	FEMALE	14	AVERAGE	AVERAGE
14	CAROL	FEMALE	14	AVERAGE	AVERAGE
15	JANE	FEMALE	12	AVERAGE	AVERAGE
16	LOUISE	FEMALE	12	AVERAGE	AVERAGE
17	BARBARA	FEMALE	13	AVERAGE	AVERAGE
18	MARY	FEMALE	15	TALL	AVERAGE
19	ALICE	FEMALE	13	AVERAGE	AVERAGE

The VALUE Statement

Recall the following example from PROC FREQ.

A sample of 193 cases was extracted from the Public Use Survey Tapes of the 1980 U.S. Census.

```
DATA CENSUS;
   INPUT AGEGROUP $ 1-8 INCOME 10-14
         SEX $ 16-21 STATUS $ 23-29;
   IF INCOME<=5000 THEN INCOME=0;
   ELSE IF 5000<INCOME<=10000 THEN INCOME=5000;
   ELSE IF 10000<INCOME<=20000 THEN INCOME=10000;
   ELSE IF INCOME>20000 THEN INCOME=20000;
   CARDS;

data lines

PROC FREQ;
   TABLES INCOME;
```

INCOME	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
0	111	57.5	111	57.5
5000	45	23.3	156	80.8
10000	26	13.5	182	94.3
20000	11	5.7	193	100.0

The VALUE Statement

Example: Define an income-level format and use the format with PROC FREQ.

```
DATA CENSUS;
   INPUT AGEGROUP $ 1-8 INCOME 10-14
         SEX $ 16-21 STATUS $ 23-29;
   CARDS;

data lines

PROC FORMAT;
    VALUE INCLEVEL
                      0-5000='$5,000 & UNDER'
                5000<-10000='$5,001-$10,000'
               10000<-20000='$10,001-$20,000'
                20000<-HIGH='OVER $20,000';

PROC FREQ;
   TABLES INCOME;
   FORMAT INCOME INCLEVEL.;
```

INCOME	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
$5,000 & UNDER	111	57.5	111	57.5
$5,001-$10,000	45	23.3	156	80.8
$10,001-$20,000	26	13.5	182	94.3
OVER $20,000	11	5.7	193	100.0

The PICTURE Statement

Format names

- must be 8 characters or fewer in length
- must begin with a letter or underscore
- cannot end with a number
- must be unique.

Pictures

- are valid for numeric variables only
- are a sequence of characters in quotes that specify how a number is to be formatted
- are restricted to a maximum length of 24 characters
- are specified with three types of characters.

The PICTURE Statement

The three types of characters that define a picture are described below.

Character Type	Function
0 (zero)	defines positions for numeric digits (leading zeros are not printed).
1-9	defines positions for numeric digits (includes leading zeros).
non-numeric	defines message characters (printed after numeric digits start being formatted).

Examples:

```
PICTURE PHONE LOW-HIGH='000/000-0000';

PICTURE CREDIT LOW-<0='00,009.99DR'
               0-HIGH='00,009.99CR';
```

The PICTURE Statement

Example: Create and use a PICTURE format for printing phone numbers.

```
DATA NUMBERS;
   INPUT PHONENUM;
   CARDS;
9194678000
6031117777
3032220987
;
PROC FORMAT;
   PICTURE PHONE LOW-HIGH='000/000-0000';
```

The PICTURE Statement

Example: Print the data set without using the format.

```
PROC PRINT;
   TITLE 'PHONE NUMBERS';
```

PHONE NUMBERS

OBS	PHONENUM
1	9194678000
2	6031117777
3	3032220987

Example: Print the data set using the format.

```
PROC PRINT;
   FORMAT PHONENUM PHONE.;
   TITLE 'FORMATTED PHONE NUMBERS';
```

FORMATTED PHONE NUMBERS

OBS	PHONENUM
1	919/467-8000
2	603/111-7777
3	303/222-0987

The PICTURE Statement

Example: Create and use a PICTURE format for printing values as debits and credits.

```
DATA ACCOUNT;
   INPUT AMOUNT;
   CARDS;
4876.91
 -203.88
 -0.77
  0.98
;
PROC FORMAT;
   PICTURE CREDIT LOW-<0='00,009.99DR'
                  0-HIGH='00,009.99CR';
```

The PICTURE Statement

Example: Print the data without using the format.

```
PROC PRINT;
   TITLE 'LISTING OF THE AMOUNTS';
```

```
                    LISTING OF THE AMOUNTS

                      OBS     AMOUNT

                       1      4876.91
                       2      -203.88
                       3        -0.77
                       4         0.98
```

Example: Print the data using the format.

```
PROC PRINT;
   FORMAT AMOUNT CREDIT.;
   TITLE 'LISTING OF THE FORMATTED AMOUNTS';
```

```
                LISTING OF THE FORMATTED AMOUNTS

                      OBS       AMOUNT

                       1     4,876.91CR
                       2       203.88DR
                       3         0.77DR
                       4         0.98CR
```

9.4 Exercises

9.1 Use the payroll data and record layout given below for the following exercises.

Variable Name	Field Location	Variable Type
DEPT	1-3	numeric
NAME	5-12	character
NUMBER	14-18	numeric
SEX	20	character
NETPAY	22-28	numeric
GROSSPAY	30-36	numeric

Exercises

9.1 Listing of Raw Data

```
|---+----1----+----2----+----3----+----4

914 LARSON    11357 F  265.47  383.92
914 RYAN      10961 M  291.56  399.20
914 JOHNSON   10546 M  435.23  618.72
914 THOMPSON  11584 M  221.09  297.56
915 WOOD       8113 F  238.17  372.24
915 TAYLOR     2943 F  287.22  401.72
915 MOORE     12649 M  557.18  728.12
915 HELMS      3658 M  479.26  701.26
916 SHIRES    10052 F  487.12  729.22
916 RICHARDS   5781 M  319.27  472.84
916 SMITH      6237 F  207.09  345.88
916 MURPHY     6927 M  272.66  385.11
916 FAIRLEY    8846 M  521.66  834.80
917 HERZOG     6433 M  692.57 1045.26
917 TALL      11931 M  355.19  492.26
917 HIGGINS   11658 M  777.50 1235.46
917 DOOB      10554 F  669.06  972.29
918 EPERT      7716 M  224.36  310.40
918 CLANCY     6648 M  245.37  347.12
918 POWELL     8123 F  789.39 1271.54
```

Exercises

9.1 **a.** Create a SAS data set from the raw data on the preceding page with all the variables listed in the description of the record layout.

b. Print the data set.

c. Print the data set using the SAS format that will list the NETPAY and GROSSPAY values with dollar signs, commas, and two decimal places.

d. Create one format that converts 'F' to 'FEMALE' and 'M' to 'MALE' and a second format that converts the department numbers to the labels defined below. Print the data set using the new formats with the appropriate variables. Also format NETPAY and GROSSPAY appropriately.

914 = MARKETING
915 = MAINTENANCE
916 = RESEARCH & DEVELOPMENT
917 = ADMINISTRATION
918 = EDUCATION

e. Compute the average NETPAY for the employees in the company. Assign the label 'NET PAY OF EMPLOYEE' to the variable NETPAY.

f. Compute the average NETPAY by DEPT. Use the label described in part (e) for NETPAY, use the label 'DEPARTMENT' for the variable DEPT, and substitute the labels defined in part (d) for the values of DEPT.

Exercises

9.2 The Realtor's Association in a particular city keeps a data file of houses that are on the market. The record layout is described below.

Field Description	Field Location	Variable Type
ADDRESS	1-25	character
STYLE	26-33	character
SQUARE FEET	34-37	numeric
BEDROOMS	38	numeric
BATHS	39	numeric
AGE	40-41	numeric
FIREPLACE (1=YES,0=NO)	42	numeric
BASEMENT (1=YES,0=NO)	43	numeric
GARAGE (1=YES,0=NO)	44	numeric
ASSUMPTION (1=YES,0=NO)	45	numeric
PRICE	46-51	numeric

Write a complete SAS job that will solve all of the problems that follow. Assume the data are in the job stream.

Exercises

9.2 **a.** Prepare a SAS data set named HOUSES from the raw data file described on the previous page.

The data set HOUSES should contain the following variables:

ADDRESS	- street address of the house
STYLE	- style of house (RANCH, SPLIT, or TWOSTORY)
SQFEET	- heated square feet in the house
BR	- number of bedrooms
BATHS	- number of bathrooms
AGE	- age of house in years
FIRE	- is there a fireplace?
BASE	- is there a basement?
GARAGE	- is there a garage?
ASSUME	- is the loan assumable?
PRICE	- asking price of the house.

Exercises

9.2 **b.** Print the data set using formats to enhance the output. Write the prices of the houses with dollar signs and commas and write the numbers of bedrooms and bathrooms in words.

c. Compute the average cost of a house for each style of house. Use the label 'STYLE OF THE HOUSE' for the variable STYLE.

d. Print the HOUSES data set such that:

- prices are written with dollar signs and commas
- the number of bedrooms and the number of bathrooms are written in words
- the label YES or NO is written in place of the values of the variables FIRE, BASE, GARAGE and ASSUME where 1=YES and 0=NO
- the values of the variable SQFEET are written in the form '1500 SQ. FT.'
- the values of the variable AGE are written in the form '12 YEARS OLD'.

10. Processing Date and Time Values

10.1 SAS Date and Time Values

10.2 Date and Time Functions

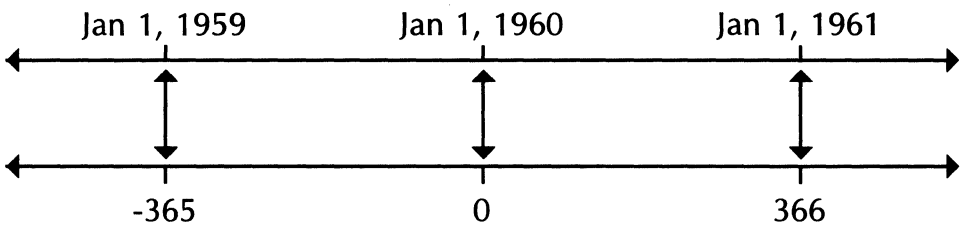
10.3 Exercises

10.1 SAS Date and Time Values

Implicit Date and Time Units

SAS date, time, and datetime values have implicit units.

- A date is represented by the number of days between January 1, 1960 and that date.
- A time is represented in seconds.
- A date and time is represented by the number of seconds between midnight January 1, 1960, and that date and time.



Date and Time Informats and Formats

Name of format or informat	Example	Informat, format, or both
DATE*w*.	04JUL1976	both
YYMMDD*w*.	76-07-04	both
MMDDYY*w*.	7/4/76	both
DDMMYY*w*.	4/7/76	both
MONYY*w*.	Jul76	both
YYQ*w*.	76Q3	both
WEEKDATE*w*.	Monday, July 4, 1976	format
WORDDATE*w*.	July 4, 1976	format
HHMM*w.d*	23:45	format
HOUR*w.d*	23	format
TIME*w.d*	23:45:23.5	both
DATETIME*w*.	04JUL1976:23:45:23.5	both

Note: see the *SAS User's Guide: Basics* for additional information on these and other formats and informats.

Reading Date Values

Example: A company wants to determine the length of employment in years for each of its employees who resigned in 1985.

```
DATA RESIGNED;
   INPUT NAME $10. +1 FIRSTDAY MMDDYY8.
        +1 LASTDAY MMDDYY8.;
   DAYS=(LASTDAY-FIRSTDAY)+1;
   YEARS=DAYS/365.25;
   CARDS;
ARNOTH, J. 12/01/73  4/30/85
DESTER, L.  7/14/61 12/31/85
HARLIN, M.  8/03/77  6/15/85
;
```

Note: remember that SAS date and time values have implicit units.

```
PROC PRINT;
```

OBS	NAME	FIRSTDAY	LASTDAY	DAYS	YEARS
1	ARNOTH, J.	5083	9251	4169	11.4141
2	DESTER, L.	560	9496	8937	24.4682
3	HARLIN, M.	6424	9297	2874	7.8686

Formatting Date Values

Example: Print the SAS date values using the DATE format.

```
PROC PRINT;
   FORMAT FIRSTDAY LASTDAY DATE7.;
```

OBS	NAME	FIRSTDAY	LASTDAY	DAYS	YEARS
1	ARNOTH, J.	01DEC73	30APR85	4169	11.4141
2	DESTER, L.	14JUL61	31DEC85	8937	24.4682
3	HARLIN, M.	03AUG77	15JUN85	2874	7.8686

Example: Print the SAS date values using the WORDDATE format.

```
PROC PRINT;
   FORMAT FIRSTDAY LASTDAY WORDDATE18.;
```

OBS	NAME	FIRSTDAY	LASTDAY	DAYS	YEARS
1	ARNOTH, J.	DECEMBER 1, 1973	APRIL 30, 1985	4169	11.4141
2	DESTER, L.	JULY 14, 1961	DECEMBER 31, 1985	8937	24.4682
3	HARLIN, M.	AUGUST 3, 1977	JUNE 15, 1985	2874	7.8686

10.2 Date and Time Functions

Selected Functions

Use the following date, time, and datetime functions to

- produce current time and date values

TIME	time of day
DATE (or **TODAY**)	today's date
DATETIME	time of day and date

- convert Julian dates

DATEJUL	Julian date to a SAS date
JULDATE	SAS date to a Julian date

- extract part of a SAS date value.

YEAR	4-digit year value
QTR	quarter of year (1-4)
MONTH	month of year (1-12)
DAY	day of month (1-31)
WEEKDAY	day of week (1-7)

Note: see the *SAS User's Guide: Basics* for additional functions.

Using Date and Time Functions

Example: Obtain the current date from the system clock and display it along with the year, quarter, month, day of the week, and day of the month.

```
DATA SHOWDATE;
   DATE=TODAY();
   NOFORMAT=DATE;
   YEAR=YEAR(DATE);
   QTR=QTR(DATE);
   MONTH=MONTH(DATE);
   WEEKDAY=WEEKDAY(DATE);
   DAY=DAY(DATE);

PROC PRINT DATA=SHOWDATE NOOBS;
   FORMAT DATE DATE7.;
```

DATE	NOFORMAT	YEAR	QTR	MONTH	WEEKDAY	DAY
29JAN86	9525	1986	1	1	4	29

Using Date and Time Functions

Example: Read a Julian date and print the date using the WEEKDATE format.

```
DATA DATES;
   INPUT JULIAN;
   DATE=DATEJUL(JULIAN);
   CARDS;
78318
79109
53188
60001
82162
;
PROC PRINT;
   FORMAT DATE WEEKDATE30.;
```

OBS	JULIAN	DATE
1	78318	TUESDAY, NOVEMBER 14, 1978
2	79109	THURSDAY, APRIL 19, 1979
3	53188	TUESDAY, JULY 7, 1953
4	60001	FRIDAY, JANUARY 1, 1960
5	82162	FRIDAY, JUNE 11, 1982

Date and Time Constants

Specific dates and times can be converted to SAS implicit units with the following SAS date, time, and datetime constants:

'04JUL76'D	SAS date value
'8:30'T	SAS time value
'04JUL76:8:30'DT	SAS datetime value.

Note: date, time, and datetime constants can be used in the DATA step or PROC step.

Date and Time Constants

Example: Given a charge account file, obtain a listing of all customers who paid late if the due date was January 29, 1986. Determine the number of days that each payment was overdue.

```
DATA ACCOUNTS;
   INPUT CUSTOMER 3. +1 DATEPAID DATE7. +1 AMOUNT 5.;
   IF DATEPAID>'29JAN86'D;
   DAYSOVER=DATEPAID-'29JAN86'D;
   FORMAT DATEPAID DATE9.;
   CARDS;
147 01FEB86 40.55
186 21JAN86 21.73
204 19JAN86 88.64
215 15FEB86 19.75
279 05MAR86 21.89
466 29JAN86 16.00
;
PROC PRINT;
```

OBS	CUSTOMER	DATEPAID	AMOUNT	DAYSOVER
1	147	01FEB1986	40.55	3
2	215	15FEB1986	19.75	17
3	279	05MAR1986	21.89	35

10.3 Exercises

10.1 A wholesaler of farm products uses a trucking firm to deliver ordered goods from a warehouse to its customers. For every order, the wholesaler records the date that the goods leave the warehouse (DATEOUT), the date the customer receives the goods (DATEIN), and the number of miles from the warehouse to the customer's establishment. The data are stored in a file with the following record layout:

Variable Name	Field Location	Variable Type
DATEOUT	1-8	YYMMDD8.
DATEIN	11-16	YYMMDD6.
MILES	21-24	numeric

Assume the data are placed in the job stream. Create a SAS data set with the variables listed above. Create a variable named RATIO that contains the average number of miles traveled per day for each order. Print the data set. Format all date values with the WEEKDATE format.

Exercises

10.2 A local insurance agent sends birthday cards to his clients. He has his data stored in a file with the following record layout:

Variable Name	Field Location	Variable Type
NAME	1-20	character
BORN	24-30	DATE7.

Obtain a listing of each client whose birthday is during the next calendar month. Format the output so that the dates are printed in the form 'July 4, 1976.'

Exercises

10.3 A local bank keeps pertinent data on the certificates of deposit it issues in a file with the following record layout:

Field Name	Field Location	Variable Type
ID	1-10	numeric
NAME	11-30	character
AMOUNT	31-38	numeric
RATE	39-43	numeric
ISSUE DATE	44-51	MMDDYY8.
EXPIRATION DATE	52-59	MMDDYY8.

Create a SAS data set that only contains the certificates that mature today. Assume the data are in the job stream. For each of these certificates, compute the amount of interest that the bank must pay. Produce a listing of the data set. Print the date values with an appropriate format and write all dollar values to two decimal places with dollar signs and commas.

Note: INTEREST=(no. of days)*(amount)*(rate)/365

11. Printer Graphics

11.1 Plotting One Variable against Another

11.2 Producing Bar, Star, Pie, and Block Charts

11.3 Exercises

11.1 Plotting One Variable against Another

The PLOT Procedure

The PLOT procedure graphs one variable against another, producing a printer plot.

PROC PLOT can be used to

- plot character variables
- reverse the scale on the Y axis
- specify the plot symbol
- scale the axes
- specify the length and width of a plot
- superimpose two or more plots
- draw horizontal and vertical reference lines
- draw contour plots.

The PLOT Procedure

General form of the PROC PLOT statement:

PROC PLOT *options*;

Selected PROC PLOT options:

DATA=*SASdataset*

Statements used with PROC PLOT:

BY *variable list*;
PLOT *vertical variable***horizontal variable . . . /options*;

The PLOT Procedure

Selected PLOT statement options:

- superimposing plots

  OVERLAY

- drawing lines on the plot

  HREF=*list of values*
  VREF=*list of values*
  HREFCHAR='*c***'**
  VREFCHAR='*c***'**

- scaling the axes

  VAXIS=*list of values*
  HAXIS=*list of values*

- length and width of the plot

  HPOS=*n*
  VPOS=*n*
  HSPACE=*n*
  VSPACE=*n*

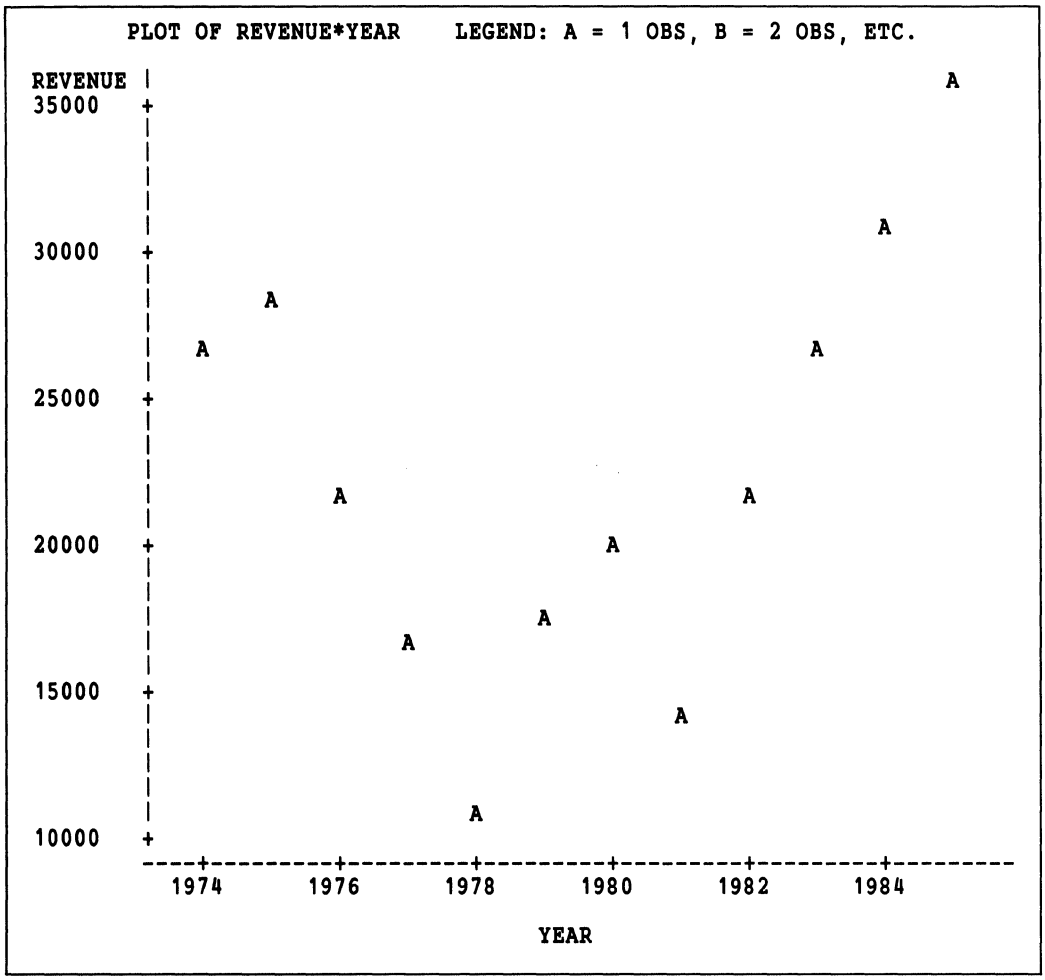
A Simple Plot

Example: Create a data set that contains revenue data over 12 years.

Plot revenue versus year using the default features of PROC PLOT.

```
OPTIONS LS=72 PS=44;
DATA TREND;
   INPUT YEAR REVENUE;
   CARDS;
1974 26250
1975 28350
1976 22000
1977 16800
1978 11220
1979 17700
1980 20180
1981 14000
1982 21500
1983 26500
1984 30450
1985 35750
;
PROC PLOT;
   PLOT REVENUE*YEAR;
```

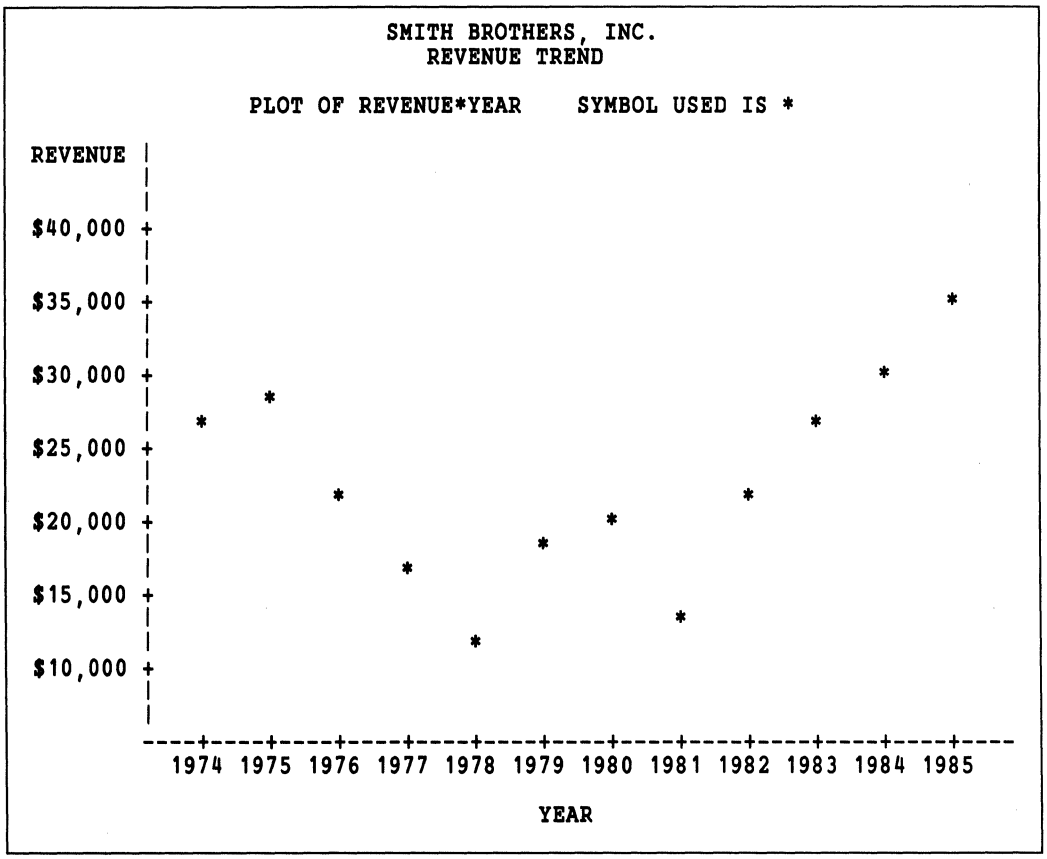
A Simple Plot



Scaling the Axes

Example: Define the plot symbol and use options to scale the axes.

```
PROC PLOT;
   PLOT REVENUE*YEAR='*'/VPOS=24
      VAXIS=10000 TO 40000 BY 5000
      HAXIS=1974 TO 1985 BY 1;
   TITLE 'SMITH BROTHERS, INC.';
   TITLE2 'REVENUE TREND';
   FORMAT REVENUE DOLLAR8.;
```



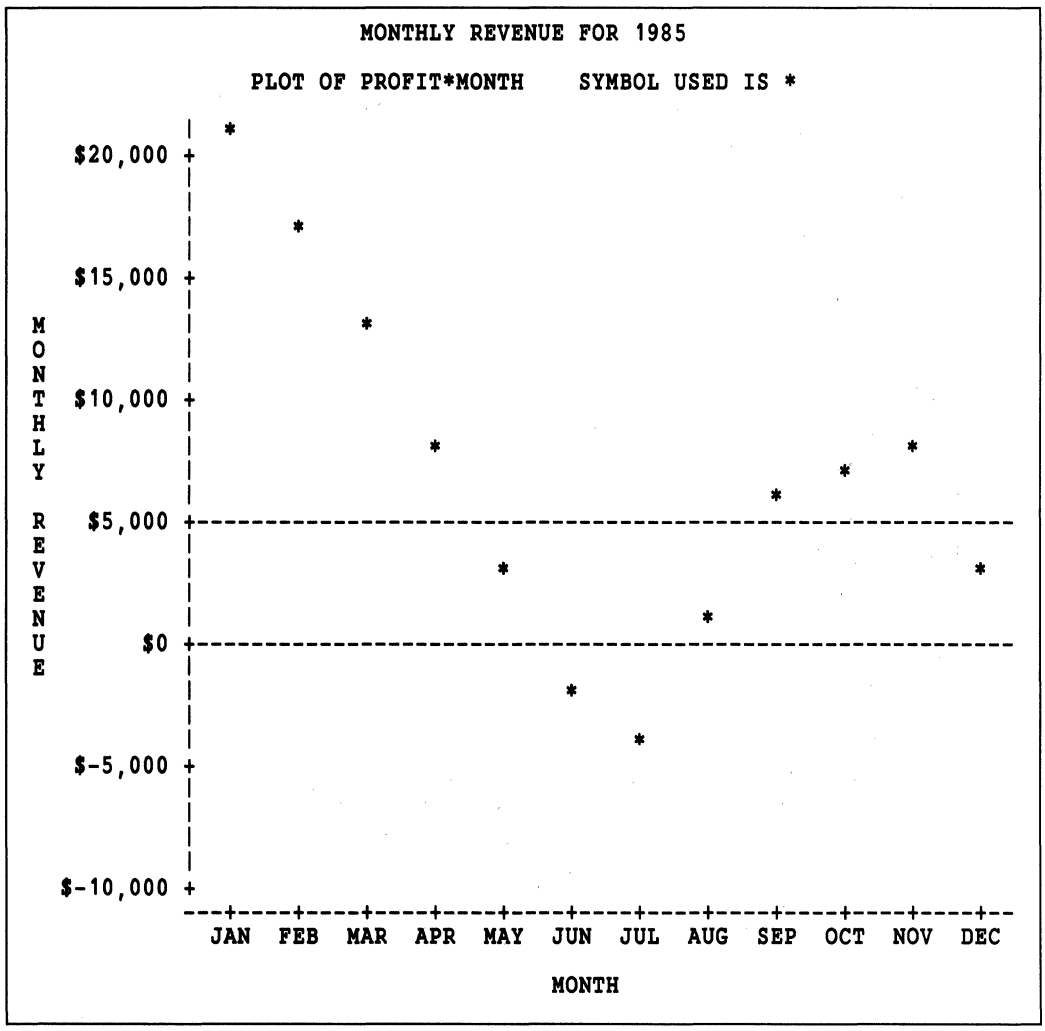
Plotting Character Variables

Example: Use PROC PLOT to examine monthly revenue for one year.

Use a LABEL statement to produce a label for the vertical axis.

```
OPTIONS LS=72 PS=44;
DATA SALES;
   INPUT MONTH $3. +1 PROFIT 5.;
   CARDS;
JAN 21317
FEB 16939
MAR 12754
APR  8489
MAY  3098
JUN -1953
JUL -3729
AUG   570
SEP  6097
OCT  6592
NOV  7644
DEC  2710
;
PROC PLOT;
   PLOT PROFIT*MONTH='*'/VREF=0 5000
      HAXIS='JAN' 'FEB' 'MAR' 'APR' 'MAY' 'JUN'
            'JUL' 'AUG' 'SEP' 'OCT' 'NOV' 'DEC';
   FORMAT PROFIT DOLLAR8.;
   LABEL PROFIT='MONTHLY REVENUE';
   TITLE 'MONTHLY REVENUE FOR 1985';
```

Plotting Character Variables

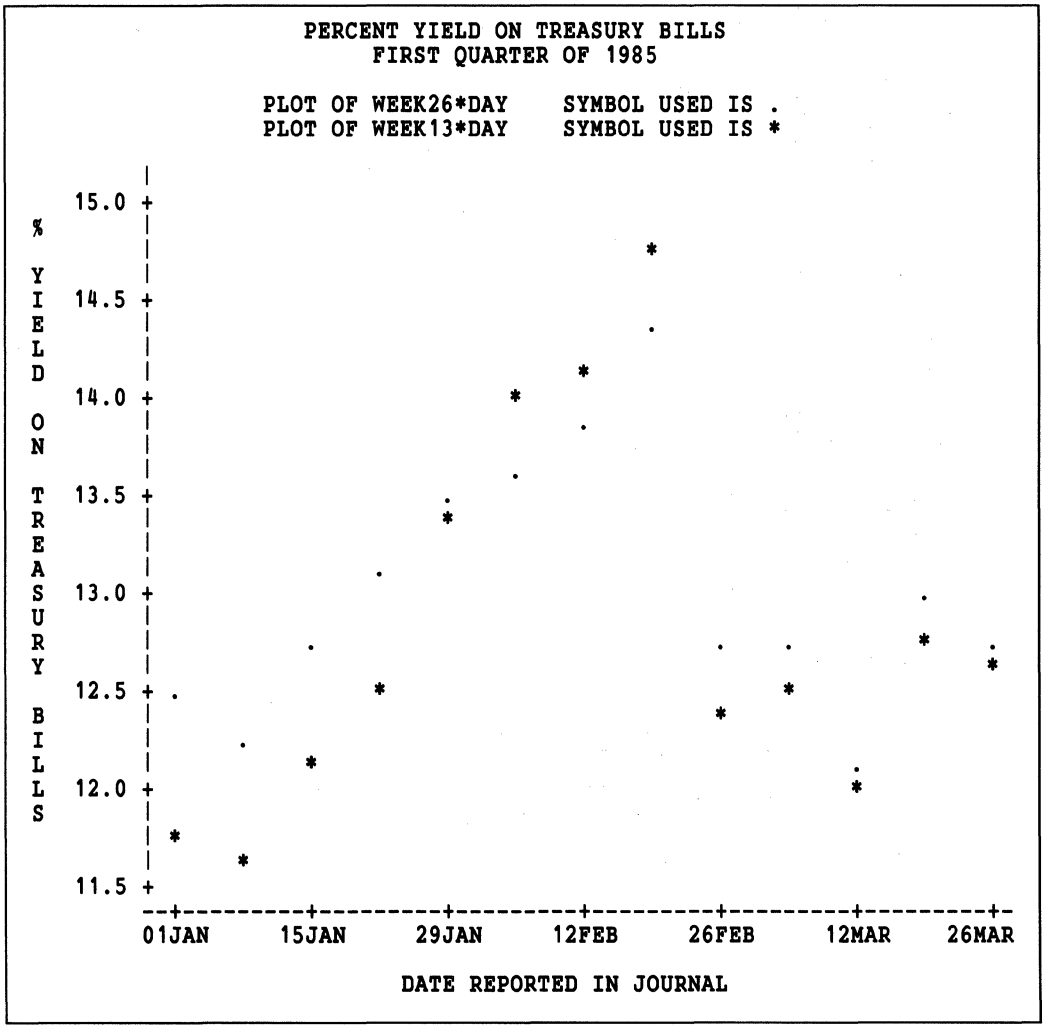


Superimposing Plots

Example: Use PROC PLOT to plot the percent yield on 13-week and 26-week Treasury bills for the first quarter of 1985.

```
OPTIONS LS=72 PS=44;
DATA BILLS;
   INPUT DAY MMDDYY8. WEEK26 WEEK13;
   CARDS;
01/01/85 12.448 11.690
01/08/85 12.282 11.658
01/15/85 12.806 12.121
01/22/85 13.102 12.505
01/29/85 13.530 13.364
02/05/85 13.570 13.974
02/12/85 13.831 14.110
02/19/85 14.360 14.740
02/26/85 12.695 12.430
03/05/85 12.786 12.450
03/12/85 12.128 12.030
03/19/85 12.962 12.809
03/26/85 12.773 12.583
;
PROC PLOT;
   PLOT WEEK26*DAY='.'  WEEK13*DAY='*' / OVERLAY;
   TITLE 'PERCENT YIELD ON TREASURY BILLS';
   TITLE2 'FIRST QUARTER OF 1985';
   LABEL DAY='DATE REPORTED IN JOURNAL'
      WEEK26='% YIELD ON TREASURY BILLS';
   FORMAT DAY DATE5.;
```

Superimposing Plots

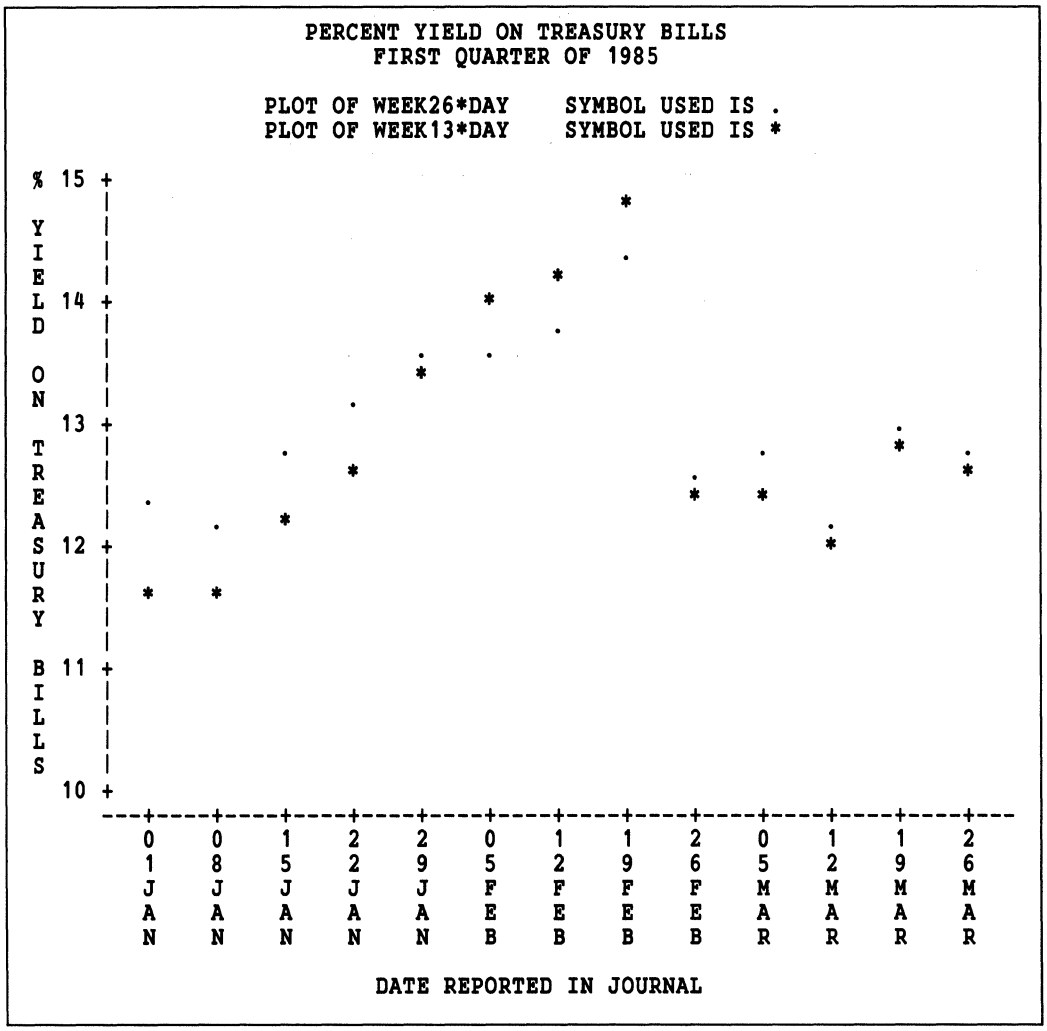


Superimposing Plots

Example: Repeat the preceding plot and specify a tick mark for each week on the horizontal axis.

```
PROC PLOT;
   PLOT WEEK26*DAY='.' WEEK13*DAY='*' / OVERLAY
      HAXIS='01JAN85'D TO '26MAR85'D BY 7;
   TITLE 'PERCENT YIELD ON TREASURY BILLS';
   TITLE2 'FIRST QUARTER OF 1985';
   LABEL DAY='DATE REPORTED IN JOURNAL'
      WEEK26='% YIELD ON TREASURY BILLS';
   FORMAT DAY DATE5.;
```

Superimposing Plots



11.2 Producing Bar, Star, Pie, and Block Charts

The CHART Procedure

The CHART procedure produces graphics using the line printer. It is useful for showing pictorially a variable's values or the relationships between two or more variables.

Kinds of charts:

- horizontal bar chart
- vertical bar chart
- pie chart
- star chart
- block chart

Quantities to chart:

- frequencies (histogram)
- cumulative frequencies
- percents
- cumulative percents
- sums
- means

PROC CHART produces charts for

- numeric variables—midpoints or discrete
- character variables—cannot exceed length 16.

The CHART Procedure

General form of the PROC CHART statement:

PROC CHART DATA=*SASdataset*;

Selected statements used with PROC CHART:

VBAR *variable names/options*;
HBAR *variable names/options*;
PIE *variable names/options*;
STAR *variable names/options*;
BLOCK *variable names/options*;

Selected options used with the above statements:

- determining the values represented by the lengths of the bars

  TYPE=*code*
  (FREQ CFREQ PCT CPCT SUM or **MEAN)**
  SUMVAR=*variable name*

- grouping among and within bars

  GROUP=*variable name*
  SUBGROUP=*variable name*

- classifying the observations into bars

  DISCRETE
  MIDPOINTS=*list of values*

- formatting the chart

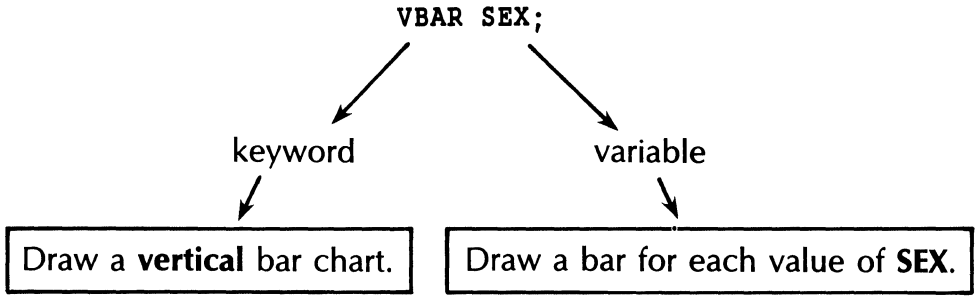
  ASCENDING
  DESCENDING
  NOZEROS
  SYMBOL=*'character'*
  NOSTAT

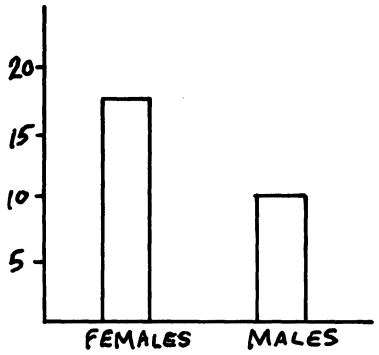
Note: see the *SAS User's Guide: Basics* for additional options.

Specifying a Chart

You specify a chart with a keyword followed by a variable name.

- The keyword VBAR, HBAR, BLOCK, PIE, or STAR indicates the **physical form** the chart will take.
- The **variable** indicates which values define the bars or sections.





Note: if no options are specified, the bars or sections represent frequencies.

Data for Upcoming Examples

Example: Evaluate payroll information for a company.

```
OPTIONS LS=72 PS=44;
TITLE 'LARIMER COMPANY PAYROLL REPORT';
DATA PAYROLL;
   INPUT BRANCH $ 1-11 DEPT 13-15 SEX $ 17
         GROSSPAY 19-25;
   FORMAT GROSSPAY DOLLAR9.2;
   CARDS;
    .
    .
    .
```

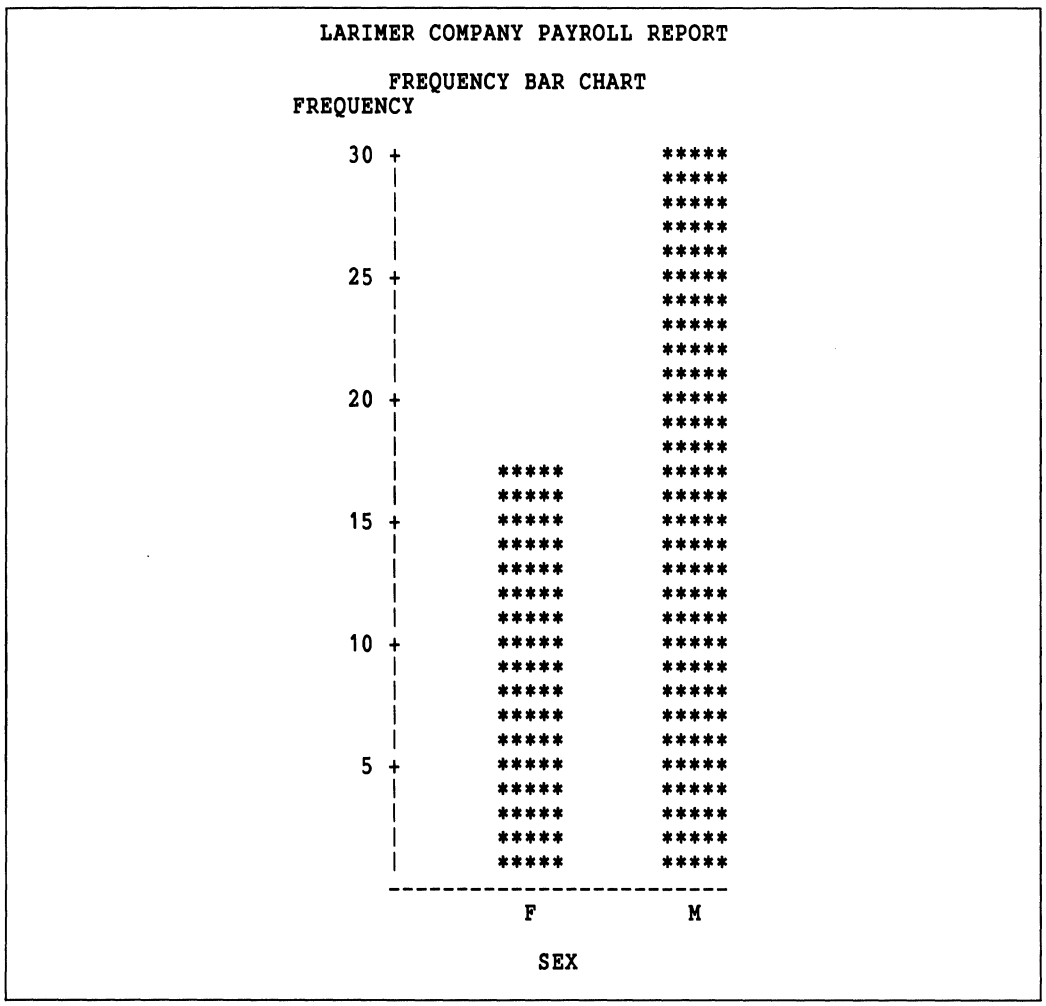
Data set PAYROLL has the following variables and values:

BRANCH	CHICAGO LOS ANGELES NEW YORK RALEIGH	DEPT	901 911 921
SEX	F M	GROSSPAY	continuous numeric

Vertical Bar Chart: Character Variable

Example: Produce a chart that displays the number of males and females in the company.

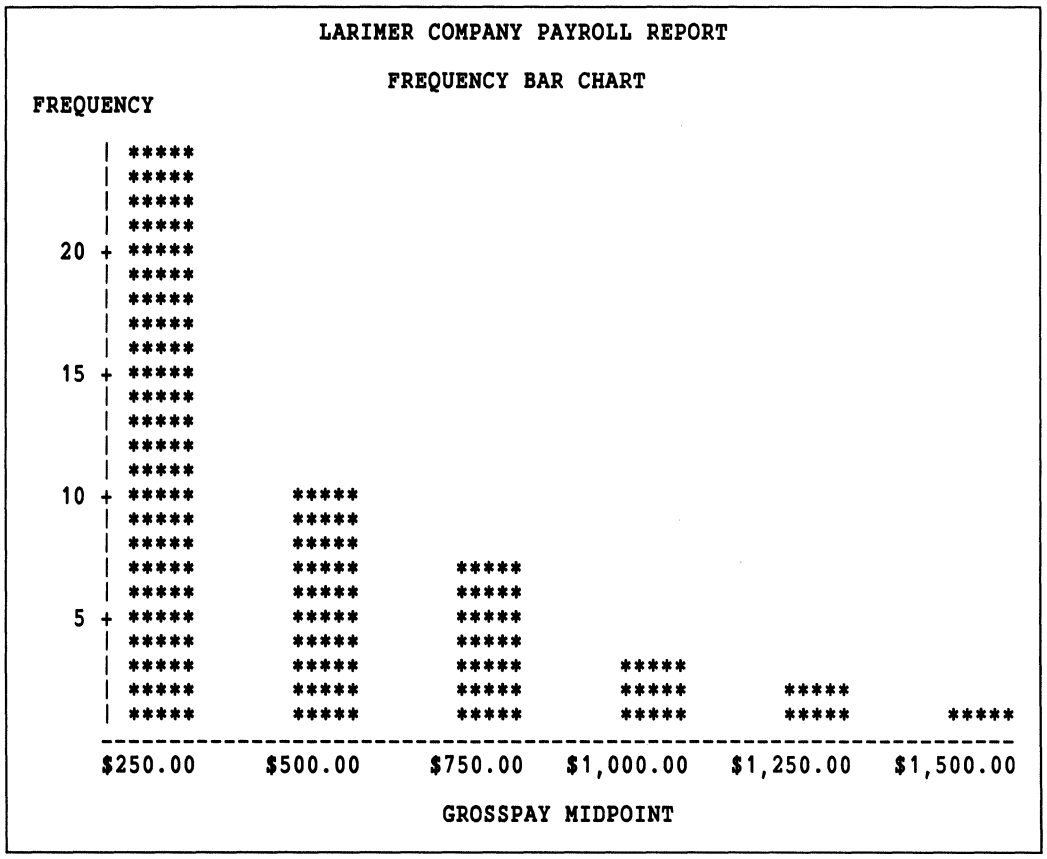
```
PROC CHART;
   VBAR SEX;
```



Vertical Bar Chart: Numeric Variable

Example: Produce a chart that displays the distribution of gross pay.

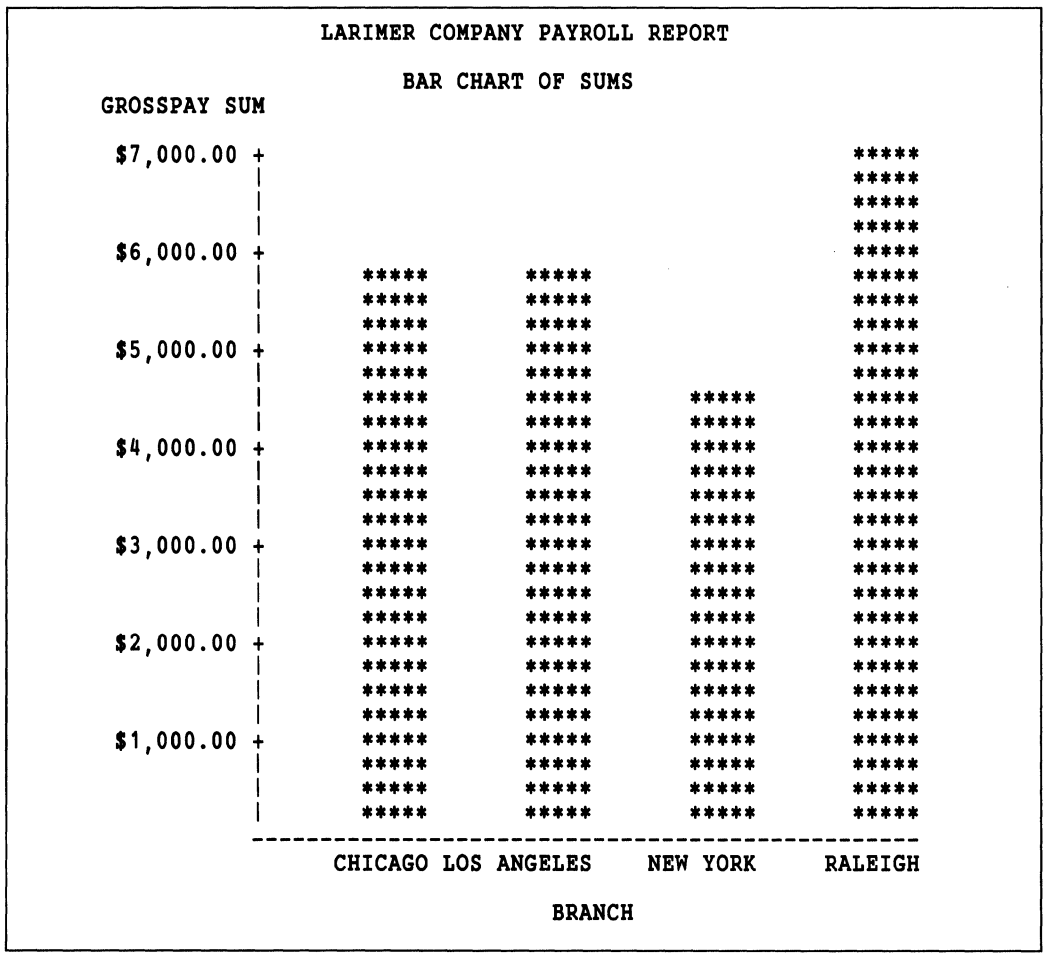
```
PROC CHART;
   VBAR GROSSPAY;
```



The SUMVAR= Option

Example: Produce a vertical bar chart that displays the total gross pay for each branch.

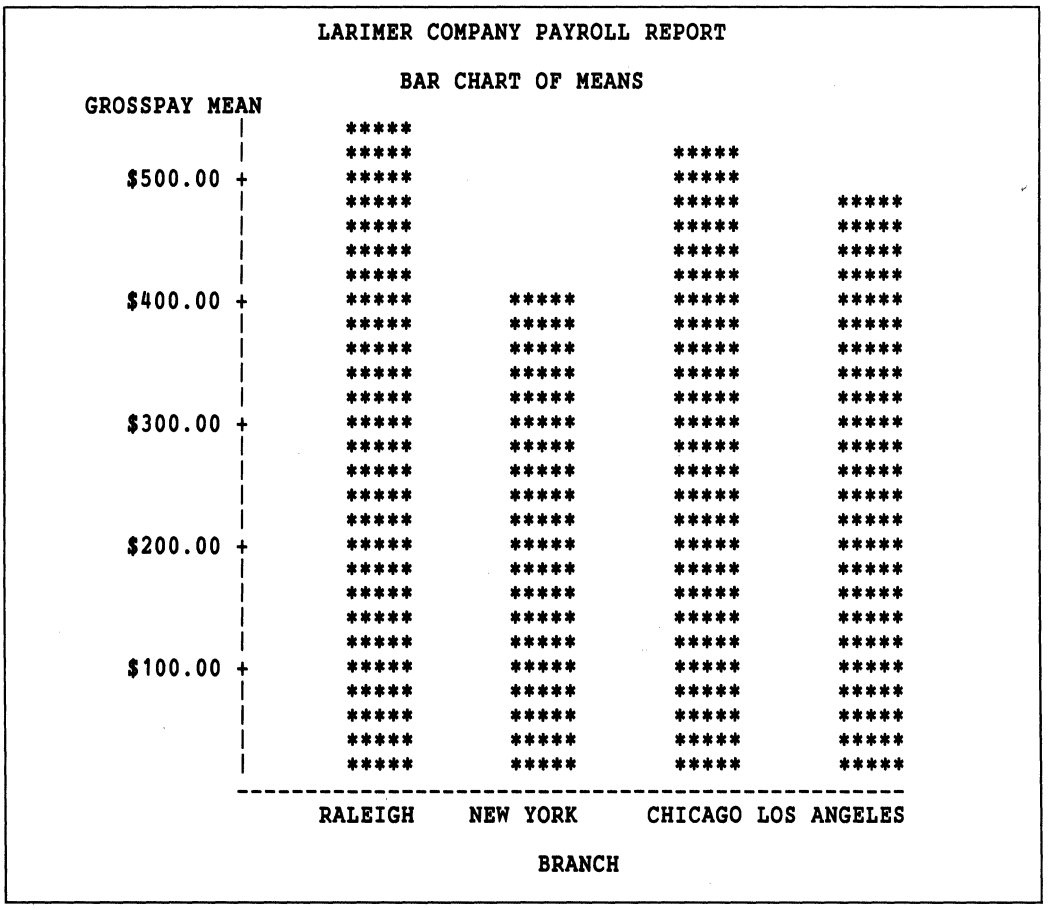
```
PROC CHART;
   VBAR BRANCH/SUMVAR=GROSSPAY;
```



Ordering the Bars

Example: Produce a vertical bar chart that displays the mean gross pay for each branch with the home office appearing first.

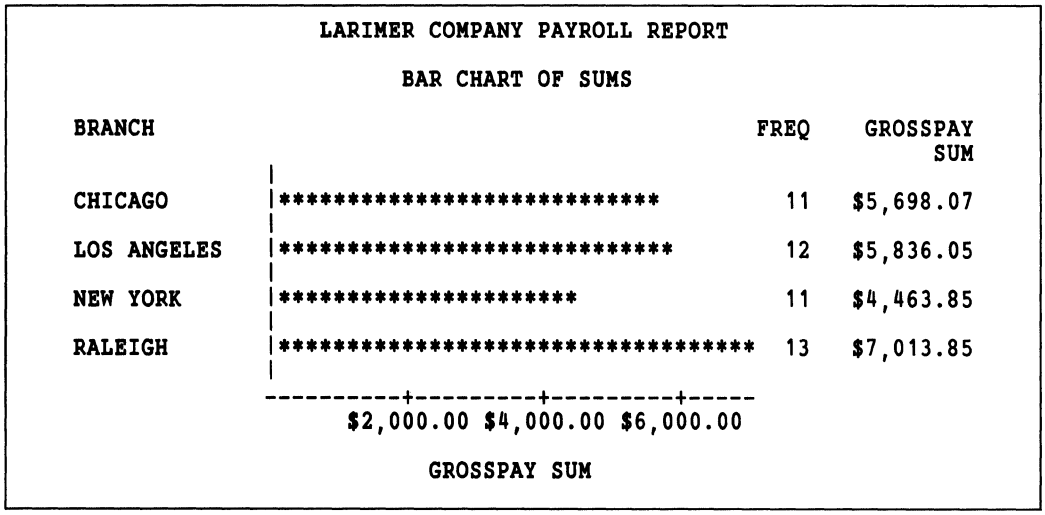
```
PROC CHART;
   VBAR BRANCH/SUMVAR=GROSSPAY TYPE=MEAN
      MIDPOINTS='RALEIGH' 'NEW YORK' 'CHICAGO'
      'LOS ANGELES';
```



A Horizontal Bar Chart

Example: Produce a horizontal bar chart that displays the total gross pay for each branch.

```
PROC CHART;
   HBAR BRANCH/SUMVAR=GROSSPAY;
```



```
                    LARIMER COMPANY PAYROLL REPORT

                              BAR CHART OF SUMS

BRANCH                                                FREQ     GROSSPAY
                                                                    SUM
                  |
CHICAGO           |****************************          11   $5,698.07
                  |
LOS ANGELES       |*****************************         12   $5,836.05
                  |
NEW YORK          |**********************              11   $4,463.85
                  |
RALEIGH           |**********************************  13   $7,013.85
                  |
                  ----------+---------+---------+-----
                        $2,000.00 $4,000.00 $6,000.00

                              GROSSPAY SUM
```

Subdividing Bars

Example: Produce a horizontal bar chart that displays the total gross pay for each department showing the contribution of each sex.

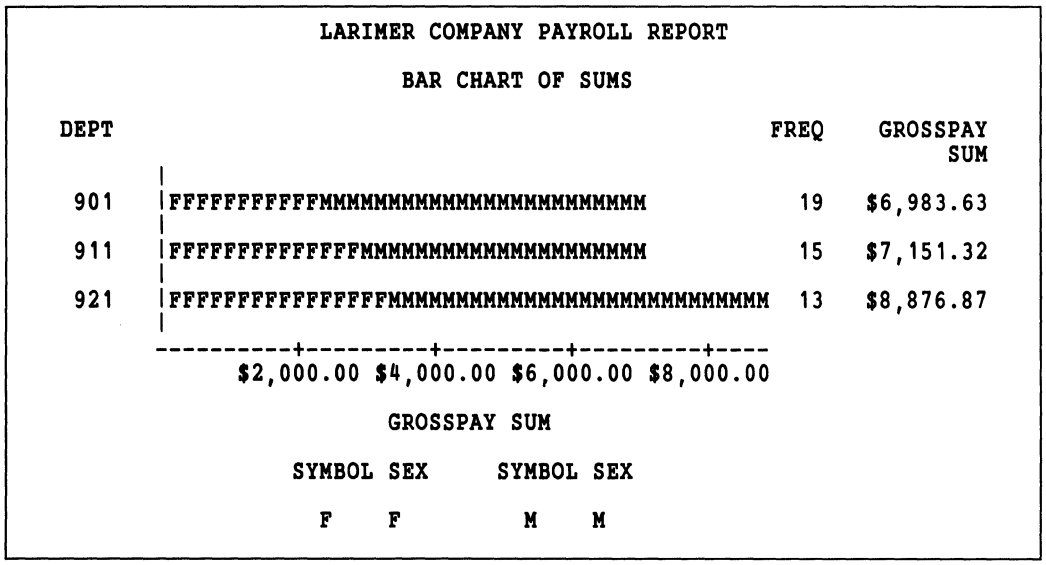
```
PROC CHART;
   HBAR DEPT/SUMVAR=GROSSPAY SUBGROUP=SEX;
```

```
                    LARIMER COMPANY PAYROLL REPORT

                              BAR CHART OF SUMS
MIDPOINT
  DEPT                                                          FREQ     GROSSPAY
                                                                           SUM
         |
   903   |FFFFFFFFFFFFMMMMMMMMMMMMMMMMMMMMMMM                     19   $6,983.63
         |
   906   |                                                         0       $0.00
         |
   909   |                                                         0       $0.00
         |
   912   |FFFFFFFFFFFFFFFFMMMMMMMMMMMMMMMMMMMMM                  15   $7,151.32
         |
   915   |                                                         0       $0.00
         |
   918   |FFFFFFFFFFFFFFFFFFMMMMMMMMMMMMMMMMMMMMMMMMMMMMM        13   $8,876.87
         |
         ----------+---------+---------+---------+----
               $2,000.00 $4,000.00 $6,000.00 $8,000.00

                           GROSSPAY SUM

                      SYMBOL SEX     SYMBOL SEX

                        F    F         M    M
```

Subdividing Bars

Example: Repeat the previous example and add the DISCRETE option.

```
PROC CHART;
   HBAR DEPT/SUMVAR=GROSSPAY SUBGROUP=SEX DISCRETE;
```



```
                    LARIMER COMPANY PAYROLL REPORT

                          BAR CHART OF SUMS

DEPT                                                          FREQ     GROSSPAY
                                                                          SUM
        |
 901    |FFFFFFFFFFFFMMMMMMMMMMMMMMMMMMMMMMMMM                  19    $6,983.63
        |
 911    |FFFFFFFFFFFFFFFMMMMMMMMMMMMMMMMMMMMM                   15    $7,151.32
        |
 921    |FFFFFFFFFFFFFFFFFMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM        13    $8,876.87
        |
        ----------+---------+---------+---------+----
              $2,000.00 $4,000.00 $6,000.00 $8,000.00

                          GROSSPAY SUM

                      SYMBOL SEX     SYMBOL SEX

                        F    F         M    M
```

Grouping Bars

Example: Produce a horizontal bar chart that displays the mean gross pay for each department grouped by branch.

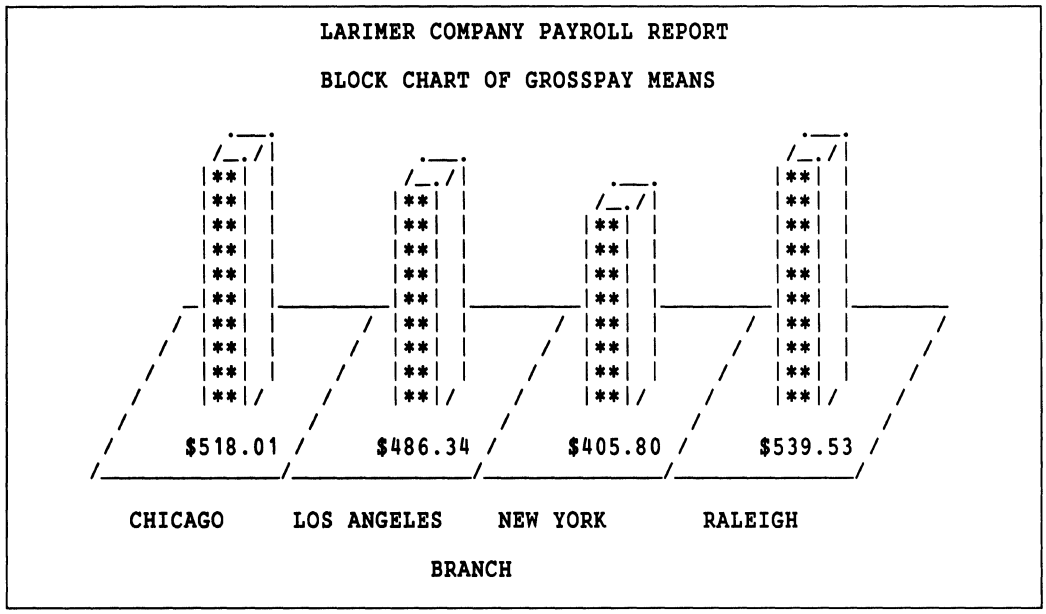
```
PROC CHART;
   HBAR DEPT/SUMVAR=GROSSPAY TYPE=MEAN GROUP=BRANCH
      DISCRETE;
```

```
                    LARIMER COMPANY PAYROLL REPORT

                              BAR CHART OF MEANS

BRANCH          DEPT                                    FREQ      GROSSPAY
                                                                        MEAN
                          |
CHICAGO          901      |*********                        5      $428.18
                 911      |***********                      3      $557.51
                 921      |**************                   3      $628.21
                          |
LOS ANGELES      901      |********                         5      $411.45
                 911      |************                     4      $576.18
                 921      |**********                       3      $491.36
                          |
NEW YORK         901      |*******                          4      $352.15
                 911      |*********                        3      $441.01
                 921      |*********                        4      $433.05
                          |
RALEIGH          901      |******                           5      $275.37
                 911      |*******                          5      $370.21
                 921      |************************         3    $1,261.98
                          |
                          ------------+-----------+-
                                    $600.00    $1,200.00

                                GROSSPAY MEAN
```

A Block Chart

Example: Produce a block chart that displays the mean gross pay for each branch.

```
PROC CHART;
   BLOCK BRANCH/SUMVAR=GROSSPAY TYPE=MEAN;
```

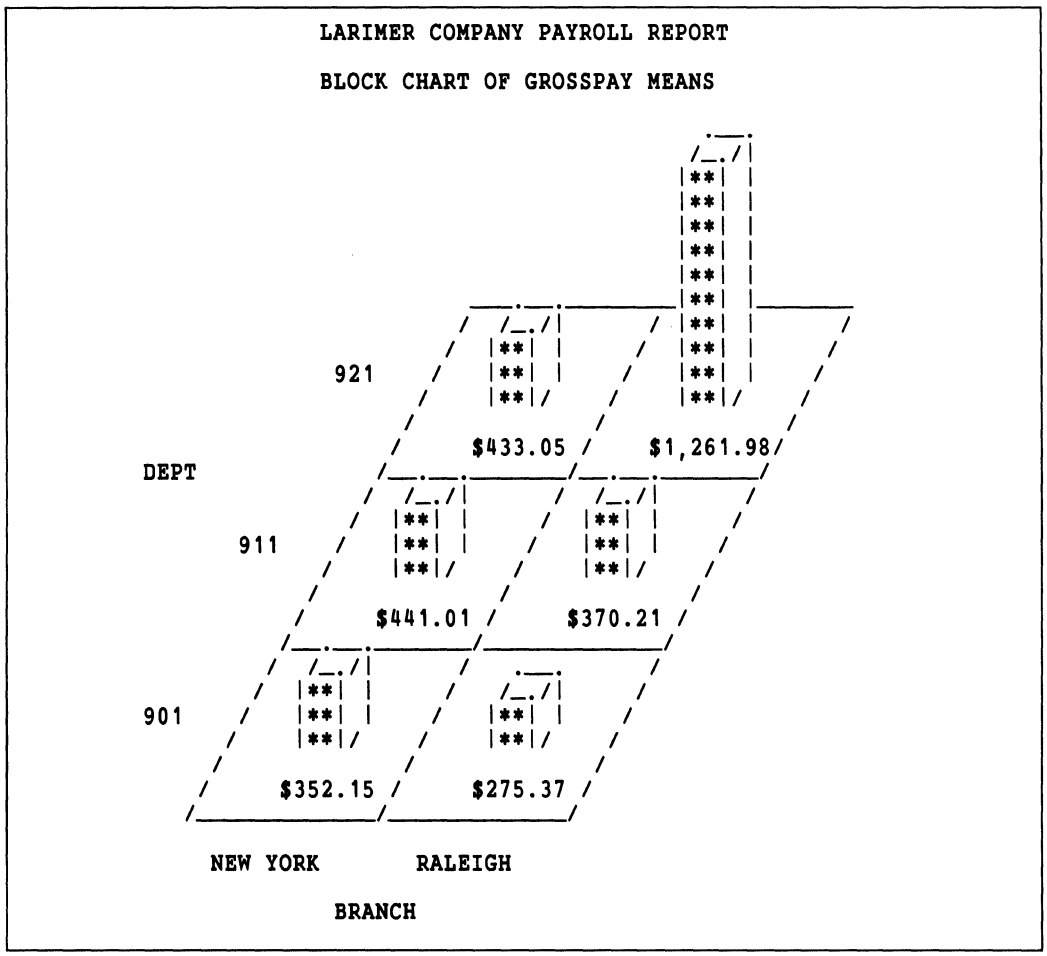


Grouping in a Block Chart

Example: Produce a block chart that displays the mean gross pay for the branches in Raleigh and New York grouped by department.

```
DATA PAYROLL2;
   SET PAYROLL;
   IF BRANCH='RALEIGH' OR BRANCH='NEW YORK';
PROC CHART DATA=PAYROLL2;
   BLOCK BRANCH/SUMVAR=GROSSPAY TYPE=MEAN GROUP=DEPT;
```

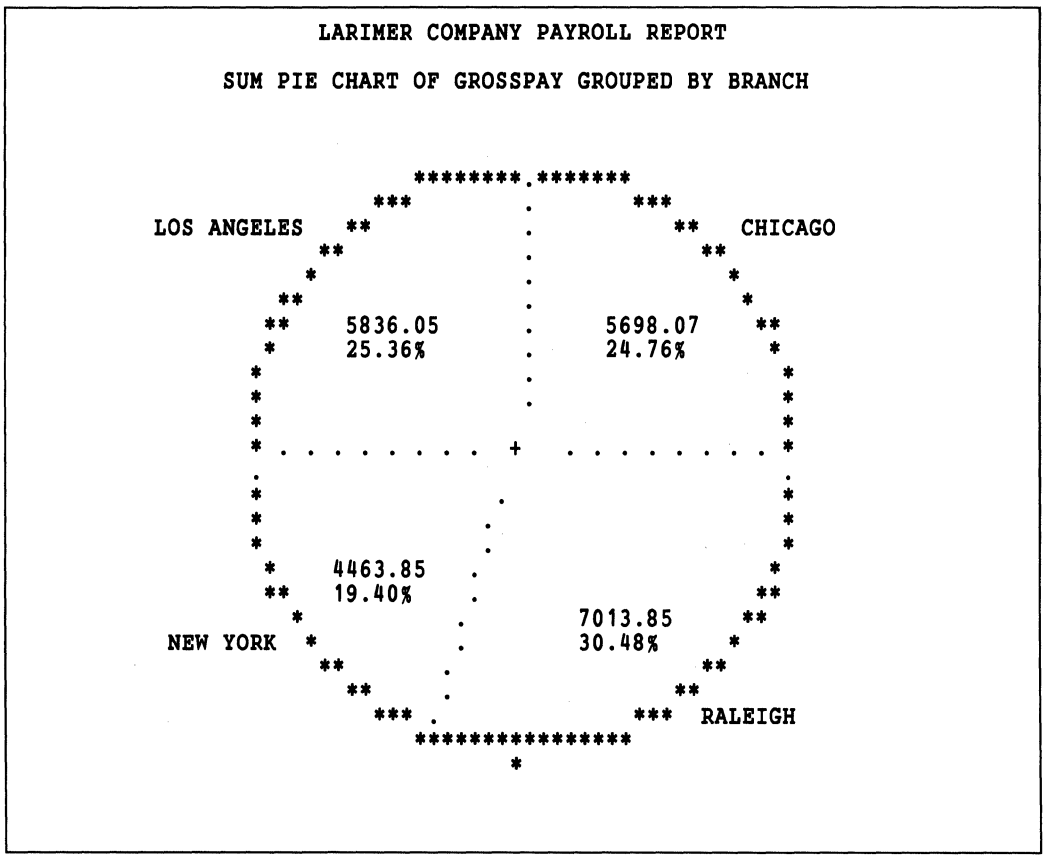
Grouping in a Block Chart



A Pie Chart

Example: Produce a pie chart that displays the total gross pay for each branch.

```
PROC CHART DATA=PAYROLL;
   PIE BRANCH/SUMVAR=GROSSPAY;
```



11.3 Exercises

11.1 The low and high temperatures were recorded each month in Chicago and Phoenix for one year. The record layout and data are given below.

Variable Name	Field Location	Variable Type
MONTH	1-2	numeric
SEASON	3	numeric
CITY	4-10	character
LOW	11-13	numeric
HIGH	14-16	numeric

Raw Data

Record Number	1...+....1....+....2....+ ← Field Position
1	14CHICAGO 17 32
2	24CHICAGO 20 35
3	31CHICAGO 29 45
4	41CHICAGO 40 59
5	51CHICAGO 50 70
6	62CHICAGO 60 81
7	72CHICAGO 65 84
8	82CHICAGO 64 83
9	93CHICAGO 56 76
10	103CHICAGO 46 65
11	113CHICAGO 33 48
12	124CHICAGO 22 35

Exercises

Raw Data

Record Number	|...+....1....+....2....+ ← Field Position
1	14PHOENIX 38 65
2	24PHOENIX 41 69
3	31PHOENIX 45 75
4	41PHOENIX 52 84
5	51PHOENIX 60 93
6	62PHOENIX 68102
7	72PHOENIX 78105
8	82PHOENIX 76102
9	93PHOENIX 69 98
10	103PHOENIX 57 88
11	113PHOENIX 45 75
12	124PHOENIX 39 66

11.1 Use the Chicago data on the preceding page for the following exercises. Write a complete SAS program that will solve all of the following problems.

a. Create a SAS data set that contains the temperature data for Chicago.

b. Plot the LOW temperature versus MONTH.

c. Plot the HIGH temperature versus MONTH.

d. Superimpose the LOW versus MONTH and HIGH versus MONTH plots on the same set of axes. Use a minus (−) as the plotting symbol for LOW versus MONTH and a plus (+) as the plotting symbol for HIGH versus MONTH.

Exercises

11.2 **a.** Use all the raw data for Chicago and Phoenix to create a SAS data set that contains temperature data for both cities. Add a variable (RANGE) to the data set that measures the difference between the high and low temperatures for each observation.

b. Produce a vertical bar chart that displays the average low temperature in each city.

c. Produce a block chart that displays the average high temperature for each city for each season.

d. For each city, produce a pie chart that displays the average high temperature for each season.

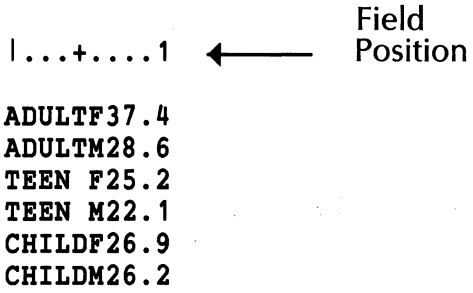
e. Produce a horizontal bar chart that displays the range in temperatures for each month within each city.

Note: You may need to adjust linesize and pagesize to produce vertical bar and block charts.

Exercises

11.3 Data used to estimate average television viewing time by residents in a large metropolitan area are collected periodically by a local pollster. Estimates produced in 1985 are shown in the table below.

Variable Name	Field Location	Variable Type
AGEGROUP	1-5	character
SEX	6	character
TIME (in hours)	7-10	numeric



a. Produce a horizontal bar chart that displays average viewing time for each AGEGROUP grouped by SEX. Use the label 'AGE CATEGORY' for the variable AGEGROUP. Order the bars such that CHILD appears first, TEEN second, and ADULT third within each level of SEX.

Exercises

11.3 **b.** Repeat part 'a' and substitute the following labels for the values of the variables SEX and AGEGROUP.

Variable Name	Variable Value	Label
AGEGROUP	ADULT	OVER 19
	TEEN	13 TO 19
	CHILD	UNDER 13
SEX	F	FEMALE
	M	MALE

Exercises

11.4 The U.S. Department of Energy evaluates U.S. passenger car efficiency each year by computing average miles traveled per gallon for all cars. The data recorded for 1968 through 1978 are given below.

Variable Name	Field Location	Variable Type
YEAR	1-4	numeric
MPG	5-9	numeric

```
                                    Field
1...+....1  <------------------     Position

196813.79
196913.63
197013.57
197113.57
197213.54
197313.10
197413.43
197513.53
197613.72
197713.94
197814.06
```

a. Plot MPG versus YEAR. Use an asterisk as the plotting symbol. Label the vertical axis 'AVERAGE MILES PER GALLON'. Request a tick mark for each value of YEAR. Select an appropriate title.

b. Produce a vertical bar chart that displays the MPG for each YEAR. Label the vertical axis 'AVERAGE MILES PER GALLON'.

12. Introduction to SAS Data Set Management

12.1 Review: Selected DATA Step Statements

Managing SAS Data Sets

SAS data sets can be changed by

- adding new variables (assignment statements)
- selecting a subset of the observations (DELETE, subsetting IF)
- selecting variables (DROP, KEEP)
- combining two or more data sets (SET, MERGE)
- changing values (UPDATE)
- adding observations (UPDATE).

Three statements read SAS data sets:

SET reads observations from one or more SAS data sets.

MERGE combines observations from two or more SAS data sets.

UPDATE provides the specialized function of updating a master file (SAS data set) with a transaction file (SAS data set).

Data for Upcoming Examples

Example: Create a SAS data set named PAYROLL.

```
DATA PAYROLL;
   INPUT DEPT 1-3 NAME $ 5-12 SEX $ 14
         NETPAY 16-22 GROSSPAY 24-30;
   CARDS;
917 DOBBS     F  169.06  272.29
918 EVANS     M  224.36  310.40
917 HIGGINS   M  777.50 1235.46
914 LARSON    F  415.47  483.92
918 POWELL    F  189.39  271.54
916 RICHARDS  M  219.27  352.84
914 RYAN      M  291.56  399.20
;
PROC PRINT;
   TITLE 'LISTING OF PAYROLL DATA SET';
```

LISTING OF PAYROLL DATA SET

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	917	DOBBS	F	169.06	272.29
2	918	EVANS	M	224.36	310.40
3	917	HIGGINS	M	777.50	1235.46
4	914	LARSON	F	415.47	483.92
5	918	POWELL	F	189.39	271.54
6	916	RICHARDS	M	219.27	352.84
7	914	RYAN	M	291.56	399.20

Creating Variables

Example: Create a variable that shows how much is withheld from each employee's check.

```
DATA PAYROLL2;
   SET PAYROLL;
   WITHHELD=GROSSPAY-NETPAY;
PROC PRINT;
   TITLE 'LISTING OF PAYROLL2 DATA SET';
```

LISTING OF PAYROLL2 DATA SET

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY	WITHHELD
1	917	DOBBS	F	169.06	272.29	103.23
2	918	EVANS	M	224.36	310.40	86.04
3	917	HIGGINS	M	777.50	1235.46	457.96
4	914	LARSON	F	415.47	483.92	68.45
5	918	POWELL	F	189.39	271.54	82.15
6	916	RICHARDS	M	219.27	352.84	133.57
7	914	RYAN	M	291.56	399.20	107.64

Deleting Observations

Example: Subset the PAYROLL data set by deleting the males.

```
DATA FEMALES;
   SET PAYROLL;
   IF SEX='M' THEN DELETE;
PROC PRINT;
   TITLE 'LISTING OF FEMALES';
```

LISTING OF FEMALES

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	917	DOBBS	F	169.06	272.29
2	914	LARSON	F	415.47	483.92
3	918	POWELL	F	189.39	271.54

Dropping Variables from a Data Set

Example: Use the PAYROLL data set to create a data set that does not contain salary information.

```
DATA GENERAL;
   SET PAYROLL;
   DROP NETPAY GROSSPAY;
PROC PRINT;
   TITLE 'GENERAL INFORMATION';
```

GENERAL INFORMATION

OBS	DEPT	NAME	SEX
1	917	DOBBS	F
2	918	EVANS	M
3	917	HIGGINS	M
4	914	LARSON	F
5	918	POWELL	F
6	916	RICHARDS	M
7	914	RYAN	M

Creating Multiple SAS Data Sets

Example: Use one DATA step to create a MALES data set and a FEMALES data set from the PAYROLL data set. Drop the variables NETPAY and GROSSPAY from both data sets.

```
DATA MALES FEMALES;
   SET PAYROLL;
   DROP NETPAY GROSSPAY;
   IF SEX='M' THEN OUTPUT MALES;
   ELSE OUTPUT FEMALES;
PROC PRINT DATA=MALES;
   TITLE 'LISTING OF MALES';
```

```
                    LISTING OF MALES

                OBS    DEPT    NAME        SEX

                 1     918     EVANS        M
                 2     917     HIGGINS      M
                 3     916     RICHARDS     M
                 4     914     RYAN         M
```

```
PROC PRINT DATA=FEMALES;
   TITLE 'LISTING OF FEMALES';
```

```
                   LISTING OF FEMALES

                OBS    DEPT    NAME        SEX

                 1     917     DOBBS        F
                 2     914     LARSON       F
                 3     918     POWELL       F
```

Creating Multiple SAS Data Sets

Example: Place different sets of variables into two SAS data sets created in the same step.

```
DATA GENERAL(DROP=NETPAY GROSSPAY)
     SALARY(DROP=DEPT SEX);
   SET PAYROLL;
PROC PRINT DATA=GENERAL;
   TITLE 'GENERAL DATA SET';
```

GENERAL DATA SET

OBS	DEPT	NAME	SEX
1	917	DOBBS	F
2	918	EVANS	M
3	917	HIGGINS	M
4	914	LARSON	F
5	918	POWELL	F
6	916	RICHARDS	M
7	914	RYAN	M

```
PROC PRINT DATA=SALARY;
   TITLE 'SALARY DATA SET';
```

SALARY DATA SET

OBS	NAME	NETPAY	GROSSPAY
1	DOBBS	169.06	272.29
2	EVANS	224.36	310.40
3	HIGGINS	777.50	1235.46
4	LARSON	415.47	483.92
5	POWELL	189.39	271.54
6	RICHARDS	219.27	352.84
7	RYAN	291.56	399.20

12.2 Multiple Input SAS Data Sets

The SET Statement

The SET statement is used to read observations from one or more SAS data sets.

General form of the SET statement:

SET *SASdatasets*;

- Up to 50 SAS data sets can be read with a single SET statement.
- The data sets are concatenated by default.
- The data sets are interleaved (combined in sorted order) if a BY statement is used.
- The input data sets must be sorted by the BY variable(s) if an interleave is desired.
- The output data set contains all variables present in the input data sets unless a DROP or KEEP statement is used.

Concatenating SAS Data Sets

Example: Combine the MALES and FEMALES data sets into one data set.

MALES DATA SET

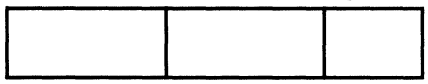
OBS	DEPT	NAME	SEX
1	918	EVANS	M
2	917	HIGGINS	M
3	916	RICHARDS	M
4	914	RYAN	M

FEMALES DATA SET

OBS	DEPT	NAME	SEX
1	917	DOBBS	F
2	914	LARSON	F
3	918	POWELL	F

```
DATA BOTH;
   SET MALES FEMALES;
PROC PRINT;
   TITLE 'RESULT OF THE CONCATENATION';
```

Program Data Vector



RESULT OF THE CONCATENATION

OBS	DEPT	NAME	SEX
1	918	EVANS	M
2	917	HIGGINS	M
3	916	RICHARDS	M
4	914	RYAN	M
5	917	DOBBS	F
6	914	LARSON	F
7	918	POWELL	F

Interleaving SAS Data Sets

Example: Combine the MALES and FEMALES data sets such that the resulting data set has its observations arranged in alphabetical order. (Note: both data sets are already in sorted order.)

MALES DATA SET

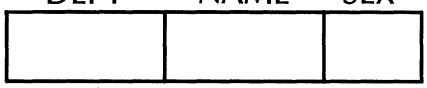
OBS	DEPT	NAME	SEX
1	918	EVANS	M
2	917	HIGGINS	M
3	916	RICHARDS	M
4	914	RYAN	M

FEMALES DATA SET

OBS	DEPT	NAME	SEX
1	917	DOBBS	F
2	914	LARSON	F
3	918	POWELL	F

```
DATA BOTHSORT;
   SET MALES FEMALES;
   BY NAME;
PROC PRINT;
   TITLE 'RESULT OF INTERLEAVING';
```

Program Data Vector



RESULT OF INTERLEAVING

OBS	DEPT	NAME	SEX
1	917	DOBBS	F
2	918	EVANS	M
3	917	HIGGINS	M
4	914	LARSON	F
5	918	POWELL	F
6	916	RICHARDS	M
7	914	RYAN	M

The MERGE Statement

The MERGE statement is used to join corresponding observations from two or more SAS data sets.

General form of the MERGE statement:

MERGE *SASdatasets*;

- Up to 50 SAS data sets can be merged in one DATA step.
- Merging is allowed with a BY statement (match merging).
- Merging is allowed without a BY statement (one-to-one merging).
- The input data sets must be sorted by the matching variable(s) if a BY statement is used.
- The output data set contains all variables present in the input data sets unless a DROP or KEEP statement is used.

One-to-One Merging

Example: Combine the observations from the GENERAL and SALARY data sets such that the new data set contains the same information as the original PAYROLL data set.

GENERAL DATA SET

OBS	DEPT	NAME	SEX
1	917	DOBBS	F
2	918	EVANS	M
3	917	HIGGINS	M
4	914	LARSON	F
5	918	POWELL	F
6	916	RICHARDS	M
7	914	RYAN	M

SALARY DATA SET

OBS	NAME	NETPAY	GROSSPAY
1	DOBBS	169.06	272.29
2	EVANS	224.36	310.40
3	HIGGINS	777.50	1235.46
4	LARSON	415.47	483.92
5	POWELL	189.39	271.54
6	RICHARDS	219.27	352.84
7	RYAN	291.56	399.20

```
DATA MERGED;
   MERGE GENERAL SALARY;
PROC PRINT;
   TITLE 'MERGED DATA SET';
```

Program Data Vector

DEPT	NAME	SEX	NETPAY	GROSSPAY

One-to-One Merging

MERGED DATA SET

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	917	DOBBS	F	169.06	272.29
2	918	EVANS	M	224.36	310.40
3	917	HIGGINS	M	777.50	1235.46
4	914	LARSON	F	415.47	483.92
5	918	POWELL	F	189.39	271.54
6	916	RICHARDS	M	219.27	352.84
7	914	RYAN	M	291.56	399.20

One-to-One Merging

Example: Suppose Larson is missing from the SALARY data set.

Merge the GENERAL and SALARY data sets again.

```
DATA MERGED;
   MERGE GENERAL SALARY;
PROC PRINT;
   TITLE 'ONE-TO-ONE MERGING';
   TITLE2 'UNEQUAL NUMBERS OF OBSERVATIONS';
```

Program Data Vector

DEPT	NAME	SEX	NETPAY	GROSSPAY

One-to-One Merging

GENERAL DATA SET

OBS	DEPT	NAME	SEX
1	917	DOBBS	F
2	918	EVANS	M
3	917	HIGGINS	M
4	914	LARSON	F
5	918	POWELL	F
6	916	RICHARDS	M
7	914	RYAN	M

SALARY DATA SET

OBS	NAME	NETPAY	GROSSPAY
1	DOBBS	169.06	272.29
2	EVANS	224.36	310.40
3	HIGGINS	777.50	1235.46
4	POWELL	189.39	271.54
5	RICHARDS	219.27	352.84
6	RYAN	291.56	399.20

Program Data Vector

DEPT	NAME	SEX	NETPAY	GROSSPAY

ONE-TO-ONE MERGING
UNEQUAL NUMBERS OF OBSERVATIONS

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	917	DOBBS	F	169.06	272.29
2	918	EVANS	M	224.36	310.40
3	917	HIGGINS	M	777.50	1235.46
4	914	POWELL	F	189.39	271.54
5	918	RICHARDS	F	219.27	352.84
6	916	RYAN	M	291.56	399.20
7	914	RYAN	M	.	.

Match Merging

Example: Suppose Larson is missing from the SALARY data set. (Note: both data sets are already in sorted order.)

Merge the GENERAL and PAYROLL data sets again.

```
DATA MERGED;
   MERGE GENERAL SALARY;
   BY NAME;
PROC PRINT;
   TITLE 'MATCH MERGING';
   TITLE2 'UNEQUAL NUMBERS OF OBSERVATIONS';
```

Program Data Vector

DEPT	NAME	SEX	NETPAY	GROSSPAY

Match Merging

GENERAL DATA SET

OBS	DEPT	NAME	SEX
1	917	DOBBS	F
2	918	EVANS	M
3	917	HIGGINS	M
4	914	LARSON	F
5	918	POWELL	F
6	916	RICHARDS	M
7	914	RYAN	M

SALARY DATA SET

OBS	NAME	NETPAY	GROSSPAY
1	DOBBS	169.06	272.29
2	EVANS	224.36	310.40
3	HIGGINS	777.50	1235.46
4	POWELL	189.39	271.54
5	RICHARDS	219.27	352.84
6	RYAN	291.56	399.20

Program Data Vector

DEPT	NAME	SEX	NETPAY	GROSSPAY

MATCH MERGING
UNEQUAL NUMBERS OF OBSERVATIONS

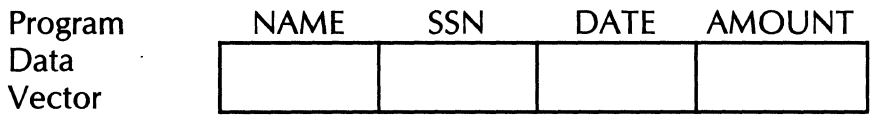
OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	917	DOBBS	F	169.06	272.29
2	918	EVANS	M	224.36	310.40
3	917	HIGGINS	M	777.50	1235.46
4	914	LARSON	F	.	.
5	918	POWELL	F	189.39	271.54
6	916	RICHARDS	M	219.27	352.84
7	914	RYAN	M	291.56	399.20

Match Merging

Example: Merge a SAS data set named CLIENTS that contains names and social security numbers of individuals in a health insurance plan and a SAS data set named AMOUNTS that contains names, dates, and amounts of claims.

Note that the AMOUNTS data set has multiple observations with the same name.

```
DATA CLAIMS;
   MERGE CLIENTS AMOUNTS;
   BY NAME;
PROC PRINT;
   TITLE 'MATCH MERGING';
   TITLE2 'MULTIPLE BY-VALUE OCCURRENCES';
```



Match Merging

CLIENTS DATA SET

OBS	NAME	SSN
1	ANKERTON,L.	111235678
2	DAVIS,R.	222987661
3	MASTERS,T.	333514272

AMOUNTS DATA SET

OBS	NAME	DATE	AMOUNT
1	ANKERTON,L.	08OCT82	92
2	ANKERTON,L.	15OCT82	43
3	DAVIS,R.	04OCT82	16
4	MASTERS,T.	13OCT82	18
5	MASTERS,T.	15OCT82	27
6	MASTERS,T.	21OCT82	15

Program Data Vector

NAME	SSN	DATE	AMOUNT

MATCH MERGING
MULTIPLE BY-VALUE OCCURRENCES

OBS	NAME	SSN	DATE	AMOUNT
1	ANKERTON,L.	111235678	08OCT82	92
2	ANKERTON,L.	111235678	15OCT82	43
3	DAVIS,R.	222987661	04OCT82	16
4	MASTERS,T.	333514272	13OCT82	18
5	MASTERS,T.	333514272	15OCT82	27
6	MASTERS,T.	333514272	21OCT82	15

Note: observations with matching values of the BY variable are joined.

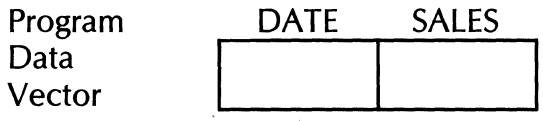
The program data vector is initialized to missing each time the BY variable's value changes.

Match Merging

Example: Merge a SAS data set named CLOTHES that contains dates and amounts of clothing sales in a sporting goods store and a SAS data set named EQUIP that contains dates and amounts of sporting equipment sales in the store.

Note that the data sets have identical variable names.

```
DATA ALLSALES;
   MERGE CLOTHES EQUIP;
   BY DATE;
PROC PRINT;
   TITLE 'MERGING DATA SETS';
   TITLE2 'WITH IDENTICAL VARIABLE NAMES';
```



Match Merging

CLOTHES DATA SET

OBS	DATE	SALES
1	18OCT82	223.93
2	19OCT82	387.82
3	20OCT82	229.28
4	21OCT82	318.32
5	22OCT82	519.07

EQUIP DATA SET

OBS	DATE	SALES
1	18OCT82	492.28
2	19OCT82	228.20
3	20OCT82	542.98
4	21OCT82	325.02
5	22OCT82	733.60

Program Data Vector

DATE	SALES

MERGING DATA SETS
WITH IDENTICAL VARIABLE NAMES

OBS	DATE	SALES
1	18OCT82	492.28
2	19OCT82	228.20
3	20OCT82	542.98
4	21OCT82	325.02
5	22OCT82	733.60

Match Merging

Example: Repeat the preceding example using the RENAME data set option and compute the total sales for each date.

CLOTHES DATA SET

OBS	DATE	SALES
1	18OCT82	223.93
2	19OCT82	387.82
3	20OCT82	229.28
4	21OCT82	318.32
5	22OCT82	519.07

EQUIP DATA SET

OBS	DATE	SALES
1	18OCT82	492.28
2	19OCT82	228.20
3	20OCT82	542.98
4	21OCT82	325.02
5	22OCT82	733.60

```
DATA ALLSALES;
   MERGE CLOTHES(RENAME=(SALES=CL_SALES))
         EQUIP  (RENAME=(SALES=EQ_SALES));
   BY DATE;
   TOTAL=CL_SALES+EQ_SALES;
PROC PRINT;
   TITLE 'RESULT OF MERGING DATA SETS';
   TITLE2 'WITH IDENTICAL VARIABLE NAMES';
   TITLE3 'USING THE RENAME DATA SET OPTION';
```

Program Data Vector

DATE	CL_SALES	EQ_SALES	TOTAL

Match Merging

RESULT OF MERGING DATA SETS
WITH IDENTICAL VARIABLE NAMES
USING THE RENAME DATA SET OPTION

OBS	DATE	CL_SALES	EQ_SALES	TOTAL
1	18OCT82	223.93	492.28	716.21
2	19OCT82	387.82	228.20	616.02
3	20OCT82	229.28	542.98	772.26
4	21OCT82	318.32	325.02	643.34
5	22OCT82	519.07	733.60	1252.67

12.3 Updating SAS Data Sets

The UPDATE Statement

The function of the UPDATE statement is to update a master file (SAS data set) with data in a transaction file (SAS data set).

General form of the UPDATE Statement:

UPDATE *masterdataset transactiondataset*;
BY *identifier-variable(s)*;

You can use the UPDATE statement to

- change the values of variables
- add observations to the SAS data set.

Restrictions for updating:

- Only two data set names can appear in the UPDATE statement.
- The master data set must be listed first.
- A BY statement that gives the matching variable(s) must be used.
- Both data sets must be sorted by the matching variable(s).
- The master data set must not contain more than one observation with the same BY value.

UPDATE Application

Example: Add new employees and make changes to existing values for old employees in the PAYROLL data set. (Note: the PAYROLL data set has already been sorted by NAME.)

```
DATA NEWINFO;
   INPUT DEPT 1-3 NAME $ 5-12 SEX $ 14
         NETPAY 16-22 GROSSPAY 24-30;
   CARDS;
    POWELL       221.75  310.62
916 SERPANT   M  207.22  398.65
    DOBBS        991.65 1272.29
918 ARCHER    F  315.17  420.00
;
PROC SORT DATA=NEWINFO;
   BY NAME;
```

UPDATE Application

```
PROC PRINT;
   TITLE 'PAYROLL DATA SET';
```

PAYROLL DATA SET

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	917	DOBBS	F	169.06	272.29
2	918	EVANS	M	224.36	310.40
3	917	HIGGINS	M	777.50	1235.46
4	914	LARSON	F	415.47	483.92
5	918	POWELL	F	189.39	271.54
6	916	RICHARDS	M	219.27	352.84
7	914	RYAN	M	291.56	399.20

```
PROC PRINT DATA=NEWINFO;
   TITLE 'NEWINFO DATA SET';
```

NEWINFO DATA SET

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	918	ARCHER	F	315.17	420.00
2	.	DOBBS		991.65	1272.29
3	.	POWELL		221.75	310.62
4	916	SERPANT	M	207.22	398.65

UPDATE Application

Example: Update the PAYROLL data set with the NEWINFO data set.

```
DATA PAYROLL2;
   UPDATE PAYROLL NEWINFO;
   BY NAME;
PROC PRINT;
   TITLE 'PAYROLL2 DATA SET';
```

PAYROLL2 DATA SET

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	918	ARCHER	F	315.17	420.00
2	917	DOBBS	F	991.65	1272.29
3	918	EVANS	M	224.36	310.40
4	917	HIGGINS	M	777.50	1235.46
5	914	LARSON	F	415.47	483.92
6	918	POWELL	F	221.75	310.62
7	916	RICHARDS	M	219.27	352.84
8	914	RYAN	M	291.56	399.20
9	916	SERPANT	M	207.22	398.65

UPDATE Application

Example: Assign missing values to variables in the master file.

MASTER DATA SET

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	917	DOBBS	F	169.06	272.29
2	917	IG*S	S	777.50	123.54
3	914	LARSON	F	415.47	483.92
4	914	RYAN	M	291.56	399.20
5	914	SON	M	21.09	297.56

```
DATA TRANS;
   LENGTH SEX $ 1;
   INPUT NAME $ DEPT SEX $ NETPAY GROSSPAY;
   MISSING _;
   CARDS;
IG*S  .  _  _  _
SON   .  .  _  _
PROC PRINT;
   TITLE 'TRANSACTION DATA SET';
```

TRANSACTION DATA SET

OBS	SEX	NAME	DEPT	NETPAY	GROSSPAY
1	_	IG*S	.	_	_
2		SON	.	_	_

UPDATE Application

MASTER DATA SET

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	917	DOBBS	F	169.06	272.29
2	917	IG*S	S	777.50	123.54
3	914	LARSON	F	415.47	483.92
4	914	RYAN	M	291.56	399.20
5	914	SON	M	21.09	297.56

TRANSACTION DATA SET

OBS	SEX	NAME	DEPT	NETPAY	GROSSPAY
1	–	IG*S	.	–	–
2		SON	.	–	–

```
DATA MASTER2;
   UPDATE MASTER1 TRANS;
   BY NAME;
PROC PRINT;
   TITLE 'MASTER2 DATA SET';
```

MASTER2 DATA SET

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	917	DOBBS	F	169.06	272.29
2	917	IG*S		.	.
3	914	LARSON	F	415.47	483.92
4	914	RYAN	M	291.56	399.20
5	914	SON	M	.	.

UPDATE Application

Example: Use the SAS System to update bank accounts. Assume both data sets have already been sorted.

MASTER DATA SET

OBS	ACCOUNT	BALANCE
1	3152617	150
2	3355912	500
3	3999249	100
4	4102477	1500

TRANSACT DATA SET

OBS	ACCOUNT	TYPE	AMOUNT
1	3355912	DEBIT	25
2	4102477	CREDIT	500
3	4102477	DEBIT	1850
4	5023541	CREDIT	100

```
DATA MASTER2;
   UPDATE MASTER TRANSACT;
   BY ACCOUNT;
   IF TYPE='CREDIT' THEN BALANCE+AMOUNT;
   IF TYPE='DEBIT' THEN BALANCE+-AMOUNT;
   KEEP ACCOUNT BALANCE;
PROC PRINT;
   TITLE 'MASTER2 DATA SET';
```

MASTER2 DATA SET

OBS	ACCOUNT	BALANCE
1	3152617	150
2	3355912	475
3	3999249	100
4	4102477	150
5	5023541	100

12.4 Special SAS Variables

Definitions of Special Variables

There are four special variables available in the DATA step that can be very useful in SET, MERGE, and UPDATE applications.

IN *variable*
: indicates which data set contributed to the current observation.

END *variable*
: indicates when the current observation is the last to be processed.

FIRST.*byvariable*
: indicates when the current observation is the first in a BY group.

LAST.*byvariable*
: indicates when the current observation is the last in a BY group.

- The special variables can only take on a value of 0 or 1.
- The special variables are automatically dropped (they appear in the program data vector, but not in the output SAS data set).
- IN and END variables are created and named by the user.
- FIRST.*byvariables* and LAST.*byvariables* are automatically created and named by the SAS System whenever a BY statement is used in the DATA step.

Values of the Special Variables

Example: Display the values of all the special variables when interleaving two data sets.

JAN DATA SET

OBS	MAKE	YEAR	PRICE
1	CHEVY	76	2550
2	CHEVY	72	750
3	CHEVY	81	8200
4	FORD	77	2200
5	FORD	71	500

FEB DATA SET

OBS	MAKE	YEAR	PRICE
1	CHEVY	70	650
2	FORD	74	1200
3	FORD	78	3500
4	OLDS	79	5500

Note: both data sets have already been sorted by MAKE.

```
DATA CARSALES;
   SET JAN(IN=J) FEB(IN=F) END=E;
   BY MAKE;
```

Program Data Vector

CARSALES data set | Special variables

OBS	MAKE	YEAR	PRICE	J	F	E	FIRST. MAKE	LAST. MAKE
1	CHEVY	76	2550	1	0	0	1	0
2	CHEVY	72	750	1	0	0	0	0
3	CHEVY	81	8200	1	0	0	0	0
4	CHEVY	70	650	0	1	0	0	1
5	FORD	77	2200	1	0	0	1	0
6	FORD	71	500	1	0	0	0	0
7	FORD	74	1200	0	1	0	0	0
8	FORD	78	3500	0	1	0	0	1
9	OLDS	79	5500	0	1	1	1	1

The IN= Variable with a SET Statement

Example: Concatenate the DEPT100 and DEPT200 SAS data sets shown below and create a variable that contains the department number of each employee.

DEPT100 DATA SET

OBS	SSN	SALARY
1	178928782	21000
2	265776543	17700
3	344278919	14700

DEPT200 DATA SET

OBS	SSN	SALARY
1	313682992	38000
2	353764567	27400
3	587348727	11300

```
DATA COMPANY;
   SET DEPT100(IN=IN100) DEPT200;
   IF IN100 THEN DEPT=100;
   ELSE DEPT=200;
PROC PRINT;
   TITLE 'COMPANY DATA SET';
```

COMPANY DATA SET

OBS	SSN	SALARY	DEPT
1	178928782	21000	100
2	265776543	17700	100
3	344278919	14700	100
4	313682992	38000	200
5	353764567	27400	200
6	587348727	11300	200

The IN= Variable with a MERGE Statement

Example: The SAS data set, INVOICE, contains invoice data for orders made from an international supply company during the month of January, and the SAS data set, ADDRESS, contains names and addresses for regular customers.

Merge the data sets so that the invoices are matched with the appropriate names and addresses. (Note: both data sets have been sorted.)

The IN= Variable with a MERGE Statement

ADDRESS DATA SET

OBS	NAME	LOCATION
1	ANDERSON	LONDON
2	CAMERON	ROME
3	KESTER	ATHENS
4	MARSTER	ROME
5	PERKINSON	PARIS
6	STECKLER	BONN

INVOICE DATA SET

OBS	NAME	INVNUM	AMOUNT
1	CAMERON	27651	655.98
2	CAMERON	28554	125.78
3	MARSTER	24813	349.03
4	STECKLER	23667	256.78

```
DATA BILLS;
   MERGE ADDRESS(IN=A) INVOICE(IN=I);
   BY NAME;
```

Program Data Vector

BILLS DATA SET

OBS	NAME	LOCATION	INVNUM	AMOUNT	A	I
1	ANDERSON	LONDON	.	.	1	0
2	CAMERON	ROME	27651	655.98	1	1
3	CAMERON	ROME	28554	125.78	1	1
4	KESTER	ATHENS	.	.	1	0
5	MARSTER	ROME	24813	349.03	1	1
6	PERKINSON	PARIS	.	.	1	0
7	STECKLER	BONN	23667	256.78	1	1

Note: the customers who did not place an order (non-matches) are also in the output data set.

The IN= Variable with a MERGE Statement

Example: Repeat the preceding example but keep only customers who need to be sent invoices.

ADDRESS DATA SET

OBS	NAME	LOCATION
1	ANDERSON	LONDON
2	CAMERON	ROME
3	KESTER	ATHENS
4	MARSTER	ROME
5	PERKINSON	PARIS
6	STECKLER	BONN

INVOICE DATA SET

OBS	NAME	INVNUM	AMOUNT
1	CAMERON	27651	655.98
2	CAMERON	28554	125.78
3	MARSTER	24813	349.03
4	STECKLER	23667	256.78

```
DATA BILLS;
   MERGE ADDRESS INVOICE(IN=I);
   BY NAME;
   IF I;
PROC PRINT;
   TITLE 'BILLS DATA SET';
```

BILLS DATA SET

OBS	NAME	LOCATION	INVNUM	AMOUNT
1	CAMERON	ROME	27651	655.98
2	CAMERON	ROME	28554	125.78
3	MARSTER	ROME	24813	349.03
4	STECKLER	BONN	23667	256.78

Note: customers who placed multiple orders appear in the output data set multiple times.

Removing Duplicate BY-Values

Example: Create a data set that contains one observation for each customer to be invoiced by merging the INVOICE and ADDRESS data sets. The data set should contain the name, location and total invoice amount for each customer. Print the data set.

ADDRESS DATA SET

OBS	NAME	LOCATION
1	ANDERSON	LONDON
2	CAMERON	ROME
3	KESTER	ATHENS
4	MARSTER	ROME
5	PERKINSON	PARIS
6	STECKLER	BONN

INVOICE DATA SET

OBS	NAME	INVNUM	AMOUNT
1	CAMERON	27651	655.98
2	CAMERON	28554	125.78
3	MARSTER	24813	349.03
4	STECKLER	23667	256.78

```
DATA LIST;
   MERGE ADDRESS INVOICE(IN=I);
   BY NAME;
   IF I;
   IF FIRST.NAME THEN TOTAL=0;
   TOTAL+AMOUNT;
   IF LAST.NAME;
   KEEP NAME LOCATION TOTAL;
PROC PRINT;
   TITLE 'LIST DATA SET';
```

LIST DATA SET

OBS	NAME	LOCATION	TOTAL
1	CAMERON	ROME	781.76
2	MARSTER	ROME	349.03
3	STECKLER	BONN	256.78

12.5 Exercises

12.1 Given the following two SAS data sets:

SAS Data Set A

ID	X	Y
1	12	11
2	15	.

SAS Data Set B

ID	X	Z
1	.	4
3	17	6
3	18	.

what will be the values of the variables ID, X, Y, and Z in the SAS data sets created by the following DATA steps?

a.
```
DATA;
    SET A B;
```

b.
```
DATA;
    SET A B;
    BY ID;
```

c.
```
DATA;
    MERGE A B;
```

d.
```
DATA;
    MERGE A B;
    BY ID;
```

e.
```
DATA;
    UPDATE A B;
    BY ID;
```

SAS Data Sets

ID	X	Y	Z

Exercises

12.2 A small college stores data in a master file that reflects the status of each of its students. The record layout for the file is described in the table below.

Field Description	Field Position	Variable Name	Type of Data
Name	1-25	NAME	character
Hours of coursework taken before the current semester	26-28	HOURS	numeric
Credits accumulated before the current semester. (A=4 credits/hour, B=3, C=2,D=1,F=0)	29-31	CREDITS	numeric

Exercises

12.2 At the end of the current semester, a second file is created with the following record layout.

Field Description	Field Position	Variable Name	Type of Data
Name	1-25	NAME	character
Hours of coursework taken during the current semester	26-28	SHOURS	numeric
Credits received during the current semester. (A=4 credits/hour, B=3,C=2,D=1,F=0)	29-31	SCREDITS	numeric

Exercises

12.2 **a.** Create a SAS data set from each file. Assume the data are placed in the job stream. Use the variable names shown in the tables on the previous page.

b. The college officials would like to combine the information from the two data sets into a SAS data set named ALL. The data set ALL should contain the variables listed below for all the students. Write the SAS code that will create the data set.

Variable Name		Description
NAME	=	student's name
THOURS	=	HOURS+SHOURS
TCREDITS	=	CREDITS+SCREDITS
OLDGPA	=	CREDITS/HOURS
SEMGPA	=	SCREDITS/SHOURS
NEWGPA	=	TCREDITS/THOURS
CLASS	=	FR if THOURS < 30 SO if 30 <= THOURS < 60 JR if 60 <= THOURS < 90 SR if 90 <= THOURS < 120 GR if THOURS >= 120 (graduating)

c. Print the data set named ALL.

d. Use the ALL data set to compute the average SEMGPA and the average NEWGPA for each CLASS.

Exercises

12.3 A retail sales company keeps track of its gross sales figures on a monthly basis. Data for recent years have been extracted for a management meeting.

a. The data and record format for 1985 sales are given below. Create a SAS data set named SALES85 from the data and plot SALES vs MONTH. Make sure that there is a tick mark for each month on the horizontal axis.

Variable Name	Field Position	Type of Data
YEAR	1-4	numeric
MONTH	6-7	numeric
SALES	9-13	numeric

Raw Data

```
Record
Number     |...+....1....+....2  <---- Field Position

   1       1985  1 12088
   2       1985  2 10928
   3       1985  3  9384
   4       1985  4  9277
   5       1985  5 10542
   6       1985  6 11342
   7       1985  7 10993
   8       1985  8 11633
   9       1985  9 13783
  10       1985 10 13732
  11       1985 11 13926
  12       1985 12 14579
```

Exercises

12.3 **b.** The data and record format for 1986 sales are given below. Create a SAS data set named SALES86 from the data and change the year values from 86 to 1986. Plot SALES vs MONTH. Make sure that there is a tick mark for each month on the horizontal axis.

Variable Name	Field Position	Type of Data
YEAR	1-2	numeric
MONTH	4-5	numeric
SALES	7-11	numeric

Raw Data

Record Number	1...+....1....+....2 ← Field Position
1	86 1 13882
2	86 2 13801
3	86 3 12917
4	86 4 13277
5	86 5 14831
6	86 6 14979
7	86 7 15003
8	86 8 15710
9	86 9 16583
10	86 10 18435
11	86 11 17553
12	86 12 18367

Exercises

12.3 **c.** Superimpose the plots described in parts 'a' and 'b' on the same set of axes. (Hint: The two data sets must be properly combined. Remember that the same variables are in both data sets.)

d. Use the MEANS procedure to determine the average sales figures by month across the two years. (Hint: Appropriately combine the SALES85 and SALES86 data sets.)

Exercises

12.4* A medical clinic keeps its patient data on cards. The cards are separated into two groups. The data recorded on the first group of cards are described below. There is only one card per patient in the first group. Create a SAS data set named PATIENT from these data.

Variable Name	Field Position	Type of Data
SSN	1-9	numeric
NAME	10-40	character
SEX	41	character
BORN	42-49	MMDDYY8. format

The record layout for the second group of cards is shown below. There is a separate card for each patient visit to the clinic in the second group. Create a SAS data set named VISITS from these data.

Variable Name	Field Position	Type of Data
SSN	1-9	numeric
DATE	10-17	MMDDYY8. format
FEE	18-20	numeric

* See Section 12.4.

Exercises

12.4* **a.** Combine these two data sets into a SAS data set named BOTH so that the data in the PATIENT data set are attached to each of the appropriate observations in the VISITS data set. Create a variable named VISIT in the data set that numbers the visits of each individual patient. In other words, if a patient visited the clinic 3 times, then the observations for that patient should be numbered from 1 to 3. Print the data set. Use appropriate formats for social security numbers and for all date variables.

b. Repeat part 'a' but keep only those patients who have an entry in the VISITS data set. Call this data set BILLS.

c. Use the BILLS data set to create a SAS data set named INVOICE. This data set should contain one observation for each patient in the BILLS data set. It should contain the variables SSN and NAME from the BILLS data set plus a variable named TOTAL that shows the total fee for all of the visits that the patient has made. Print the data set and use appropriate formats.

13. Reading and Writing Raw Data

13.1 Introduction

13.2 OS Batch, TSO, and CMS Environments

13.3 VSE Batch and ICCF Environments

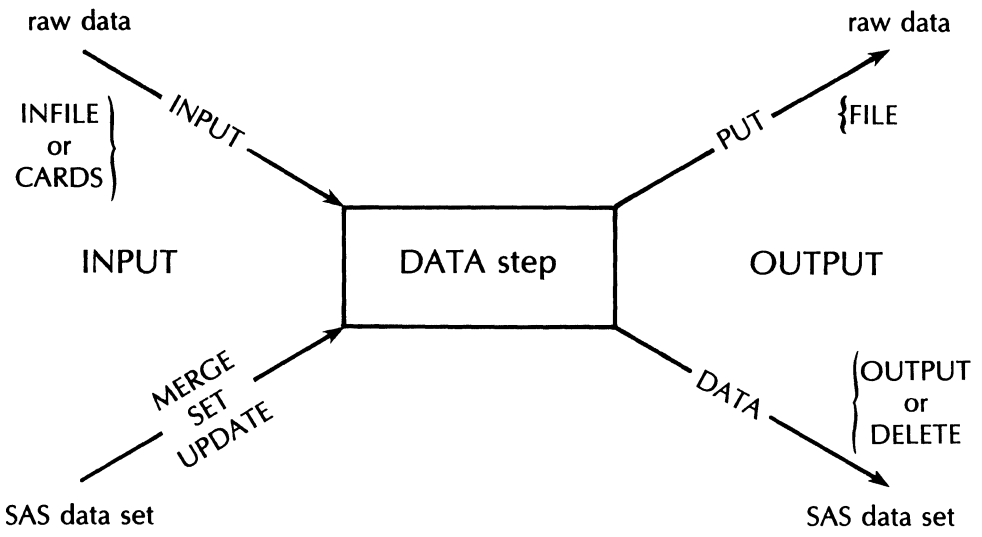
13.4 Minicomputer Environments

13.5 Exercises

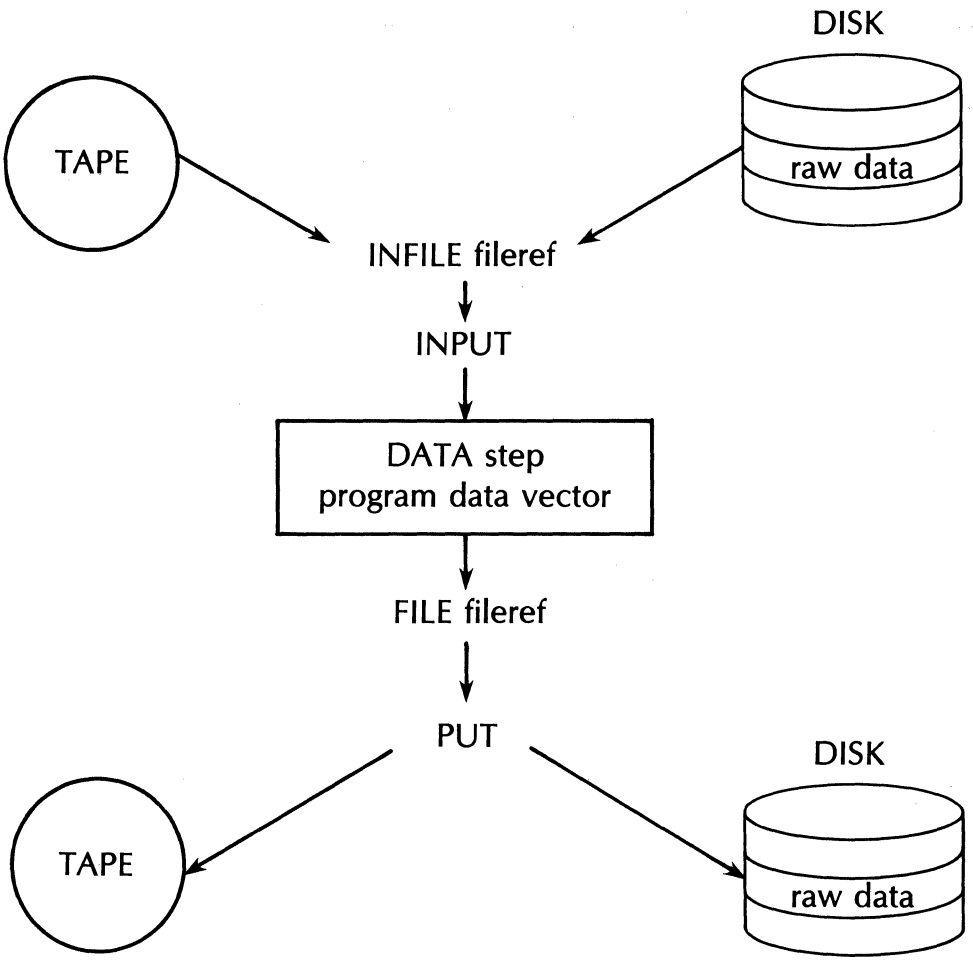
13.1 Introduction

SAS Input and Output Operations

The SAS System has a full range of input and output operations available for both SAS data sets and raw data files.



Reading and Writing Raw Data Files



In-stream Data versus External Files

DATA in JOB STREAM

```
DATA PAYROLL;
   INPUT ID 1-3 WAGE 7-10;
   CARDS;
101      7.63
102      4.21
 .        .
 .        .
PROC PRINT;
```

DATA IN JOB STREAM

OBS	ID	WAGE
1	101	7.63
2	102	4.21
.	.	.
.	.	.
.	.	.

DATA in EXTERNAL FILE

External File

101	7.63
102	4.21
.	.
.	.
.	.

→ fileref=WAGES

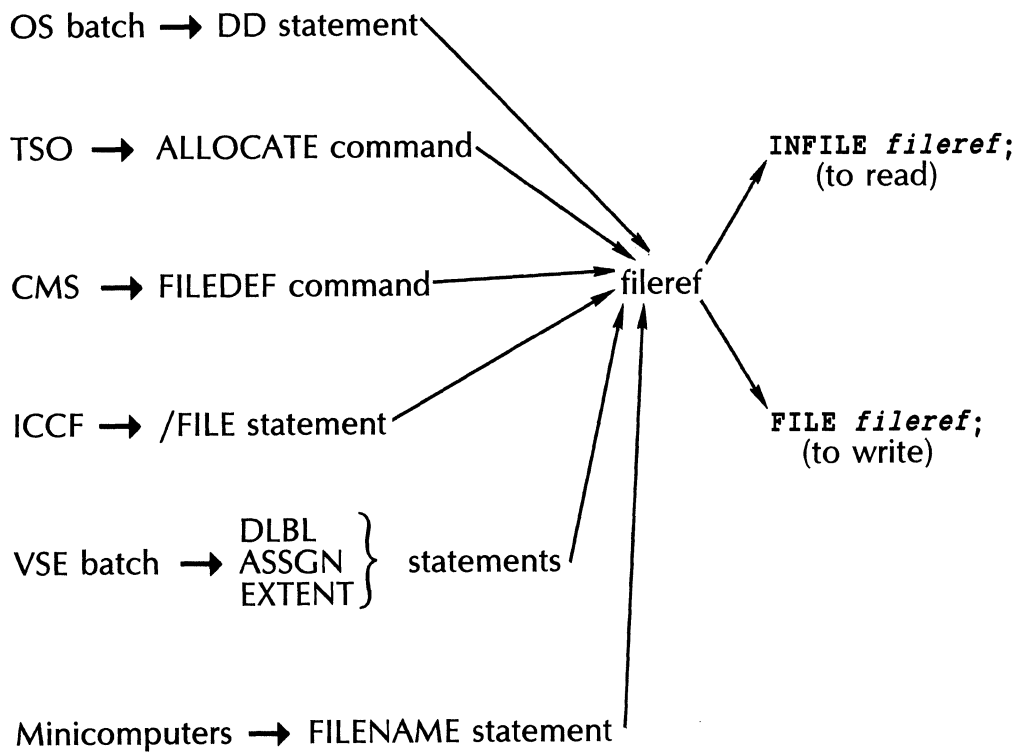
```
DATA PAYROLL;
   INFILE WAGES;
   INPUT ID 1-3 WAGE 7-10;
PROC PRINT;
```

OBS	ID	WAGE
1	101	7.63
2	102	4.21
.	.	.
.	.	.
.	.	.

Pointing to External Files

You use a system command (mainframe environments) or a SAS FILENAME statement (minicomputer environments) to point to the appropriate raw data file.

The SAS System links to the system command through a fileref (file reference name).



Note: the term fileref is equivalent to DDname for mainframe users.

13.2 OS Batch, TSO, and CMS Environments

System Commands

General form of the system commands:

OS batch

```
//fileref DD DSN=OSdatasetname,DISP=xxx,
// UNIT=xxx,LABEL=xxx,VOL=xxx,SPACE=xxx
```

TSO

```
ALLOCATE FILE(fileref) DATASET('OSdatasetname')
         disp unit_of_space SPACE(xxx)


ALLOC F(fileref) DA('OSdatasetname')
      disp unit_of_space SPACE(xxx)
```

CMS

```
FILEDEF fileref DISK filename filetype filemode
```

Reading External Files

To read data values that are stored on disk or tape,

1. use a system command to associate a fileref with the input data file

   System command with a fileref

2. use a DATA statement to begin the DATA step

   DATA ***SASdataset*****;**

3. point to the system command by specifying the same fileref in an INFILE statement in the SAS program

   INFILE ***fileref*****;**

4. indicate how to read the data lines with an INPUT statement.

   INPUT ***variables*****;**

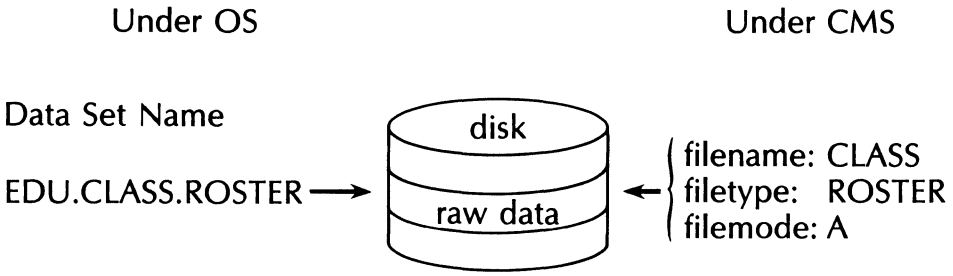
Reading External Files

Example: The data for the class example are stored in a disk file.

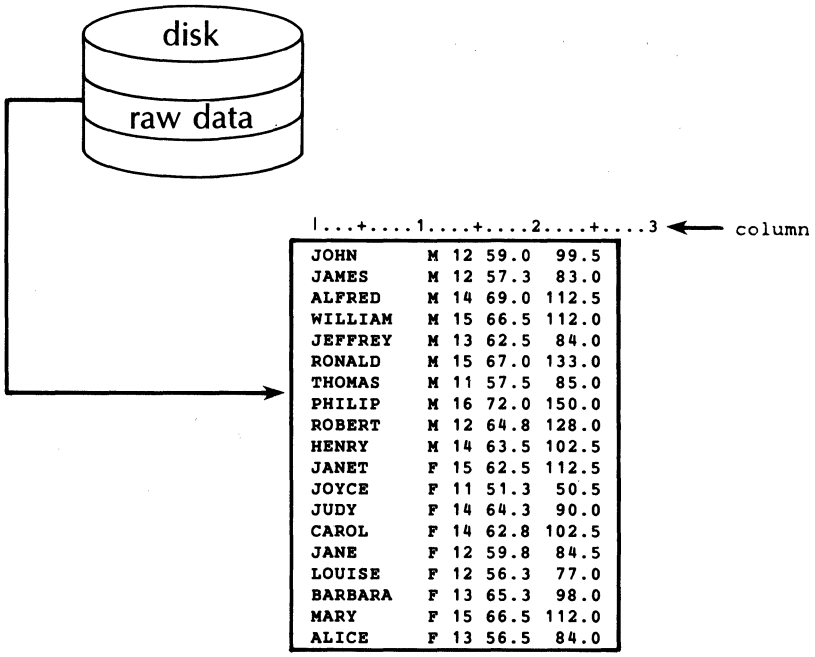
Create a SAS data set and print the data set.

The names of the files are shown below.

A listing of the data is shown on the next page.



Reading External Files




```
|...+....1....+....2....+....3
JOHN       M 12 59.0  99.5
JAMES      M 12 57.3  83.0
ALFRED     M 14 69.0 112.5
WILLIAM    M 15 66.5 112.0
JEFFREY    M 13 62.5  84.0
RONALD     M 15 67.0 133.0
THOMAS     M 11 57.5  85.0
PHILIP     M 16 72.0 150.0
ROBERT     M 12 64.8 128.0
HENRY      M 14 63.5 102.5
JANET      F 15 62.5 112.5
JOYCE      F 11 51.3  50.5
JUDY       F 14 64.3  90.0
CAROL      F 14 62.8 102.5
JANE       F 12 59.8  84.5
LOUISE     F 12 56.3  77.0
BARBARA    F 13 65.3  98.0
MARY       F 15 66.5 112.0
ALICE      F 13 56.5  84.0
```

Reading External Files

Select a system command with fileref=STUDENT.

OS batch

```
//STUDENT DD DSN=EDU.CLASS.ROSTER,DISP=SHR
```

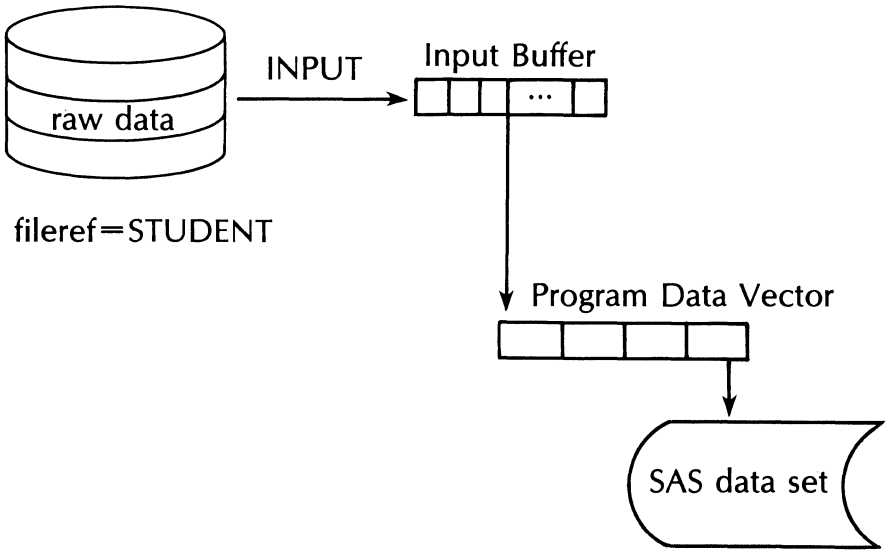
TSO

```
ALLOC F(STUDENT) DA('EDU.CLASS.ROSTER') SHR
```

CMS

```
FILEDEF STUDENT DISK CLASS ROSTER A
```

```
DATA CLASS;
   INFILE STUDENT;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
PROC PRINT DATA=CLASS;
```



Reading External Files

Notes in the SAS log provide information about the input data file.

Partial SAS Log

```
1          DATA CLASS;
2             INFILE STUDENT;
3             INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
4                   HEIGHT 16-19 WEIGHT 21-25;
```

OS batch and TSO

```
NOTE: INFILE STUDENT IS:
      DSNAME=EDU.CLASS.ROSTER,
      UNIT=DISK,VOL=SER=EDU999,DISP=SHR,
      DCB=(BLKSIZE=9040,LRECL=80,RECFM=FB)

NOTE: 19 LINES WERE READ FROM INFILE STUDENT.
NOTE: DATA SET WORK.CLASS HAS 19 OBSERVATIONS AND 5 VARIABLES. 5
15 OBS/TRK
```

CMS

```
NOTE: INFILE STUDENT IS FILE CLASS ROSTER A1
NOTE: 19 LINES WERE READ FROM INFILE STUDENT.
NOTE: DATA SET WORK.CLASS HAS 19 OBSERVATIONS AND 5 VARIABLES.
```

Reading External Files

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0
3	ALFRED	M	14	69.0	112.5
4	WILLIAM	M	15	66.5	112.0
5	JEFFREY	M	13	62.5	84.0
6	RONALD	M	15	67.0	133.0
7	THOMAS	M	11	57.5	85.0
8	PHILIP	M	16	72.0	150.0
9	ROBERT	M	12	64.8	128.0
10	HENRY	M	14	63.5	102.5
11	JANET	F	15	62.5	112.5
12	JOYCE	F	11	51.3	50.5
13	JUDY	F	14	64.3	90.0
14	CAROL	F	14	62.8	102.5
15	JANE	F	12	59.8	84.5
16	LOUISE	F	12	56.3	77.0
17	BARBARA	F	13	65.3	98.0
18	MARY	F	15	66.5	112.0
19	ALICE	F	13	56.5	84.0

Reading External Files

Example: Payroll data are stored in a disk file. The name of the file is

EDU.COMPANY.PAYROLL under OS batch and TSO

COMPANY PAYROLL A under CMS.

Create a SAS data set named SALARIES from the raw data and print the data set.

Select a system command with fileref=MONEY.

OS batch

```
//MONEY DD DSN=EDU.COMPANY.PAYROLL,DISP=SHR
```

TSO

```
ALLOC F(MONEY) DA('EDU.COMPANY.PAYROLL') SHR
```

CMS

```
FILEDEF MONEY DISK COMPANY PAYROLL A


DATA SALARIES;
   INFILE MONEY;
   INPUT BRANCH $ 1-11 DEPT 13-15 SEX $ 17
         GROSSPAY 19-25;
PROC PRINT;
```

Reading External Files

Notes in the SAS log provide information about the input data file.

Partial SAS Log

```
1          DATA SALARIES;
2             INFILE MONEY;
3             INPUT BRANCH $ 1-11 DEPT 13-15 SEX $ 17
4                   GROSSPAY 19-25;
```

OS batch and TSO

```
NOTE: INFILE MONEY IS:
      DSNAME=EDU.COMPANY.PAYROLL,
      UNIT=DISK,VOL=SER=EDU999,DISP=SHR,
      DCB=(BLKSIZE=9040,LRECL=80,RECFM=FB)

NOTE: 21 LINES WERE READ FROM INFILE MONEY.
NOTE: DATA SET WORK.SALARIES HAS 21 OBSERVATIONS AND 4 VARIABLES
. 595 OBS/TRK
```

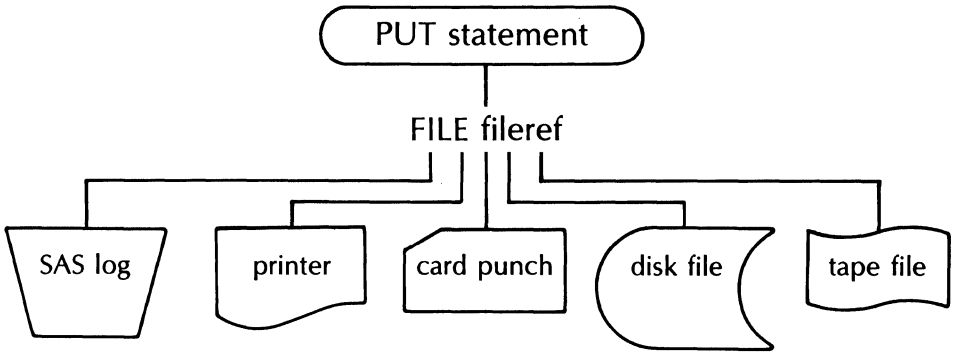
CMS

```
NOTE: INFILE MONEY IS FILE COMPANY PAYROLL A1
NOTE: 21 LINES WERE READ FROM INFILE MONEY.
NOTE: DATA SET WORK.SALARIES HAS 21 OBSERVATIONS AND 4 VARIABLES.
```

Reading External Files

OBS	BRANCH	DEPT	SEX	GROSSPAY
1	RALEIGH	901	F	121.95
2	RALEIGH	901	M	313.60
3	LOS ANGELES	901	F	242.40
4	RALEIGH	901	M	292.00
5	RALEIGH	901	M	243.95
6	LOS ANGELES	901	M	344.78
7	LOS ANGELES	901	M	279.36
8	RALEIGH	901	F	405.37
9	RALEIGH	911	M	399.20
10	RALEIGH	911	M	180.72
11	RALEIGH	911	M	297.56
12	LOS ANGELES	911	M	385.47
13	LOS ANGELES	911	M	402.12
14	LOS ANGELES	911	F	728.12
15	RALEIGH	911	F	272.29
16	LOS ANGELES	911	M	789.00
17	LOS ANGELES	921	F	842.03
18	LOS ANGELES	921	M	279.22
19	LOS ANGELES	921	M	352.84
20	RALEIGH	921	M	1235.46
21	RALEIGH	921	F	1158.22

Writing External Files



The FILE and PUT statements are analogous to the INFILE and INPUT statements.

The FILE statement points to an output file by matching the fileref in a system command (INFILE points to an input file).

FILE *fileref*;

Special filerefs are

LOG for writing to the SAS log

PRINT for writing to the print file (report writing)

PUNCH for writing to the punch file (punching cards).

The PUT statement specifies the output record format (INPUT specifies the input record format).

PUT *variables formats pointer controls 'literals'*;

Writing External Files

To write raw data values to disk or tape,

1. use a system command to associate a fileref with the output file

   System command with a fileref

2. use a DATA statement to begin the DATA step

   DATA _NULL_;

3. read the data

   SET *SASdataset*; (or INPUT if reading raw data)

4. point to the system command by specifying the same fileref in a FILE statement in the SAS program

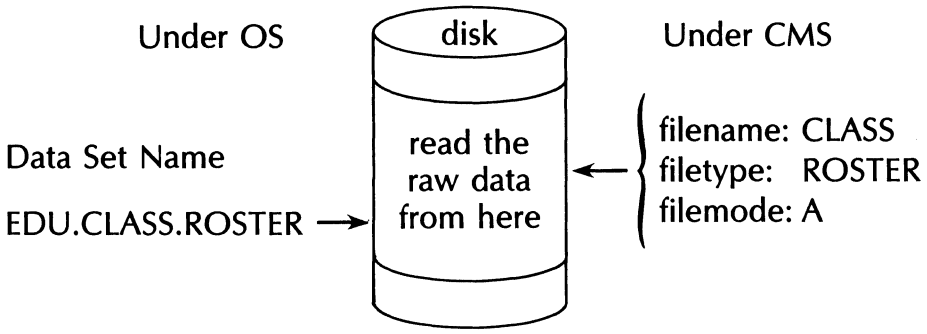
   FILE *fileref*;

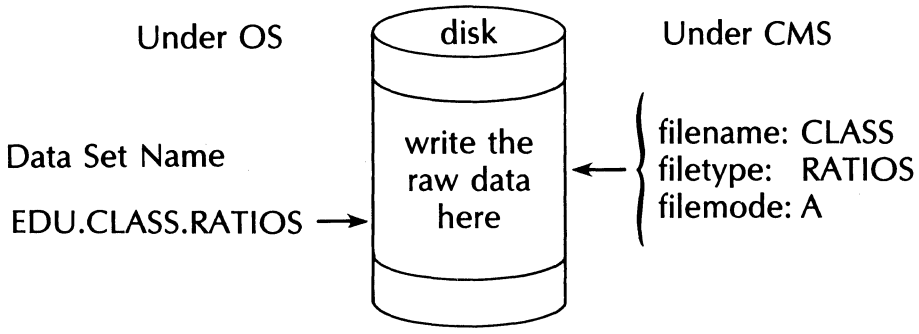
5. indicate how to write the data lines with a PUT statement.

   PUT *variables formats pointer controls 'literals'*;

Writing External Files

Example: Read the data for the class example from an external file, compute the ratio of height to weight, and write the values of the variables NAME, SEX, and RATIO (4 decimal places) to another external file.





Writing External Files

Select a system command with fileref=OLD for the input file and fileref=PUPIL for the output file.

OS batch

```
//OLD DD DSN=EDU.CLASS.ROSTER,DISP=SHR
//PUPIL DD DSN=EDU.CLASS.RATIOS,UNIT=DISK,
// DISP=(NEW,CATLG),VOL=SER=EDU999,SPACE=(TRK,(2))
```

TSO

```
ALLOC F(OLD) DA('EDU.CLASS.ROSTER') SHR
ALLOC F(PUPIL) DA('EDU.CLASS.RATIOS') NEW TRACKS SPACE(2)
```

CMS

```
FILEDEF OLD DISK CLASS ROSTER A
FILEDEF PUPIL DISK CLASS RATIOS A


DATA _NULL_;
   INFILE OLD;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   RATIO=HEIGHT/WEIGHT;
   FILE PUPIL;
   PUT NAME $8. @10 SEX $1. @12 RATIO 6.4;
```

Note: the name _NULL_ in the DATA statement tells the SAS System to build a program data vector and process the observations without creating a SAS data set.

Writing External Files

Notes in the SAS log provide information about the output data file.

Partial SAS Log

```
1          DATA _NULL_;
2             INFILE OLD;
3             INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
4                   HEIGHT 16-19 WEIGHT 21-25;
5             RATIO=HEIGHT/WEIGHT;
6             FILE PUPIL;
7             PUT NAME $8. @10 SEX $1. @12 RATIO 6.4;
```

OS batch and TSO

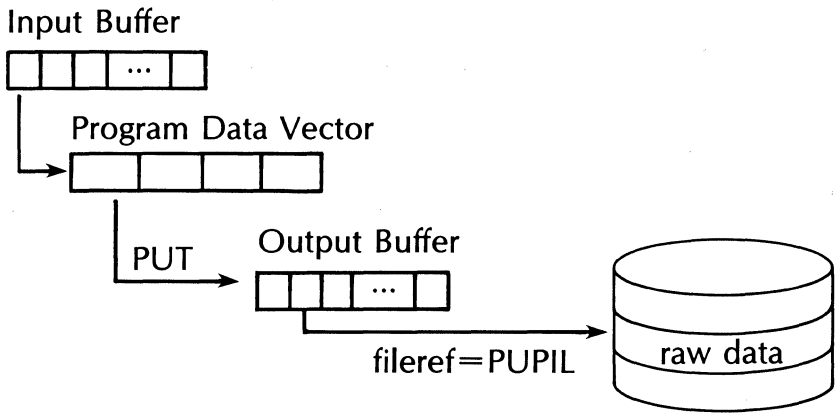
```
NOTE: INFILE OLD IS:
      DSNAME=EDU.CLASS.ROSTER,
      UNIT=DISK,VOL=SER=EDU999,DISP=SHR,
      DCB=(BLKSIZE=9040,LRECL=80,RECFM=FB)

NOTE: FILE PUPIL IS:
      DSNAME=EDU.CLASS.RATIOS,
      UNIT=DISK,VOL=SER=EDU999,DISP=NEW,
      DCB=(BLKSIZE=9040,LRECL=80,RECFM=FB)

NOTE: 19 LINES WERE READ FROM INFILE OLD.
NOTE: 19 LINES WERE WRITTEN TO FILE PUPIL.
```

CMS

```
NOTE: FILE OLD IS CLASS ROSTER A1
NOTE: 19 LINES WERE READ FROM INFILE OLD.

NOTE: FILE PUPIL IS CLASS RATIOS A1
NOTE: 19 LINES WERE WRITTEN TO FILE PUPIL.
```

Writing External Files

The data written to the disk file are shown below.




```
|...+....1....+....2
JOHN      M 0.5930
JAMES     M 0.6904
ALFRED    M 0.6133
WILLIAM   M 0.5938
JEFFREY   M 0.7440
RONALD    M 0.5038
THOMAS    M 0.6765
PHILIP    M 0.4800
ROBERT    M 0.5062
HENRY     M 0.6195
JANET     F 0.5556
JOYCE     F 1.0158
JUDY      F 0.7144
CAROL     F 0.6127
JANE      F 0.7077
LOUISE    F 0.7312
BARBARA   F 0.6663
MARY      F 0.5938
ALICE     F 0.6726
```

← Column

Writing to the SAS Log

Example: The raw data for students 12 to 16 years old in a summer camp are in a disk file with the name

EDU.CAMP.DATA under OS batch and TSO

CAMP DATA A under CMS.

Write a SAS program to read the data and write a message to the SAS log when invalid age data are encountered.

Select a system command with fileref=INCAMP.

OS batch

```
//INCAMP DD DSN=EDU.CAMP.DATA,DISP=SHR
```

TSO

```
ALLOC F(INCAMP) DA('EDU.CAMP.DATA') SHR
```

CMS

```
FILEDEF INCAMP DISK CAMP DATA A
```

```
DATA SUMMER;
   INFILE INCAMP;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   IF AGE<12 OR AGE>16 THEN
      PUT 'INVALID AGE' +1 _N_= NAME= AGE=;
```

Writing to the SAS Log

The information specified in the PUT statement is written to the SAS log when an invalid age is encountered.

Partial SAS Log

```
1          DATA SUMMER;
2             INFILE INCAMP;
3             INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
4                   HEIGHT 16-19 WEIGHT 21-25;
5             IF AGE<12 OR AGE>16 THEN
6                PUT 'INVALID AGE' +1 _N_= NAME= AGE=;
```

OS batch and TSO

```
NOTE: INFILE INCAMP IS:
      DSNAME=EDU.CAMP.DATA,
      UNIT=DISK,VOL=SER=EDU999,DISP=SHR,
      DCB=(BLKSIZE=9040,LRECL=80,RECFM=FB)
```

CMS

```
NOTE: INFILE INCAMP IS CAMP DATA A1
```

```
INVALID AGE _N_=3 NAME=ALFRED AGE=17
INVALID AGE _N_=5 NAME=THOMAS AGE=17
INVALID AGE _N_=11 NAME=CAROL AGE=18
INVALID AGE _N_=15 NAME=ALICE AGE=17

NOTE: 15 LINES WERE READ FROM INFILE INCAMP.
NOTE: DATA SET WORK.SUMMER HAS 15 OBSERVATIONS AND 5 VARIABLES.
```

Writing to the Print File

Example: The raw data for a company payroll are in a disk file with the name

EDU.COMPANY.PAYROLL under OS batch and TSO

COMPANY PAYROLL A under CMS.

Write a SAS program that uses FILE and PUT statements to write a payroll report.

Select a system command with fileref=PAY to read the input file.

OS batch

```
//PAY DD DSN=EDU.COMPANY.PAYROLL,DISP=SHR
```

TSO

```
ALLOC F(PAY) DA('EDU.COMPANY.PAYROLL') SHR
```

CMS

```
FILEDEF PAY DISK COMPANY PAYROLL A
```

```
DATA _NULL_;
   INFILE PAY;
   INPUT BRANCH $ 1-11 DEPT 13-15 SEX $ 17
         GROSSPAY 19-25;
   FILE PRINT;
   PUT @15 BRANCH $11. @30 DEPT 3. @37 SEX $1.
       @45 GROSSPAY DOLLAR9.2;
```

Writing to the Print File

FILE PRINT sends the generated report to the same print file that receives procedure output.

```
RALEIGH        901    F         $121.95
RALEIGH        901    M         $313.60
LOS ANGELES    901    F         $242.40
RALEIGH        901    M         $292.00
RALEIGH        901    M         $243.95
LOS ANGELES    901    M         $344.78
LOS ANGELES    901    M         $279.36
RALEIGH        901    F         $405.37
RALEIGH        911    M         $399.20
RALEIGH        911    M         $180.72
RALEIGH        911    M         $297.56
LOS ANGELES    911    M         $385.47
LOS ANGELES    911    M         $402.12
LOS ANGELES    911    F         $728.12
RALEIGH        911    F         $272.29
LOS ANGELES    911    M         $789.00
LOS ANGELES    921    F         $842.03
LOS ANGELES    921    M         $279.22
LOS ANGELES    921    M         $352.84
RALEIGH        921    M       $1,235.46
RALEIGH        921    F       $1,158.22
```

13.3 VSE Batch and ICCF Environments

System Commands

General form of the system commands:

VSE batch (VSE disk files)

```
// DLBL fileref,'file-ID',date
// EXTENT SYSnnn,volser,type,seq,begin,tracks|blocks
// ASSGN logicalunit,device,VOL=volser,SHR
```

ICCF (VSE disk files)

```
/FILE NAME=fileref,ID='file-ID',UNITS=SYSnnn,
/ SERIAL=volume,DATE=n,LOC=begin,number
```

ICCF (ICCF library members)

```
/FILE NAME=member,UNITS=SYSnnn,TYPE=ICCF
```

Note: see the *SAS Companion for the VSE Operating System* for information on accessing data in an ICCF library member and for additional information on accessing VSE files.

Reading External Files

To read data values that are stored on disk or tape,

1. use a system command to associate a fileref with the input data file

   ```
   System command with a fileref
   ```

2. use a DATA statement to begin the DATA step

   ```
   DATA SASdataset;
   ```

3. point to the system command by specifying the same fileref in an INFILE statement in the SAS program

   ```
   INFILE fileref RECFM=recfm
          BLKSIZE=blksize LRECL=lrecl;
   ```

4. indicate how to read the data lines with an INPUT statement.

   ```
   INPUT variables;
   ```

Note: the RECFM, LRECL, and BLKSIZE of the file must be specified in the INFILE statement.

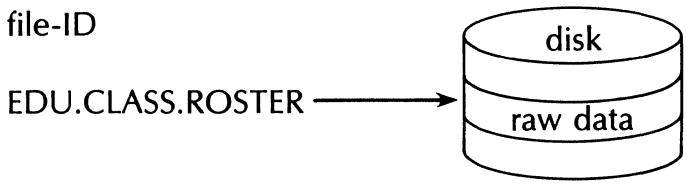
Reading External Files

Example: The data for the class example are stored in a disk file.

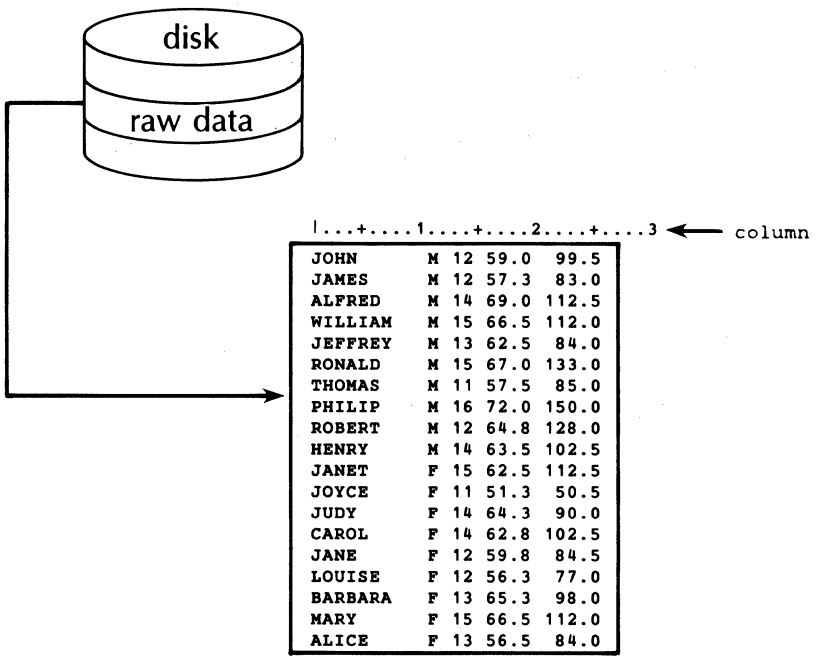
Create a SAS data set and print the data set.

The file-ID of the file is shown below.

A listing of the data is shown on the next page.



Reading External Files




```
1...+....1....+....2....+....3
JOHN       M 12 59.0  99.5
JAMES      M 12 57.3  83.0
ALFRED     M 14 69.0 112.5
WILLIAM    M 15 66.5 112.0
JEFFREY    M 13 62.5  84.0
RONALD     M 15 67.0 133.0
THOMAS     M 11 57.5  85.0
PHILIP     M 16 72.0 150.0
ROBERT     M 12 64.8 128.0
HENRY      M 14 63.5 102.5
JANET      F 15 62.5 112.5
JOYCE      F 11 51.3  50.5
JUDY       F 14 64.3  90.0
CAROL      F 14 62.8 102.5
JANE       F 12 59.8  84.5
LOUISE     F 12 56.3  77.0
BARBARA    F 13 65.3  98.0
MARY       F 15 66.5 112.0
ALICE      F 13 56.5  84.0
```

Reading External Files

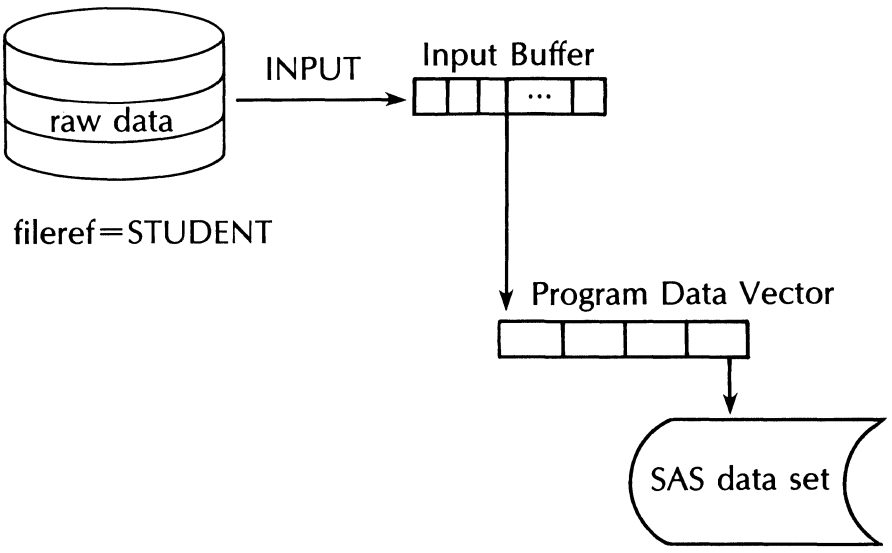
Select a system command with fileref=STUDENT.

VSE batch

```
// DLBL STUDENT,'EDU.CLASS.ROSTER'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
```

ICCF

```
/FILE NAME=STUDENT,ID='EDU.CLASS.ROSTER',
/ UNITS=SYS055,SERIAL=DOSRES


DATA CLASS;
   INFILE STUDENT RECFM=FB LRECL=80 BLKSIZE=9040;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
PROC PRINT DATA=CLASS;
```



Note: under ICCF, the logical unit SYS055 must be assigned to DOSRES in the ICCF start-up job.

Reading External Files

Notes in the SAS log provide information about the input data file.

Partial SAS Log

```
1          DATA CLASS;
2             INFILE STUDENT RECFM=FB LRECL=80 BLKSIZE=9040;
3             INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
4                   HEIGHT 16-19 WEIGHT 21-25;

NOTE: INFILE STUDENT HAS THE FOLLOWING CHARACTERISTICS:
      DCB=(BLKSIZE=9040,LRECL=80,RECFM=FB)
NOTE: 19 LINES WERE READ FROM INFILE STUDENT.
NOTE: DATA SET WORK.CLASS HAS 19 OBSERVATIONS AND 5 VARIABLES. 1268 OBS/
TRK.
```

Reading External Files

SAS

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0
3	ALFRED	M	14	69.0	112.5
4	WILLIAM	M	15	66.5	112.0
5	JEFFREY	M	13	62.5	84.0
6	RONALD	M	15	67.0	133.0
7	THOMAS	M	11	57.5	85.0
8	PHILIP	M	16	72.0	150.0
9	ROBERT	M	12	64.8	128.0
10	HENRY	M	14	63.5	102.5
11	JANET	F	15	62.5	112.5
12	JOYCE	F	11	51.3	50.5
13	JUDY	F	14	64.3	90.0
14	CAROL	F	14	62.8	102.5
15	JANE	F	12	59.8	84.5
16	LOUISE	F	12	56.3	77.0
17	BARBARA	F	13	65.3	98.0
18	MARY	F	15	66.5	112.0
19	ALICE	F	13	56.5	84.0

Reading External Files

Example: Payroll data are stored in a disk file. The file-ID is

EDU.COMPANY.PAYROLL under VSE.

Create a SAS data set named SALARIES from the raw data and print the data set.

Select a system command with fileref=MONEY.

VSE batch

```
// DLBL MONEY,'EDU.COMPANY.PAYROLL'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
```

ICCF

```
/FILE NAME=MONEY,ID='EDU.COMPANY.PAYROLL',
/ UNITS=SYS055,SERIAL=DOSRES


DATA SALARIES;
   INFILE MONEY RECFM=FB LRECL=80 BLKSIZE=9040;
   INPUT BRANCH $ 1-11 DEPT 13-15 SEX $ 17
         GROSSPAY 19-25;
PROC PRINT;
```

Note: under ICCF, the logical unit SYS055 must be assigned to DOSRES in the ICCF start-up job.

Reading External Files

Notes in the SAS log provide information about the input data file.

Partial SAS Log

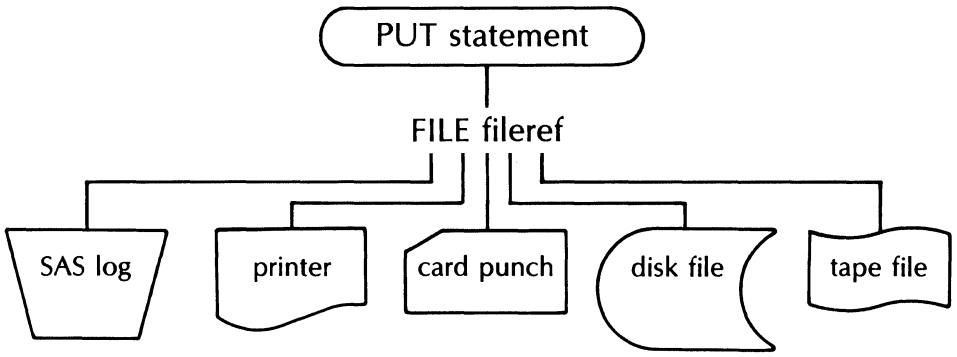
```
1          DATA SALARIES;
2             INFILE MONEY RECFM=FB LRECL=80 BLKSIZE=9040;
3             INPUT BRANCH $ 1-11 DEPT 13-15 SEX $ 17
4                   GROSSPAY 19-25;

NOTE: INFILE MONEY HAS THE FOLLOWING CHARACTERISTICS:
      DCB=(BLKSIZE=9040,LRECL=80,RECFM=FB)
NOTE: 21 LINES WERE READ FROM INFILE MONEY.
NOTE: DATA SET WORK.SALARIES HAS 21 OBSERVATIONS AND 4 VARIABLES. 1466
OBS/TRK.
```

Reading External Files

SAS

OBS	BRANCH	DEPT	SEX	GROSSPAY
1	RALEIGH	901	F	121.95
2	RALEIGH	901	M	313.60
3	LOS ANGELES	901	F	242.40
4	RALEIGH	901	M	292.00
5	RALEIGH	901	M	243.95
6	LOS ANGELES	901	M	344.78
7	LOS ANGELES	901	M	279.36
8	RALEIGH	901	F	405.37
9	RALEIGH	911	M	399.20
10	RALEIGH	911	M	180.72
11	RALEIGH	911	M	297.56
12	LOS ANGELES	911	M	385.47
13	LOS ANGELES	911	M	402.12
14	LOS ANGELES	911	F	728.12
15	RALEIGH	911	F	272.29
16	LOS ANGELES	911	M	789.00
17	LOS ANGELES	921	F	842.03
18	LOS ANGELES	921	M	279.22
19	LOS ANGELES	921	M	352.84
20	RALEIGH	921	M	1235.46
21	RALEIGH	921	F	1158.22

Writing External Files



The FILE and PUT statements are analogous to the INFILE and INPUT statements.

The FILE statement points to an output file by matching the fileref in a system command (INFILE points to an input file).

FILE *fileref* **RECFM=***recfm* **LRECL=***lrecl* **BLKSIZE=***blksize*;

Special filerefs are

LOG	for writing to the SAS log
PRINT	for writing to the print file (report writing)
PUNCH	for writing to the punch file (punching cards).

The PUT statement specifies the output record format (INPUT specifies the input record format).

PUT *variables formats pointer controls 'literals'*;

Writing External Files

To write raw data values to disk or tape,

1. use a system command to associate a fileref with the output file

   ```
   System command with a fileref
   ```

2. use a DATA statement to begin the DATA step

   ```
   DATA _NULL_;
   ```

3. read the data

   ```
   SET SASdataset;
   ```
   (or INPUT if reading raw data)

4. point to the system command by specifying the same fileref in a FILE statement in the SAS program

   ```
   FILE fileref RECFM=recfm LRECL=lrecl BLKSIZE=blksize;
   ```

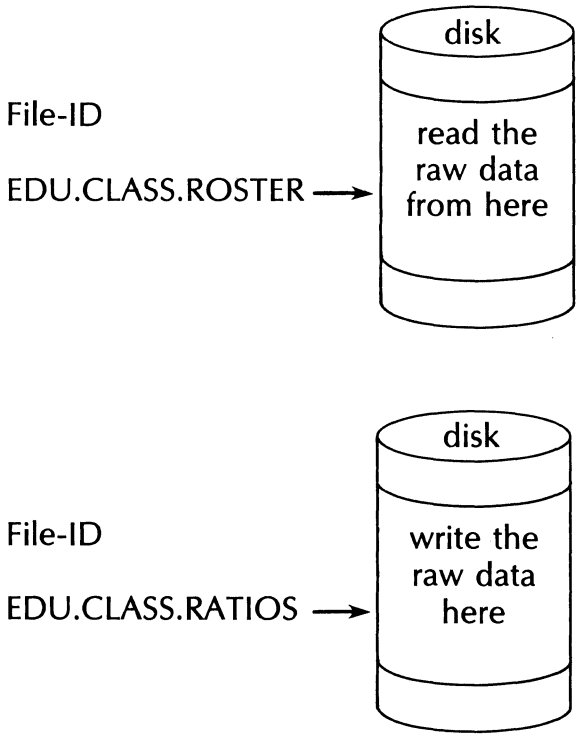
5. indicate how to write the data lines with a PUT statement.

   ```
   PUT variables formats pointer controls 'literals';
   ```

Note: the RECFM, LRECL, and BLKSIZE of the file must be specified in the FILE statement.

Writing External Files

Example: Read the data for the class example from an external file, compute the ratio of height to weight, and write the values of the variables NAME, SEX, and RATIO (4 decimal places) to another external file.



Writing External Files

Select a system command with fileref=OLD for the input file and fileref=PUPIL for the output file.

VSE batch

```
// DLBL OLD,'EDU.CLASS.ROSTER'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
// DLBL PUPIL,'EDU.CLASS.RATIOS',99/365
// EXTENT SYS055,DOSRES,1,0,9947,1
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
```

ICCF

```
/FILE NAME=OLD,ID='EDU.CLASS.ROSTER',UNITS=SYS055,
/ SERIAL=DOSRES
/FILE NAME=PUPIL,ID='EDU.CLASS.RATIOS',UNITS=SYS055,
/ SERIAL=DOSRES,DATE=99/365,LOC=9947,1


DATA _NULL_;
   INFILE OLD RECFM=FB LRECL=80 BLKSIZE=9040;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   RATIO=HEIGHT/WEIGHT;
   FILE PUPIL RECFM=FB LRECL=80 BLKSIZE=9040;
   PUT NAME $8. @10 SEX $1. @12 RATIO 6.4;
```

Note: the name _NULL_ in the DATA statement tells the SAS System to build a program data vector and process the observations without creating a SAS data set.

Under ICCF, the logical unit SYS055 must be assigned to DOSRES in the ICCF start-up job.

Writing External Files

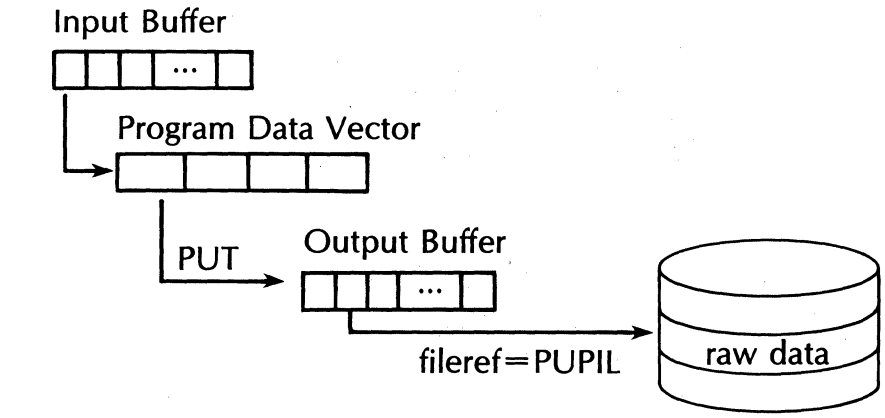
Notes in the SAS log provide information about the output data file.

Partial SAS Log

```
1          DATA _NULL_;
2             INFILE OLD RECFM=FB LRECL=80 BLKSIZE=9040;
3             INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
4                   HEIGHT 16-19 WEIGHT 21-25;
5             RATIO=HEIGHT/WEIGHT;
6             FILE PUPIL RECFM=FB LRECL=80 BLKSIZE=9040;
7             PUT NAME $8. @10 SEX $1. @12 RATIO 6.4;

NOTE: INFILE OLD HAS THE FOLLOWING CHARACTERISTICS:
      DCB=(BLKSIZE=9040,LRECL=80,RECFM=FB)
NOTE: FILE PUPIL HAS THE FOLLOWING CHARACTERISTICS:
      DCB=(BLKSIZE=9040,LRECL=80,RECFM=FB)
NOTE: 19 LINES WERE READ FROM INFILE OLD.
NOTE: 19 LINES WERE WRITTEN TO FILE PUPIL.
```

Writing External Files

The data written to the disk file are shown below.




```
|...+....1....+....2
JOHN      M 0.5930
JAMES     M 0.6904
ALFRED    M 0.6133
WILLIAM   M 0.5938
JEFFREY   M 0.7440
RONALD    M 0.5038
THOMAS    M 0.6765
PHILIP    M 0.4800
ROBERT    M 0.5062
HENRY     M 0.6195
JANET     F 0.5556
JOYCE     F 1.0158
JUDY      F 0.7144
CAROL     F 0.6127
JANE      F 0.7077
LOUISE    F 0.7312
BARBARA   F 0.6663
MARY      F 0.5938
ALICE     F 0.6726
```

← Column

Writing to the SAS Log

Example: The raw data for students 12 to 16 years old in a summer camp are in a disk file with file-ID

EDU.CAMP.DATA under VSE.

Write a SAS program to read the data and write a message to the SAS log when invalid age data are encountered.

Select a system command with fileref=INCAMP.

VSE batch

```
// DLBL INCAMP,'EDU.CAMP.DATA'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
```

ICCF

```
/FILE NAME=INCAMP,ID='EDU.CAMP.DATA',UNITS=SYS055,
/ SERIAL=DOSRES


DATA SUMMER;
   INFILE INCAMP RECFM=FB LRECL=80 BLKSIZE=9040;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   IF AGE<12 OR AGE>16 THEN
      PUT 'INVALID AGE' +1 _N_= NAME= AGE=;
```

Note: under ICCF, the logical unit SYS055 must be assigned to DOSRES in the ICCF start-up job.

Writing to the SAS Log

The information specified on the PUT statement is written to the SAS log when an invalid age is encountered.

Partial SAS Log

```
1          DATA SUMMER;
2             INFILE INCAMP RECFM=FB LRECL=80 BLKSIZE=9040;
3             INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
4                   HEIGHT 16-19 WEIGHT 21-25;
5             IF AGE<12 OR AGE>16 THEN
6                PUT 'INVALID AGE' +1 _N_= NAME= AGE=;

NOTE: INFILE INCAMP HAS THE FOLLOWING CHARACTERISTICS:
      DCB=(BLKSIZE=9040,LRECL=80,RECFM=FB)
INVALID AGE _N_=3 NAME=ALFRED AGE=17
INVALID AGE _N_=5 NAME=THOMAS AGE=17
INVALID AGE _N_=11 NAME=CAROL AGE=18
INVALID AGE _N_=15 NAME=ALICE AGE=17
NOTE: 15 LINES WERE READ FROM INFILE INCAMP.
NOTE: DATA SET WORK.SUMMER HAS 15 OBSERVATIONS AND 5 VARIABLES. 1268 OBS
/TRK.
```

Writing to the Print File

Example: The raw data for a company payroll are in a disk file with file-ID

EDU.COMPANY.PAYROLL under VSE.

Write a SAS program that uses FILE and PUT statements to write a payroll report.

Select a system command with fileref=PAY to read the input file.

VSE batch

```
// DLBL PAY,'EDU.COMPANY.PAYROLL'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
```

ICCF

```
/FILE NAME=PAY,ID='EDU.COMPANY.PAYROLL',UNITS=SYS055,
/ SERIAL=DOSRES


DATA _NULL_;
   INFILE PAY RECFM=FB LRECL=80 BLKSIZE=9040;
   INPUT BRANCH $ 1-11 DEPT 13-15 SEX $ 17
         GROSSPAY 19-25;
   FILE PRINT;
   PUT @15 BRANCH $11. @30 DEPT 3. @37 SEX $1.
       @45 GROSSPAY DOLLAR9.2;
```

Note: under ICCF, the logical unit SYS055 must be assigned to DOSRES in the ICCF start-up job.

Writing to the Print File

FILE PRINT sends the generated report to the same print file that receives procedure output.

```
                              SAS

RALEIGH          901    F          $121.95
RALEIGH          901    M          $313.60
LOS ANGELES      901    F          $242.40
RALEIGH          901    M          $292.00
RALEIGH          901    M          $243.95
LOS ANGELES      901    M          $344.78
LOS ANGELES      901    M          $279.36
RALEIGH          901    F          $405.37
RALEIGH          911    M          $399.20
RALEIGH          911    M          $180.72
RALEIGH          911    M          $297.56
LOS ANGELES      911    M          $385.47
LOS ANGELES      911    M          $402.12
LOS ANGELES      911    F          $728.12
RALEIGH          911    F          $272.29
LOS ANGELES      911    M          $789.00
LOS ANGELES      921    F          $842.03
LOS ANGELES      921    M          $279.22
LOS ANGELES      921    M          $352.84
RALEIGH          921    M        $1,235.46
RALEIGH          921    F        $1,158.22
```

13.4 Minicomputer Environments

Pointing to External Files

A name must be specified in the INFILE and FILE statements to point the SAS System to the appropriate external data file.

There are two methods for matching this name with the name of the file you want to access.

- Use the actual filename in the SAS statements (restrictions apply).

  INFILE ***filename***; (to read)

  FILE ***filename***; (to write)

- Define a file reference name (fileref) for the file and specify the fileref in the SAS statements.

  INFILE ***fileref***; (to read)

  FILE ***fileref***; (to write)

Pointing to External Files

Using the Actual Filename

In order to use the actual filename in the INFILE and FILE statements, the file to be accessed

- must be in your current default (or working) directory
- must have a valid SAS name
- must have filetype=DAT under VMS.

For example, if the filename of the raw data file is

```
MYFILE.DAT   (under VMS)
MYFILE       (under AOS/VS)
MYFILE       (under PRIMOS)
```

then the SAS code would be

```
INFILE MYFILE;      (to read)
FILE MYFILE;        (to write)
```

Pointing to External Files

Using a Fileref

A fileref must be defined for the file to be accessed if

- the file is not in your current default (or working) directory, and/or
- the name of the file is not a valid SAS name.

The fileref must be used in the INFILE and FILE statements to access the file.

Note: under VMS, if the filename is a valid SAS name, but the filetype is not DAT, then a fileref must be assigned to the file.

Pointing to External Files

Using a Fileref

A FILENAME statement is used to define the fileref of the file.

General form of the FILENAME statement:

FILENAME *fileref 'filename'*;

where

fileref specifies the fileref to be associated with the external file.

The fileref must be a valid SAS name.

A fileref can only refer to one file.

filename specifies the name of the file you want to access.

The filename must be a complete pathname that specifies the filename, not just the directory name.

Pointing to External Files

Using a Fileref

In order to access a file,

- use a FILENAME statement to define a fileref for the file

VMS

```
FILENAME fileref '[directoryname]filename.filetype';
```

AOS/VS

```
FILENAME fileref ':UDD:directoryname:filename';
```

PRIMOS

```
FILENAME fileref '<MFD>userdir>filename';
```

- specify the fileref in the INFILE statement to read the file

```
INFILE fileref;
```

- specify the fileref in the FILE statement to write to the file.

```
FILE fileref;
```

Reading External Files

To read data values that are stored on disk,

1. use a FILENAME statement to associate a fileref with the input data file

   ```
   FILENAME fileref 'filename';
   ```

2. use a DATA statement to begin the DATA step

   ```
   DATA SASdataset;
   ```

3. point to the FILENAME statement by specifying the fileref in an INFILE statement

   ```
   INFILE fileref;
   ```

4. indicate how to read the data lines with an INPUT statement.

   ```
   INPUT variables;
   ```

Reading External Files

Example: The data for the class example are stored in a file in your default (or working) directory with the name CLASS.RAW.

Create a SAS data set and print the data set.

A listing of the raw data file is shown below.

```
|...+....1....+....2....+....3 <---- Column

JOHN      M 12 59.0  99.5
JAMES     M 12 57.3  83.0
ALFRED    M 14 69.0 112.5
WILLIAM   M 15 66.5 112.0
JEFFREY   M 13 62.5  84.0
RONALD    M 15 67.0 133.0
THOMAS    M 11 57.5  85.0
PHILIP    M 16 72.0 150.0
ROBERT    M 12 64.8 128.0
HENRY     M 14 63.5 102.5
JANET     F 15 62.5 112.5
JOYCE     F 11 51.3  50.5
JUDY      F 14 64.3  90.0
CAROL     F 14 62.8 102.5
JANE      F 12 59.8  84.5
LOUISE    F 12 56.3  77.0
BARBARA   F 13 65.3  98.0
MARY      F 15 66.5 112.0
ALICE     F 13 56.5  84.0
```

Reading External Files

Store the following SAS statements in a file with the name C13EX1.SAS.

```
FILENAME STUDENT 'CLASS.RAW';
DATA CLASS;
   INFILE STUDENT;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
PROC PRINT;
```

Execute the SAS program by issuing the SAS command followed by the name of the file that contains the SAS statements.

```
SAS C13EX1
```

Reading External Files

Notes in the SAS log provide information about the input data file.

Issue a TYPE (VMS, AOS/VS) or SLIST (PRIMOS) command to print the SAS log on the screen.

```
TYPE C13EX1.LOG      (VMS and AOS/VS)
SLIST C13EX1.LOG     (PRIMOS)
```

Partial SAS Log

```
1 FILENAME STUDENT 'CLASS.RAW';
2 DATA CLASS;
3    INFILE STUDENT;
4    INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
5          HEIGHT 16-19 WEIGHT 21-25;
```

VMS

```
NOTE: INFILE STUDENT IS FILE SYS$SYSDEVICE:[SMITH]CLASS.RAW .
```

Reading External Files

AOS/VS

```
NOTE: INFILE STUDENT IS FILE :UDD:SMITH:CLASS.RAW .
```

PRIMOS

```
NOTE: INFILE STUDENT IS FILE <MFD>SMITH>CLASS.RAW .
```

```
NOTE: 19 LINES WERE READ FROM INFILE STUDENT.
      THE MINIMUM LINE LENGTH IS 25.
      THE MAXIMUM LINE LENGTH IS 25.
NOTE: THE DATA SET WORK.CLASS HAS 19 OBSERVATIONS AND 5
      VARIABLES.
```

Reading External Files

The SAS System stores the procedure output in a listing file.

Issue a TYPE (VMS, AOS/VS) or SLIST (PRIMOS) command to print the listing file on the screen.

```
TYPE  C13EX1.LIS      (VMS and AOS/VS)
SLIST C13EX1.LIST     (PRIMOS)
```

SAS

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0
3	ALFRED	M	14	69.0	112.5
4	WILLIAM	M	15	66.5	112.0
5	JEFFREY	M	13	62.5	84.0
6	RONALD	M	15	67.0	133.0
7	THOMAS	M	11	57.5	85.0
8	PHILIP	M	16	72.0	150.0
9	ROBERT	M	12	64.8	128.0
10	HENRY	M	14	63.5	102.5
11	JANET	F	15	62.5	112.5
12	JOYCE	F	11	51.3	50.5
13	JUDY	F	14	64.3	90.0
14	CAROL	F	14	62.8	102.5
15	JANE	F	12	59.8	84.5
16	LOUISE	F	12	56.3	77.0
17	BARBARA	F	13	65.3	98.0
18	MARY	F	15	66.5	112.0
19	ALICE	F	13	56.5	84.0

Reading from Your Default Directory

Example: Payroll data are stored in a disk file in your default directory. The name of the file is

PAYROLL.DAT under VMS

PAYROLL under AOS/VS

PAYROLL under PRIMOS.

Create a SAS data set named SALARIES from the raw data and print the data set.

Store the following SAS statements in a file with the name C13EX2.SAS .

```
DATA SALARIES;
   INFILE PAYROLL;
   INPUT BRANCH $ 1-11 DEPT 13-15 SEX $ 17
         GROSSPAY 19-25;
PROC PRINT;
```

Execute the SAS program by issuing the SAS command followed by the name of the file that contains the SAS statements.

```
SAS C13EX2
```

Note: a fileref does not have to be assigned to the file since it has a valid SAS name.

Reading from Your Default Directory

Issue a TYPE (VMS, AOS/VS) or SLIST (PRIMOS) command to print the SAS log on the screen.

```
TYPE C13EX2.LOG      (VMS and AOS/VS)
SLIST C13EX2.LOG     (PRIMOS)
```

Partial SAS Log

```
1 DATA SALARIES;
2    INFILE PAYROLL;
3    INPUT BRANCH $ 1-11 DEPT 12-15 SEX $ 17
4          GROSSPAY 19-25;
```

VMS

```
NOTE: INFILE PAYROLL IS FILE SYS$SYSDEVICE:[SMITH]PAYROLL.DAT .
```

AOS/VS

```
NOTE: INFILE PAYROLL IS FILE :UDD:SMITH:PAYROLL
```

Reading from Your Default Directory

PRIMOS

```
NOTE: INFILE PAYROLL IS FILE <MFD>SMITH>PAYROLL
```

```
NOTE: 21 LINES WERE READ FROM INFILE PAYROLL.
      THE MINIMUM LINE LENGTH IS 25.
      THE MAXIMUM LINE LENGTH IS 25.
NOTE: THE DATA SET WORK.SALARIES HAS 21 OBSERVATIONS AND 4
      VARIABLES.
```

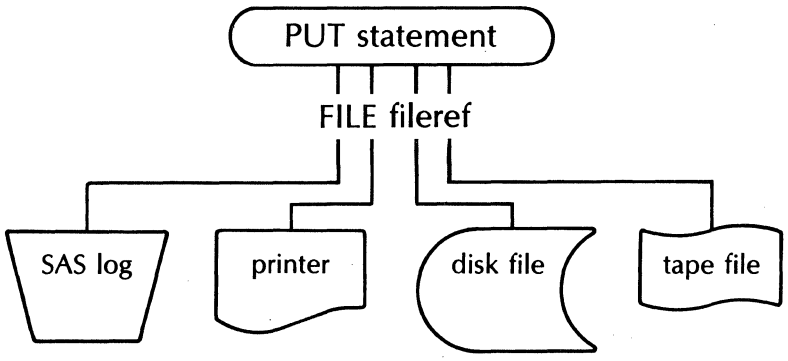
Reading from Your Default Directory

Issue a TYPE (VMS, AOS/VS) or SLIST (PRIMOS) command to print the listing file on the screen.

```
TYPE C13EX2.LIS      (VMS and AOS/VS)
SLIST C13EX2.LIST    (PRIMOS)
```

```
                          SAS

 OBS     BRANCH          DEPT     SEX     GROSSPAY

   1     RALEIGH         901       F        121.95
   2     RALEIGH         901       M        313.60
   3     LOS ANGELES     901       F        242.40
   4     RALEIGH         901       M        292.00
   5     RALEIGH         901       M        243.95
   6     LOS ANGELES     901       M        344.78
   7     LOS ANGELES     901       M        279.36
   8     RALEIGH         901       F        405.37
   9     RALEIGH         911       M        399.20
  10     RALEIGH         911       M        180.72
  11     RALEIGH         911       M        297.56
  12     LOS ANGELES     911       M        385.47
  13     LOS ANGELES     911       M        402.12
  14     LOS ANGELES     911       F        728.12
  15     RALEIGH         911       F        272.29
  16     LOS ANGELES     911       M        789.00
  17     LOS ANGELES     921       F        842.03
  18     LOS ANGELES     921       M        279.22
  19     LOS ANGELES     921       M        352.84
  20     RALEIGH         921       M       1235.46
  21     RALEIGH         921       F       1158.22
```

Writing External Files



The FILE and PUT statements are analogous to the INFILE and INPUT statements.

The FILE statement points to an output file by matching the fileref in a system command (INFILE points to an input file).

FILE *fileref*;

Special filerefs are

LOG for writing to the SAS log

PRINT for writing to the print file (report writing)

PUNCH for writing to the punch file (punching cards).

The PUT statement specifies the output record format (INPUT specifies the input record format).

PUT *variables formats pointer controls 'literals'*;

Writing External Files

To write raw data values to a file,

1. use a FILENAME statement to define a fileref for the output file

   ```
   FILENAME fileref 'filename';
   ```

2. use a DATA statement to begin the DATA step

   ```
   DATA _NULL_;
   ```

3. read the data

   ```
   SET SASdataset;
   ```
   (or INPUT if reading raw data)

4. point to the FILENAME statement by specifying the fileref in a FILE statement

   ```
   FILE fileref;
   ```

5. indicate how to write the data lines with a PUT statement.

   ```
   PUT variables formats pointer controls 'literals';
   ```

Writing External Files

Example: Read the data for the class example from an external file named CLASS.RAW, compute the ratio of height to weight, and write the values of the variables NAME, SEX, and RATIO (4 decimal places) to another external file named RATIOS.RAW.

Store the following statements in a file with the name C13EX3.SAS.

```
FILENAME OLD 'CLASS.RAW';
FILENAME PUPIL 'RATIOS.RAW';
DATA _NULL_;
   INFILE OLD;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   RATIO=HEIGHT/WEIGHT;
   FILE PUPIL;
   PUT NAME $8. @10 SEX $1. @12 RATIO 6.4;
```

Execute the SAS program by issuing the SAS command followed by the name of the file that contains the SAS statements.

```
SAS C13EX3
```

Note: the name _NULL_ in the DATA statement tells the SAS System to build a program data vector and process the observations without creating a SAS data set.

Writing External Files

Notes in the SAS log provide information about the output data file.

Issue a TYPE (VMS, AOS/VS) or SLIST (PRIMOS) command to print the SAS log on the screen.

```
TYPE C13EX3.LOG      (VMS and AOS/VS)
SLIST C13EX3.LOG     (PRIMOS)
```

Partial SAS Log

```
1 FILENAME OLD 'CLASS.RAW';
2 FILENAME PUPIL 'RATIOS.RAW';
3 DATA _NULL_;
4    INFILE OLD;
5    INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
6          HEIGHT 16-19 WEIGHT 21-25;
7    RATIO=HEIGHT/WEIGHT;
8    FILE PUPIL;
9    PUT NAME $8. @10 SEX $1. @12 RATIO 6.4;
```

Writing External Files

Partial SAS Log

VMS

```
NOTE: INFILE OLD IS FILE SYS$SYSDEVICE:[SMITH]CLASS.RAW .
NOTE: FILE PUPIL IS FILE SYS$SYSDEVICE:[SMITH]RATIOS.RAW .
```

AOS/VS

```
NOTE: INFILE OLD IS FILE :UDD:SMITH:CLASS.RAW .
NOTE: FILE PUPIL IS FILE :UDD:SMITH:RATIOS.RAW .
```

PRIMOS

```
NOTE: INFILE OLD IS FILE <UDD>SMITH>CLASS.RAW .
NOTE: FILE PUPIL IS FILE <MFD>SMITH>RATIOS.RAW .
```

```
NOTE: 19 LINES WERE READ FROM INFILE OLD.
      THE MINIMUM LINE LENGTH IS 25.
      THE MAXIMUM LINE LENGTH IS 25.
NOTE: 19 LINES WERE WRITTEN TO FILE PUPIL.
```

Writing to Your Default Directory

Example: Repeat the preceding example except store the data in a disk file in your default (or working) directory. The name of the output file should be

RATIOS.DAT under VMS

RATIOS under AOS/VS

RATIOS under PRIMOS.

Store the following SAS statements in a file with the name C13EX4.SAS .

```
FILENAME OLD 'CLASS.RAW';
DATA _NULL_;
   INFILE OLD;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   RATIO=HEIGHT/WEIGHT;
   FILE RATIOS;
   PUT NAME $8. @10 SEX $1. @12 RATIO 6.4;
```

Execute the SAS program by issuing the SAS command followed by the name of the file that contains the SAS statements.

```
SAS C13EX4
```

Note: a fileref does not have to be assigned to the file since it has a valid SAS name.

If the file does not already exist, it will be created.

Writing to Your Default Directory

Notes in the SAS log provide information about the output data file.

Issue a TYPE (VMS, AOS/VS) or SLIST (PRIMOS) command to print the SAS log on the screen.

```
TYPE C13EX4.LOG      (VMS and AOS/VS)
SLIST C13EX4.LOG     (PRIMOS)
```

Partial SAS Log

```
1 FILENAME OLD 'CLASS.RAW';
2 DATA _NULL_;
3     INFILE OLD;
4     INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
5           HEIGHT 16-19 WEIGHT 21-25;
6     RATIO=HEIGHT/WEIGHT;
7     FILE RATIOS;
8     PUT NAME $8. @10 SEX $1. @12 RATIO 6.4;
```

Writing to Your Default Directory

Partial SAS Log

VMS

```
NOTE: INFILE OLD IS FILE SYS$SYSDEVICE:[SMITH]CLASS.RAW .
NOTE: FILE RATIOS IS FILE SYS$SYSDEVICE:[SMITH]RATIOS.DAT .
```

AOS/VS

```
NOTE: INFILE OLD IS FILE :UDD:SMITH:CLASS.RAW .
NOTE: FILE RATIOS IS FILE :UDD:SMITH:RATIOS .
```

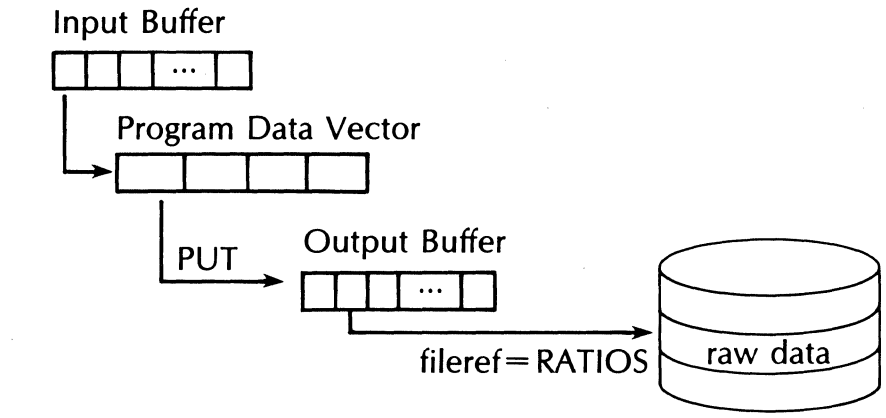
PRIMOS

```
NOTE: INFILE OLD IS FILE <UDD>SMITH>CLASS.RAW .
NOTE: FILE RATIOS IS FILE <MFD>SMITH>RATIOS .
```

```
NOTE: 19 LINES WERE READ FROM INFILE OLD.
      THE MINIMUM LINE LENGTH IS 25.
      THE MAXIMUM LINE LENGTH IS 25.
NOTE: 19 LINES WERE WRITTEN TO FILE RATIOS.
```

Writing to Your Default Directory

The data written to the disk file are shown below.




```
|...+....1....+....2
JOHN      M 0.5930
JAMES     M 0.6904
ALFRED    M 0.6133
WILLIAM   M 0.5938
JEFFREY   M 0.7440
RONALD    M 0.5038
THOMAS    M 0.6765
PHILIP    M 0.4800
ROBERT    M 0.5062
HENRY     M 0.6195
JANET     F 0.5556
JOYCE     F 1.0158
JUDY      F 0.7144
CAROL     F 0.6127
JANE      F 0.7077
LOUISE    F 0.7312
BARBARA   F 0.6663
MARY      F 0.5938
ALICE     F 0.6726
```

← Column

Writing to the SAS Log

Example: The raw data for students 12 to 16 years old in a summer camp are in a disk file in your default (or working) directory with the name

CAMP.DAT under VMS

CAMP under AOS/VS

CAMP under PRIMOS.

Write a SAS program to read the data and write a message to the SAS log when invalid age data are encountered.

Store the following SAS statements in a file with the name C13EX5.SAS .

```
DATA SUMMER;
   INFILE CAMP;
   INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
         HEIGHT 16-19 WEIGHT 21-25;
   IF AGE<12 OR AGE>16 THEN
      PUT 'INVALID AGE' +1 _N_= NAME= AGE=;
```

Execute the program by issuing the SAS command followed by the name of the file that contains the SAS statements.

```
SAS C13EX5
```

Writing to the SAS Log

The information specified on the PUT statement is written to the SAS log when an invalid age is encountered.

Issue a TYPE (VMS, AOS/VS) or SLIST (PRIMOS) command to print the SAS log on the screen.

```
TYPE C13EX5.LOG      (VMS and AOS/VS)
SLIST C13EX5.LOG     (PRIMOS)
```

Partial SAS Log

```
1 DATA SUMMER;
2    INFILE CAMP;
3    INPUT NAME $ 1-8 SEX $ 11 AGE 13-14
4          HEIGHT 16-19 WEIGHT 21-25;
5    IF AGE<12 OR AGE>16 THEN
6       PUT 'INVALID AGE' +1 _N_= NAME= AGE=;
```

Writing to the SAS Log

Partial SAS Log

VMS

```
NOTE: INFILE CAMP IS SYS$SYSDEVICE:[SMITH]CAMP.DAT .
```

AOS/VS

```
NOTE: INFILE CAMP IS :UDD:SMITH:CAMP .
```

PRIMOS

```
NOTE: INFILE CAMP IS <MFD>SMITH>CAMP .
```

```
INVALID AGE _N_=3 NAME=ALFRED AGE=17
INVALID AGE _N_=5 NAME=THOMAS AGE=17
INVALID AGE _N_=11 NAME=CAROL AGE=18
INVALID AGE _N_=15 NAME=ALICE AGE=17
NOTE: 15 LINES WERE READ FROM INFILE CAMP .
      THE MINIMUM LINE LENGTH IS 25.
      THE MAXIMUM LINE LENGTH IS 25.
NOTE: THE DATA SET WORK.SUMMER HAS 15 OBSERVATIONS AND 5
      VARIABLES.
```

Writing to the Print File

Example: The raw data for a company payroll are in a disk file in your default (or working) directory with the name

PAYROLL.DAT under VMS

PAYROLL under AOS/VS

PAYROLL under PRIMOS.

Write a SAS program that uses FILE and PUT statements to write a payroll report.

Store the following SAS statements in a file with the name C13EX6.SAS .

```
DATA _NULL_;
   INFILE PAYROLL;
   INPUT BRANCH $ 1-11 DEPT 13-15 SEX $ 17
         GROSSPAY 19-25;
   FILE PRINT;
   PUT @15 BRANCH $11. @30 DEPT 3. @37 SEX $1.
       @45 GROSSPAY DOLLAR9.2;
```

Execute the program by issuing the SAS command followed by the name of the file that contains the SAS statements.

```
SAS C13EX6
```

Writing to the Print File

FILE PRINT sends the generated report to the listing file.

Issue a TYPE (VMS, AOS/VS) or SLIST (PRIMOS) command to print the listing file on the screen.

```
TYPE C13EX6.LIS        (VMS and AOS/VS)
SLIST C13EX6.LIST      (PRIMOS)
```

```
                              SAS

RALEIGH          901    F          $121.95
RALEIGH          901    M          $313.60
LOS ANGELES      901    F          $242.40
RALEIGH          901    M          $292.00
RALEIGH          901    M          $243.95
LOS ANGELES      901    M          $344.78
LOS ANGELES      901    M          $279.36
RALEIGH          901    F          $405.37
RALEIGH          911    M          $399.20
RALEIGH          911    M          $180.72
RALEIGH          911    M          $297.56
LOS ANGELES      911    M          $385.47
LOS ANGELES      911    M          $402.12
LOS ANGELES      911    F          $728.12
RALEIGH          911    F          $272.29
LOS ANGELES      911    M          $789.00
LOS ANGELES      921    F          $842.03
LOS ANGELES      921    M          $279.22
LOS ANGELES      921    M          $352.84
RALEIGH          921    M        $1,235.46
RALEIGH          921    F        $1,158.22
```

13.5 Exercises

13.1 The National Safety Council annually compiles information on accidental deaths in the home. Each record of information represents one year of data. The record layout is given in the table below.

Field Description	Field Location	Variable Type
Year	1-4	numeric
Falls	5-9	numeric
Fires, burns	10-14	numeric
Suffocation, ingested	15-19	numeric
Suffocation, mechanical	20-24	numeric
Poison, solid or liquid	25-29	numeric
Poison by gas	30-34	numeric
Firearms	35-39	numeric
Other	40-44	numeric

Assume that the data file has been assigned fileref=ACCIDENT.

a. Create a SAS data set from the raw data file with the variables on the next page.

Exercises

13.1

Variable Name	Description (cause of death)
YEAR	year
FALLS	falls
BURNS	fires or burns
INGEST	suffocation, ingested
MECH	suffocation, mechanical
SOLID	poison, solid or liquid
GAS	poison by gas
GUNS	firearms
OTHER	other
TOTAL	total deaths by all causes

b. Print the data set.

c. Compute the average number of deaths over the years for each cause and for all causes.

Exercises

13.2 The National Center for Health Statistics keeps records on the numbers of marriages and divorces for each state for each year. The data are stored on a disk file with the following record layout. Assume the disk file has been assigned fileref=STATS.

Field Description	Field Position	Variable Type
Year	1-4	numeric
State	5-19	character
Number of marriages	20-25	numeric
Number of divorces	26-31	numeric

a. Create a SAS data set that contains the data stored in the file. Choose appropriate variable names.

b. Compute the total number of marriages and total number of divorces in the U.S. for each year.

c. Compute the average number of marriages and average number of divorces over the years for each state.

Exercises

13.3 Use the SAS data set created in Exercise 13.1 to create a raw data file with the following record format. Assume that the space needed to store the file has already been set up and assigned fileref=NEWDATA.

Variable Name	Field Location	Variable Type
YEAR	1-4	numeric
TOTAL	5-9	numeric
FALLS	10-14	numeric
BURNS	15-19	numeric
INGEST	20-24	numeric
MECH	25-29	numeric
SOLID	30-34	numeric
GAS	35-39	numeric
GUNS	40-44	numeric
OTHER	45-49	numeric

Exercises

13.4 A survey of attitudes toward nuclear power plants produced data with a record format described in the table below.

Variable Name	Record Number	Field Location	Variable Type
REGION	1	1-20	character
PLANTS	1	21-25	numeric
PROXIMIT	1	26-30	numeric
LOCATION	1	31-40	character
ACCIDENT	2	26-30	numeric
ENERGY	2	31-35	numeric
FAVOR	2	36	character
QUEST1	3	1	character
QUEST2	3	2	character
QUEST3	3	3	character
QUEST4	3	4	character

The data are stored on cards.

Prepare SAS statements that will read the data and store the data on disk with the record format shown on the next page.

Exercises

13.4

Variable Name	Field Location	Variable Type
REGION	1-20	character
PLANTS	21-25	numeric
PROXIMIT	26-30	numeric
LOCATION	31-40	character
ACCIDENT	41-45	numeric
ENERGY	46-50	numeric
FAVOR	51	character
QUEST1	52	character
QUEST2	53	character
QUEST3	54	character
QUEST4	55	character

Assume that the space needed to store the data has already been provided and assigned fileref=DISKDATA.

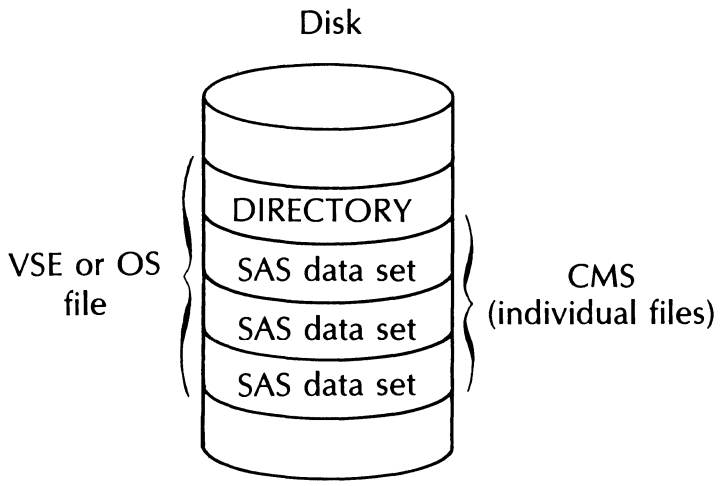
Note that the data on three consecutive cards are stored on one record in the disk file.

14. Permanent SAS Data Sets

14.1 Introduction: Mainframe Environments

SAS Data Libraries

- In OS and VSE environments, a SAS data library is a collection of SAS data sets stored in a single system file.
- In the CMS environment, a SAS data library is a collection of SAS data sets (individual CMS files with identical filetypes) located on the same minidisk.
- In OS and VSE environments, one of the SAS data sets in the library is a directory.
- The directory is automatically updated whenever a SAS data set is added to or deleted from the library.



Two-Level SAS Data Set Names

Every SAS data set has a two-level name:

libref.SASdataset

libref (short for SAS data library reference) is a name that is associated with a storage location.

The libref is equivalent to a DDname for mainframe users.

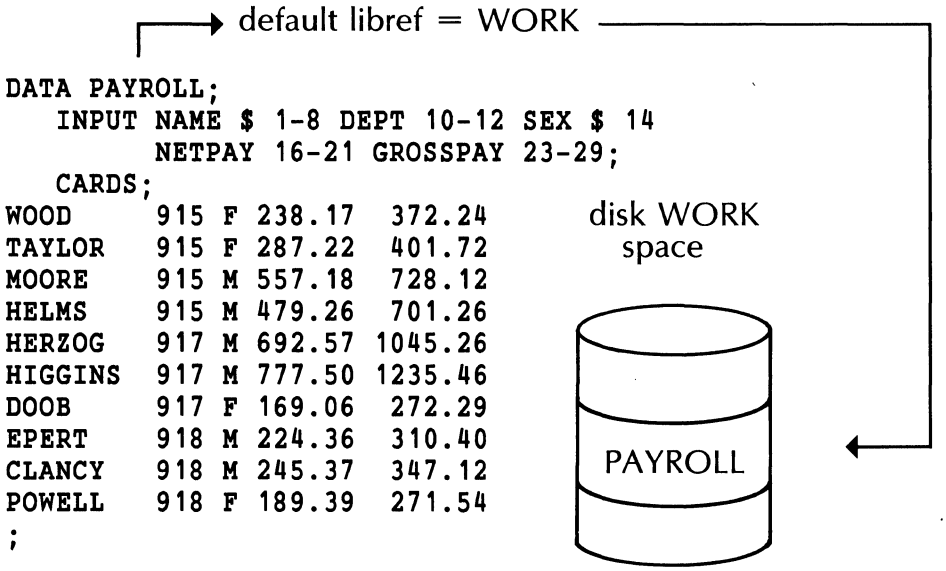
The libref is associated with the appropriate file through a system command.

SASdataset refers to the SAS data set that is being created (or read) (for example, CLASS or PAYROLL).

If a libref is not specified, the default libref WORK is used.

WORK data sets are stored in temporary work space and exist only for the duration of the job or session.

Temporary SAS Data Sets



The note in the SAS log for this DATA step would read:

```
NOTE: DATA SET WORK.PAYROLL HAS 10 OBSERVATIONS AND 5 VARIABLES.
 515 OBS/TRK.
```

The data set WORK.PAYROLL has been stored in WORK space and will be deleted at the end of the job or session.

The data set may be referred to as PAYROLL or WORK.PAYROLL.

```
PROC PRINT DATA=WORK.PAYROLL;
```

is equivalent to

```
PROC PRINT DATA=PAYROLL;
```

System Commands

- When storing a SAS data set, a system command is used to point to the storage location for the SAS data set.
- The appropriate command depends upon the operating system.

General form of the system commands:

OS batch

```
//libref DD DSN=OSdatasetname,DISP=xxx,UNIT=xxx,
// LABEL=xxx,VOL=xxx,SPACE=xxx
```

TSO

```
ALLOCATE FILE(libref) DATASET('OSdatasetname')
         disp unit_of_space SPACE(xxx)

ALLOC F(libref) DA('OSdatasetname')
      disp unit_of_space SPACE(xxx)
```

CMS

```
FILEDEF libref DISK filename filetype filemode
```

Under CMS, the FILEDEF command is optional since the following FILEDEF command is automatically generated for your A disk whenever you use a two-level SAS data set name.

```
FILEDEF libref DISK SASdataset libref A
```

System Commands

VSE batch

```
// DLBL libref,'file-ID',date
// EXTENT SYSnnn,volser,type,seq,begin,tracks|blocks
// ASSGN logicalunit,device,VOL=volser,SHR
```

ICCF

```
/FILE NAME=libref,ID='file-ID',UNITS=logicalunit,
/ SERIAL=volser,DATE=date,LOC=begin,tracks|blocks
```

Why Store SAS Data Sets?

There are several advantages to storing data in SAS data sets:

- SAS data sets are self-documenting.
- The data can be directly accessed by SAS procedures.
- The data can be directly accessed by SET, MERGE, and UPDATE statements (in other words, an INPUT statement is not needed).
- Data conversion is not necessary (computer time is saved since the data are stored in the form that the SAS System requires).

14.2 OS Batch, TSO, and CMS Environments

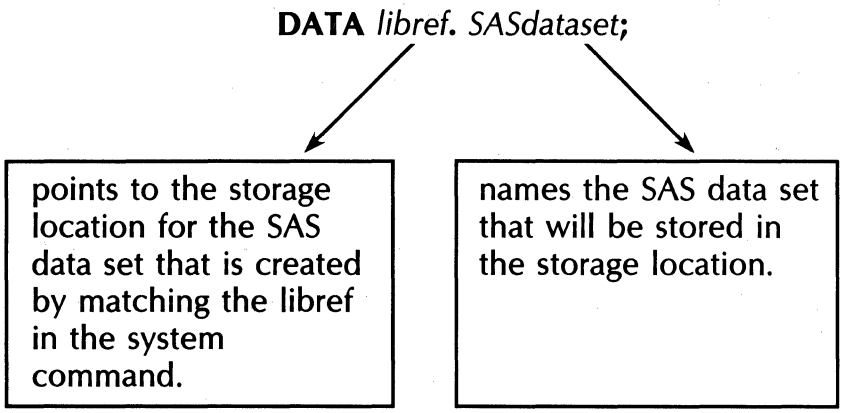
Data Libraries: OS Batch and TSO

Under OS batch and TSO, SAS data libraries on disk

- have a directory that records what data sets are in the library and where they are located
- are data files that may exist in multiple extents on the disk
- contain SAS data sets that may occupy several noncontiguous extents
- are not "compressed" when a SAS data set is deleted
- are able to reuse the space left by a deleted SAS data set when a SAS data set is added.

Permanently Storing SAS Data Sets

- A SAS data set can be permanently stored by specifying a libref other than WORK when the data set is created.



- A system command with the same libref must also be specified to point to the appropriate storage location.

Creating a Library: OS Batch and TSO

To create a SAS data library in the OS batch and TSO environments,

- specify (as required by your installation):

  data set name
  disposition
  unit
  volume serial number

- specify the amount of disk space (request a primary and, optionally, a secondary space allocation)
- **do not** specify directory blocks
- **do not** specify DCB information.

Creating a Library: CMS

If a SAS data set is to be stored on your A disk, simply use a two-level SAS data set name in your SAS program.

If the SAS data set is to be stored on another minidisk, then you must specify the appropriate FILEDEF command.

Specify

- a libref
- DISK
- a dummy filename and filetype
- the appropriate filemode.

Permanently Storing SAS Data Sets

Example: An airline company stores its accounting information in raw data files for each month.

Create a SAS data library and store January and February data in separate SAS data sets within the library.

OS batch

Use DD statements to create the SAS data library and to point to the raw data files.

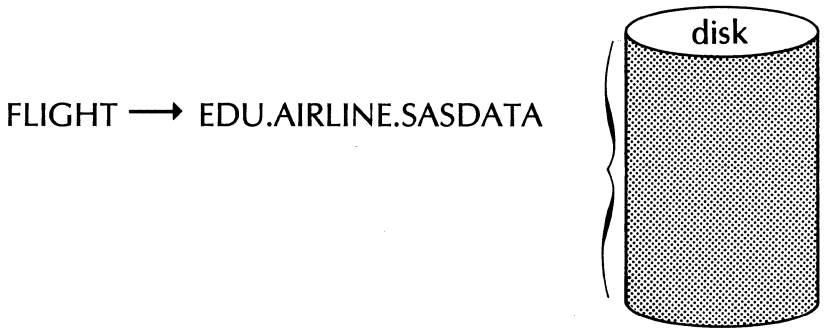
```
// EXEC SAS
//FLIGHT DD DSN=EDU.AIRLINE.SASDATA,
// DISP=(NEW,CATLG),SPACE=(TRK,(15)),
// UNIT=DISK,VOL=SER=DISK01
//DATA1 DD DSN=EDU.JANFLY.DATA,DISP=SHR
//DATA2 DD DSN=EDU.FEBFLY.DATA,DISP=SHR
```

Permanently Storing SAS Data Sets

TSO

Use allocate statements to create the SAS data library and to point to the raw data files.

```
READY
alloc f(flight) da('edu.airline.sasdata')
         new tracks space(15)
READY
alloc f(data1) da('edu.janfly.data') shr
READY
alloc f(data2) da('edu.febfly.data') shr
```



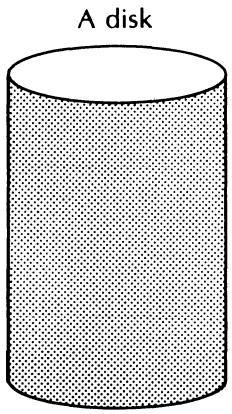
Permanently Storing SAS Data Sets

CMS

Use FILEDEF commands to point to the raw data files.

```
R;
filedef data1 disk jan flydata a
R;
filedef data2 disk feb flydata a
R;
```

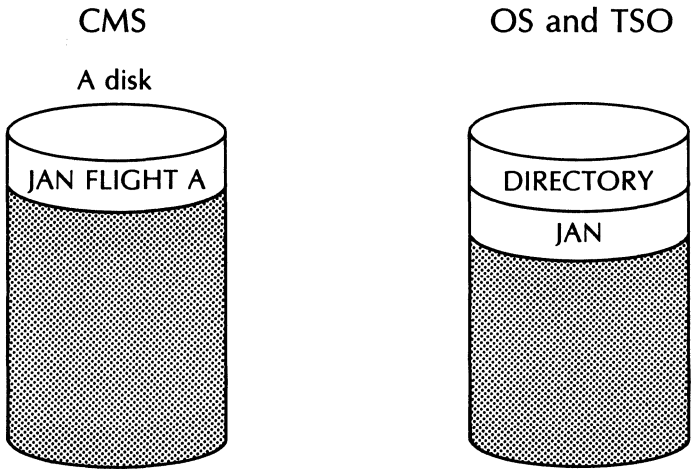
Note: under CMS, a FILEDEF command is not necessary when storing SAS data sets on your A disk.



Permanently Storing SAS Data Sets

Example: Read the January raw data from an external file and store the SAS data set in the SAS data library.

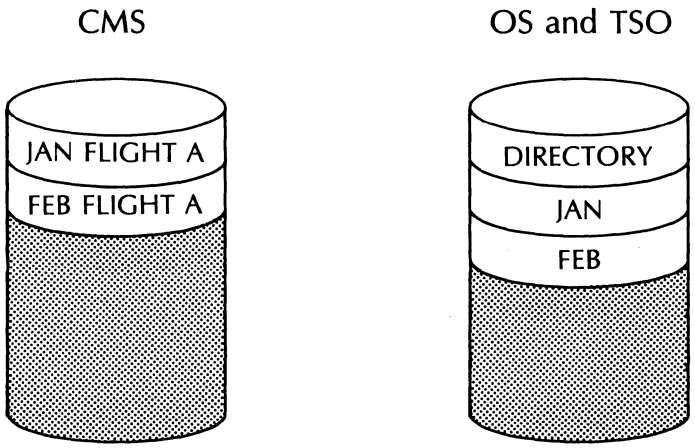
```
DATA FLIGHT.JAN;
   INFILE DATA1;
   INPUT FLIGHTNO 1-3 DATE MMDDYY8. ORIGIN $ 12-14
         DEST $ 15-17 COUNT 18-19 REVENUE 20-24;
```



Permanently Storing SAS Data Sets

Example: Read the February raw data file and store the SAS data set in the SAS data library.

```
DATA FLIGHT.FEB;
   INFILE DATA2;
   INPUT FLIGHTNO 1-3 DATE MMDDYY8. ORIGIN $ 12-14
         DEST $ 15-17 COUNT 18-19 REVENUE 20-24;
```



Permanently Storing SAS Data Sets

Example: Use another SAS program to place March flight data in the SAS data library and to concatenate the first-quarter data.

Write system commands to access the SAS data library and the raw data for March.

OS batch

```
// EXEC SAS
//MARCH DD DSN=EDU.MARFLY.DATA,DISP=SHR
//FLIGHT DD DSN=EDU.AIRLINE.SASDATA,DISP=OLD
```

TSO

```
READY
alloc f(march) da('edu.marfly.data') shr
READY
alloc f(flight) da('edu.airline.sasdata') old
READY
```

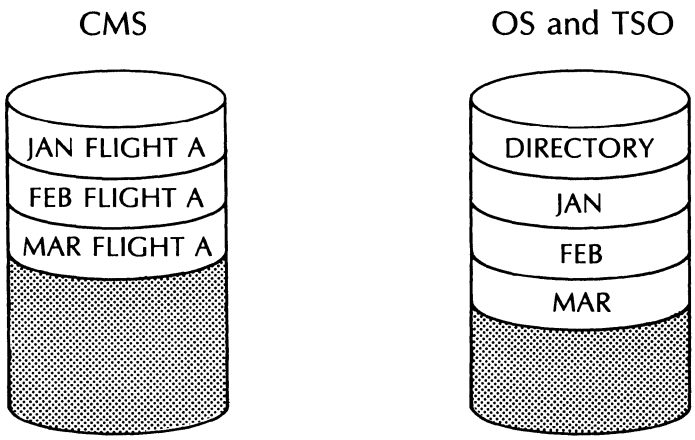
CMS

```
R;
filedef march disk mar flydata a
R;
```

Permanently Storing SAS Data Sets

Example: Store the March flight data in a SAS data set in the library.

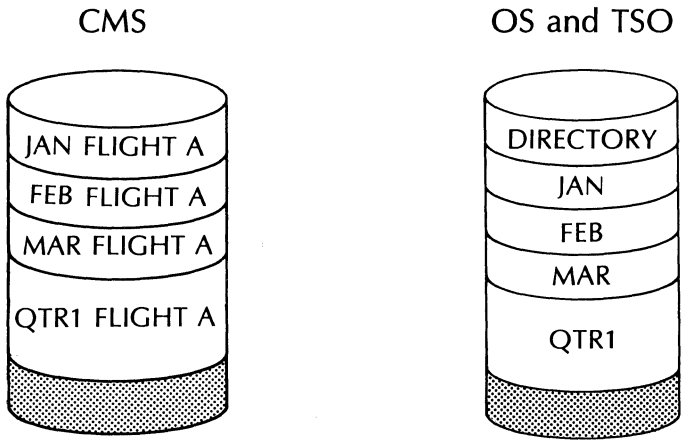
```
DATA FLIGHT.MAR;
   INFILE MARCH;
   INPUT FLIGHTNO 1-3 DATE MMDDYY8. ORIGIN $ 12-14
         DEST $ 15-17 COUNT 18-19 REVENUE 20-24;
```



Permanently Storing SAS Data Sets

Example: Concatenate the three SAS data sets into one SAS data set.

```
DATA FLIGHT.QTR1;
   SET FLIGHT.JAN FLIGHT.FEB FLIGHT.MAR;
```



SAS Utility Procedures

The SAS System provides several utility procedures for managing SAS data libraries.

CONTENTS prints descriptions of the contents of one or more data sets in a SAS data library.

COPY copies an entire SAS data library or selected members of the library.

DATASETS deletes and renames SAS data sets stored in a SAS data library on disk.

General form of the PROC CONTENTS statement:

PROC CONTENTS *options*;

Selected options:

DATA=*libref.member*
refers to an entire library or a single data set in the library.

member can be

- *SASdataset* to refer to a specific SAS data set in the library
- _ALL_ to refer to all SAS data sets in the library.

NODS
suppresses printing the contents of individual SAS data sets when _ALL_ is specified.

SAS Utility Procedures

General form of the PROC COPY statement:

PROC COPY IN=*libref* **OUT=***libref options*;

Selected statements used with PROC COPY:

SELECT *SASdataset . . . / options*;

EXCLUDE *SASdataset . . . / options*;

Note: if neither a SELECT nor EXCLUDE statement is used, the entire library is copied.

General form of the PROC DATASETS statement:

PROC DATASETS LIBRARY=*libref options*;

Selected statements used with PROC DATASETS:

DELETE *SASdataset . . . / options*;

SAVE *SASdataset . . . / options*;

CHANGE *oldname=newname . . . / options*;

Note: PROC DATASETS can be executed in line mode or full-screen mode.

The NOFS option in the PROC DATASETS statement puts the procedure in line mode.

Displaying the Contents of a Library

Example: Write a system command to access the SAS data library.

Display the names of all the SAS data sets in the library, but do not show any individual data set information.

OS batch

```
// EXEC SAS
//FLIGHT DD DSN=EDU.AIRLINE.SASDATA,DISP=SHR
```

TSO

```
READY
alloc f(flight) da('edu.airline.sasdata') shr
READY


PROC CONTENTS DATA=FLIGHT._ALL_ NODS;
```

Note: under CMS, a FILEDEF command is not necessary since the SAS data library is on the A disk.

Displaying the Contents of a Library

```
                          CONTENTS PROCEDURE
                PHYSICAL CHARACTERISTICS OF THE OS DATA SET


DSNAME= EDU.AIRLINE.SASDATA   UNIT=DISK   VOL=SER=EDU999.   DISP=SHR
DEVICE=3380 DISK   MAX BLKSIZE=32760 BYTES
CREATED ON FEBRUARY 21, 1986   15 TRACKS ALLOCATED IN 1 EXTENT(S)


                       SAS DATA LIBRARY DIRECTORY

        NAME        MEMTYPE     #OBS        TRACKS      EXTENTS


        FEB         DATA        19             1           1
        JAN         DATA        35             1           1
        MAR         DATA        47             1           1
        QTR1        DATA        101            1           1

                      TOTAL TRACKS USED =      5
                      HIGH  TRACKS USED =      5
```

Displaying the Contents of a Library

Example: Display the contents of the QTR1 data set.

```
PROC CONTENTS DATA=FLIGHT.QTR1;
```

```
                        CONTENTS PROCEDURE
                  CONTENTS OF SAS MEMBER FLIGHT.QTR1


CREATED BY TSO USERID EDU       ON CPUID 00-0000-000000
AT 10:06 FRIDAY, FEBRUARY 21, 1986   BY SAS RELEASE 5.XX
DSNAME=EDU.AIRLINE.SASDATA   OBSERVATIONS PER TRACK =1116
BLKSIZE=23440   LRECL=42   GENERATED BY DATA
NUMBER OF OBSERVATIONS: 101   NUMBER OF VARIABLES: 6
MEMTYPE:   DATA

          ----ALPHABETIC LIST OF VARIABLES AND ATTRIBUTES-----
       # VARIABLE TYPE   LENGTH POSITION FORMAT       INFORMAT     LABEL
       5 COUNT    NUM         8       26
       2 DATE     NUM         8       12
       4 DEST     CHAR        3       23
       1 FLIGHTNO NUM         8        4
       3 ORIGIN   CHAR        3       20
       6 REVENUE  NUM         8       34


----------------------------- SOURCE RECORDS -------------------------
| DATA FLIGHT.QTR1;
|    SET FLIGHT.JAN FLIGHT.FEB FLIGHT.MAR;
------------------------------------------------------------------------
```

Deleting and Renaming SAS Data Sets

Example: Write system commands to access the SAS data library and to access the raw data for April.

OS batch

```
// EXEC SAS
//APRIL DD DSN=EDU.APRFLY.DATA,DISP=SHR
//FLIGHT DD DSN=EDU.AIRLINE.SASDATA,DISP=OLD
```

TSO

```
READY
alloc f(april) da('edu.aprfly.data') shr
READY
alloc f(flight) da('edu.airline.sasdata') old
READY
```

CMS

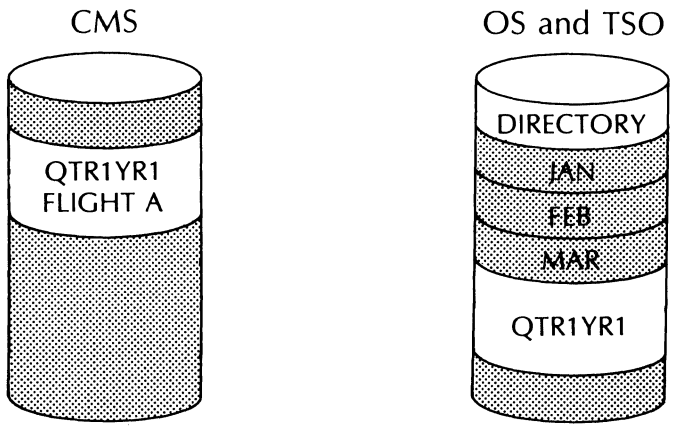
```
R;
filedef april disk apr flydata a
R;
```

Note: under CMS, a FILEDEF command is not necessary since the SAS data library is on the A disk.

Deleting and Renaming SAS Data Sets

Example: Delete the three monthly data sets and rename the first-quarter data set.

```
PROC DATASETS LIBRARY=FLIGHT;
   DELETE JAN FEB MAR;
   CHANGE QTR1=QTR1YR1;
```



Deleting and Renaming SAS Data Sets

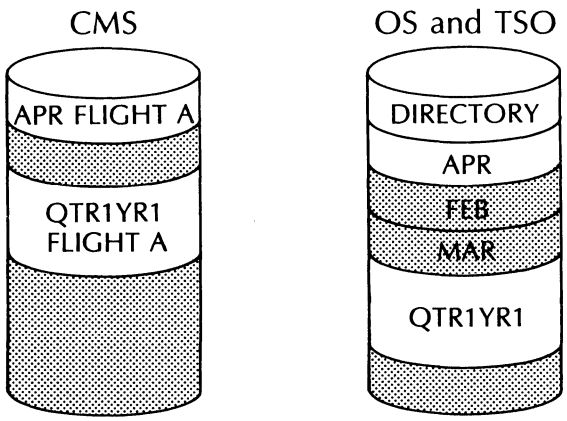
Partial SAS Log

```
 OS SAS DATA LIBRARY: DSNAME=EDU.AIRLINE.SASDATA
                      VOLUME=EDU999.
LIST OF MEMBERS BEFORE UPDATE OF DIRECTORY.
NAME     MEMTYPE             OBS  TRACKS PROT
FEB      /DATA                19       1
JAN      /DATA                35       1
MAR      /DATA                47       1
QTR1     /DATA               101       1
LIST OF MEMBERS AFTER UPDATE OF DIRECTORY.
NAME     MEMTYPE             OBS  TRACKS PROT
QTR1YR1 /DATA                101       1
     2 TRACKS USED
    15 TRACKS ALLOCATED
     5 HIGH TRACK USED
    10 TRACKS THAT CAN BE RELEASED FROM OS DATA SET
     1 EXTENTS
```

Adding Data Sets to the Library

Example: Store the APRIL data in a SAS data set in the SAS data library.

```
DATA FLIGHT.APR;
   INFILE APRIL;
   INPUT FLIGHTNO 1-3 DATE MMDDYY8. ORIGIN $ 12-14
         DEST $ 15-17 COUNT 18-19 REVENUE 20-24;
```



Copying a Library: OS Batch and TSO

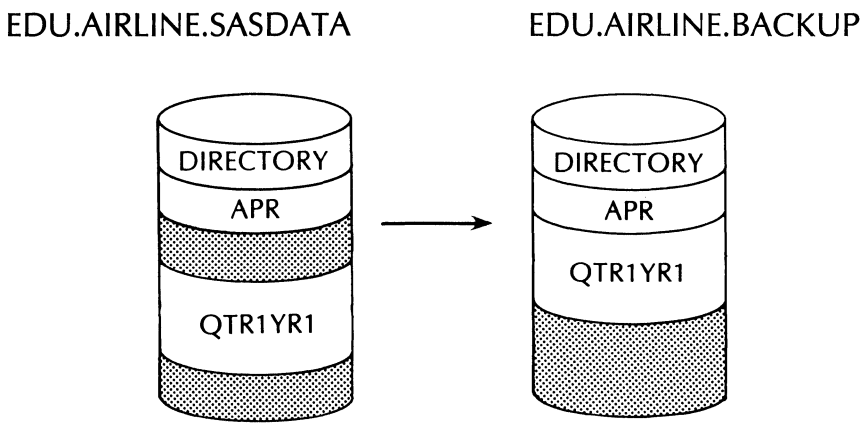
Example: Use another SAS program to create a backup of the library.

OS batch

```
// EXEC SAS
//OLD DD DSN=EDU.AIRLINE.SASDATA,DISP=SHR
//NEW DD DSN=EDU.AIRLINE.BACKUP,
// DISP=(NEW,CATLG),SPACE=(TRK,(15)),
// UNIT=DISK,VOL=SER=DISK01
```

TSO

```
READY
alloc f(old) da('edu.airline.sasdata') shr
READY
alloc f(new) da('edu.airline.backup')
         new tracks space(15)
READY


PROC COPY IN=OLD OUT=NEW;
```

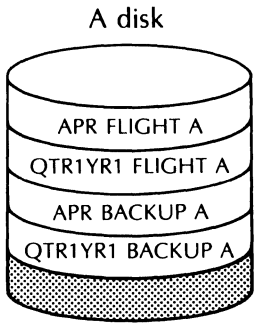


Copying a Library: CMS

CMS

Example: Place a backup of the library on your A disk.

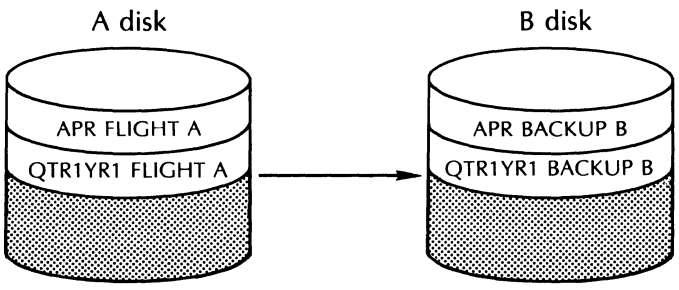
```
PROC COPY IN=FLIGHT OUT=BACKUP;
```



Note: FLIGHT is the filetype of all the data sets in the existing library, and BACKUP will be the filetype of all the data sets in the new library.

Example: Place a backup of the library on your B disk.

```
R;
filedef backup disk dummy1 dummy2 b
R;


PROC COPY IN=FLIGHT OUT=BACKUP;
```



Note: all SAS data sets with a filetype of FLIGHT on the A disk will be copied to the B disk with a filetype of BACKUP.

14.3 VSE and ICCF Environments

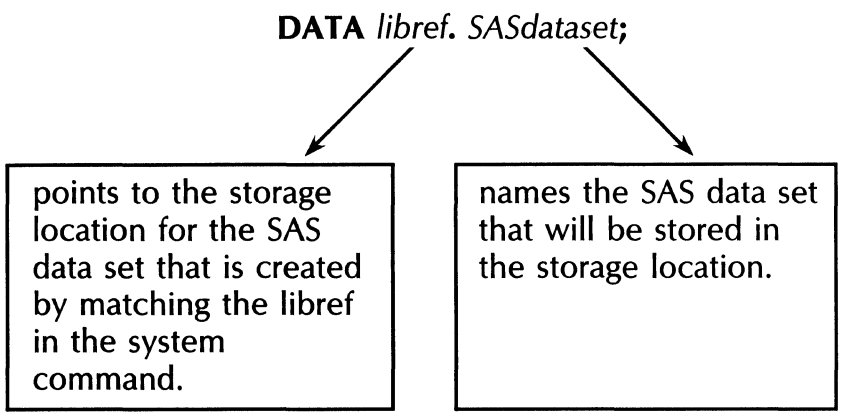
SAS Data Libraries

Under VSE batch and ICCF, SAS data libraries on disk

- have a directory that records what data sets are in the library and where they are located
- are data files that exist in a single extent on the disk
- contain SAS data sets that may occupy several noncontiguous extents
- are not "compressed" when a SAS data set is deleted
- are able to reuse the space left by a deleted SAS data set when a SAS data set is added.

Permanently Storing SAS Data Sets

- A SAS data set can be permanently stored by specifying a libref other than WORK when the data set is created.



- A system command with the same libref must also be specified to point to the appropriate storage location.

Libref Naming Conventions

The SAS System refers to SAS data libraries by their librefs.

Librefs used for SAS data libraries in the VSE environment must follow the rule stated below.

Rule: The first letter of the libref specifies the way the SAS data library is to be accessed. The types of access and the letters used to specify them are shown below.

Access Mode	Libref First Letter
Work	W
Output	O
Input	I or S
Update	Any except W, O, I, or S

Note: see the *SAS Companion for the VSE Operating System* for additional documentation.

Creating a Data Library

To create a SAS data library in the VSE environment,

- specify (as required by your installation):

  data set name
  device
  logical unit
  volume serial

- specify the amount of disk space.

Permanently Storing SAS Data Sets

Example: An airline company stores its accounting information in raw data files for each month.

Create a SAS data library and store January and February data in separate SAS data sets within the library.

VSE batch

Use DLBL, ASSGN, and EXTENT statements to create the SAS data library and point to the raw data files.

```
// DLBL DATA1,'EDU.JANFLY.DATA'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
// DLBL DATA2,'EDU.FEBFLY.DATA'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
// DLBL OFLIGHT,'EDU.AIRLINE.SASDATA',99/365
// EXTENT SYS055,DOSRES,1,0,9950,15
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
```

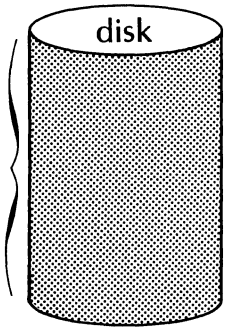
Permanently Storing SAS Data Sets

ICCF

Use FILE statements to create the SAS data library and point to the raw data files.

```
/FILE NAME=DATA1,ID='EDU.JANFLY.DATA',UNITS=SYS055,
/ SERIAL=DOSRES
/FILE NAME=DATA2,ID='EDU.FEBFLY.DATA',UNITS=SYS055,
/ SERIAL=DOSRES
/FILE NAME=OFLIGHT,ID='EDU.AIRLINE.SASDATA',UNITS=SYS055,
/ SERIAL=DOSRES,DATE=99/365,LOC=9950,15
```

OFLIGHT⟶EDU.AIRLINE.SASDATA

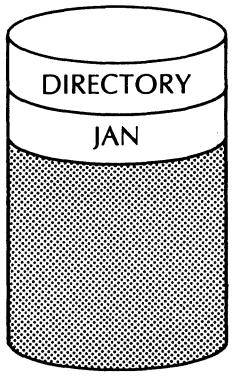


Note: under ICCF, the logical unit SYS055 must be assigned to DOSRES in the ICCF start-up job.

Permanently Storing SAS Data Sets

Example: Read the January raw data from an external file and store the SAS data set in the SAS data library.

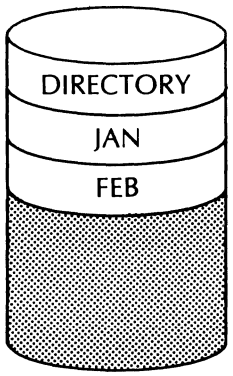
```
DATA OFLIGHT.JAN;
   INFILE DATA1 RECFM=FB LRECL=80 BLKSIZE=9040;
   INPUT FLIGHTNO 1-3 DATE MMDDYY8. ORIGIN $ 12-14
         DEST $ 15-17 COUNT 18-19 REVENUE 20-24;
```



Permanently Storing SAS Data Sets

Example: Read the February raw data file and store the SAS data set in the SAS data library.

```
DATA OFLIGHT.FEB;
   INFILE DATA2 RECFM=FB LRECL=80 BLKSIZE=9040;
   INPUT FLIGHTNO 1-3 DATE MMDDYY8. ORIGIN $ 12-14
         DEST $ 15-17 COUNT 18-19 REVENUE 20-24;
```



Permanently Storing SAS Data Sets

Example: Use another SAS program to place March flight data in the SAS data library and to concatenate the first-quarter data.

Write system commands to access the SAS data library and the raw data file for March.

VSE batch

```
// DLBL MARCH,'EDU.MARFLY.DATA'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
// DLBL FLIGHT,'EDU.AIRLINE.SASDATA'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
```

ICCF

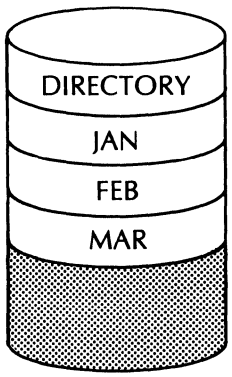
```
/FILE NAME=MARCH,ID='EDU.MARFLY.DATA',UNITS=SYS055,
/ SERIAL=DOSRES
/FILE NAME=FLIGHT,ID='EDU.AIRLINE.SASDATA',UNITS=SYS055,
/ SERIAL=DOSRES
```

Note: under ICCF, the logical unit SYS055 must be assigned to DOSRES in the ICCF start-up job.

Permanently Storing SAS Data Sets

Example: Store the March flight data in a SAS data set in the library.

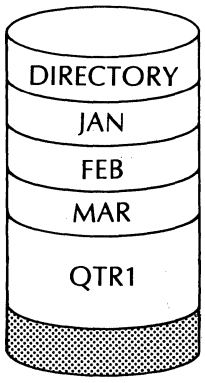
```
DATA FLIGHT.MAR;
   INFILE MARCH RECFM=FB LRECL=80 BLKSIZE=9040;
   INPUT FLIGHTNO 1-3 DATE MMDDYY8. ORIGIN $ 12-14
         DEST $ 15-17 COUNT 18-19 REVENUE 20-24;
```



Permanently Storing SAS Data Sets

Example: Concatenate the three SAS data sets into one SAS data set.

```
DATA FLIGHT.QTR1;
   SET FLIGHT.JAN FLIGHT.FEB FLIGHT.MAR;
```



SAS Utility Procedures

The SAS System provides several utility procedures for managing SAS data libraries.

CONTENTS prints descriptions of the contents of one or more data sets in a SAS data library.

COPY copies an entire SAS data library or selected members of the library.

DATASETS deletes and renames SAS data sets stored in a SAS data library on disk.

General form of the PROC CONTENTS statement:

PROC CONTENTS *options*;

Selected options:

DATA=*libref.member*
refers to an entire library or a single data set in the library.

member can be

- *SASdataset* to refer to a specific SAS data set in the library
- _ALL_ to refer to all SAS data sets in the library.

NODS
suppresses printing the contents of individual SAS data sets when _ALL_ is specified.

SAS Utility Procedures

General form of the PROC COPY statement:

PROC COPY IN=*libref* **OUT=***libref options*;

Selected statements used with PROC COPY:

SELECT *SASdataset . . . / options*;

EXCLUDE *SASdataset . . . / options*;

Note: if neither a SELECT nor EXCLUDE statement is used, the entire library is copied.

General form of the PROC DATASETS statement:

PROC DATASETS LIBRARY=*libref options*;

Selected statements used with PROC DATASETS:

DELETE *SASdataset . . . / options*;

SAVE *SASdataset . . . / options*;

CHANGE *oldname=newname . . . / options*;

Note: PROC DATASETS can be executed in line mode or full-screen mode.

The NOFS option in the PROC DATASETS statement puts the procedure in line mode.

Displaying the Contents of a Library

Example: Write a system command to access the SAS data library.

Display the names of all the SAS data sets in the library, but do not show any individual data set information.

VSE batch

```
// DLBL FLIGHT,'EDU.AIRLINE.SASDATA'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
```

ICCF

```
/FILE NAME=FLIGHT,ID='EDU.AIRLINE.SASDATA',UNITS=SYS055,
/ SERIAL=DOSRES
```

Note: under ICCF, the logical unit SYS055 must be assigned to DOSRES in the ICCF start-up job.

```
PROC CONTENTS DATA=FLIGHT._ALL_ NODS;
```

Displaying the Contents of a Library

```
                                SAS

                          CONTENTS PROCEDURE
                PHYSICAL CHARACTERISTICS OF THE VSE DATA SET


FILENAME= FLIGHT   VOL=DOSRES   DEVICE=3380 DISK   STARTING TRACK=9950
NUMBER OF TRACKS=15   MAX BLKSIZE=47476 BYTES
15 TRACKS ALLOCATED IN 1 EXTENT


                       SAS DATA LIBRARY DIRECTORY

        NAME         MEMTYPE     #OBS         TRACKS      EXTENTS


        FEB          DATA        19              1           1
        JAN          DATA        35              1           1
        MAR          DATA        47              1           1
        QTR1         DATA        101             1           1

                      TOTAL TRACKS USED =      5
                      HIGH  TRACKS USED =      5
                      PERCENT FILE USED =     33
```

Displaying the Contents of a Library

Example: Display the contents of the QTR1 data set.

```
PROC CONTENTS DATA=FLIGHT.QTR1;
```

```
                                SAS

                          CONTENTS PROCEDURE
                    CONTENTS OF SAS MEMBER FLIGHT.QTR1


CREATED BY VSE JOB BATCH2   ON CPUID 00-0000-000000
AT 9:45 TUESDAY, MARCH 4, 1986   BY SAS RELEASE 5.XX
OBSERVATIONS PER TRACK =1116   BLKSIZE=23440   LRECL=42
GENERATED BY DATA
NUMBER OF OBSERVATIONS: 101   NUMBER OF VARIABLES: 6
MEMTYPE:   DATA

          ----ALPHABETIC LIST OF VARIABLES AND ATTRIBUTES-----
       # VARIABLE TYPE   LENGTH POSITION FORMAT       INFORMAT     LABEL
       5 COUNT    NUM         8       26
       2 DATE     NUM         8       12
       4 DEST     CHAR        3       23
       1 FLIGHTNO NUM         8        4
       3 ORIGIN   CHAR        3       20
       6 REVENUE  NUM         8       34


------------------------------ SOURCE RECORDS --------------------------
| DATA FLIGHT.QTR1;
|    SET FLIGHT.JAN FLIGHT.FEB FLIGHT.MAR;
------------------------------------------------------------------------
```

Deleting and Renaming SAS Data Sets

Example: Write system commands to access the SAS data library and the raw data for April.

VSE batch

```
// DLBL APRIL,'EDU.APRFLY.DATA'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
// DLBL FLIGHT,'EDU.AIRLINE.SASDATA'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
```

ICCF

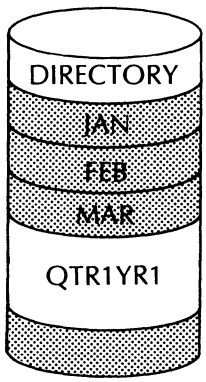
```
/FILE NAME=APRIL,ID='EDU.APRFLY.DATA',UNITS=SYS055,
/ SERIAL=DOSRES
/FILE NAME=FLIGHT,ID='EDU.AIRLINE.SASDATA',UNITS=SYS055,
/ SERIAL=DOSRES
```

Note: under ICCF, the logical unit and SYS055 must be assigned to DOSRES in the ICCF start-up job.

Deleting and Renaming SAS Data Sets

Example: Delete the three monthly data sets and rename the first-quarter data set.

```
PROC DATASETS LIBRARY=FLIGHT;
   DELETE JAN FEB MAR;
   CHANGE QTR1=QTR1YR1;
```



Deleting and Renaming SAS Data Sets

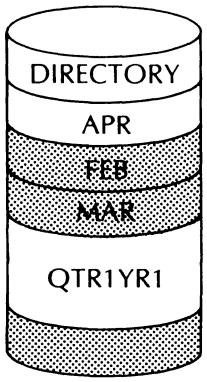
Partial SAS Log

```
LIST OF MEMBERS BEFORE UPDATE OF DIRECTORY.
NAME     MEMTYPE             OBS  TRACKS PROT
FEB      /DATA                19       1
JAN      /DATA                35       1
MAR      /DATA                47       1
QTR1     /DATA               101       1
LIST OF MEMBERS AFTER UPDATE OF DIRECTORY.
NAME     MEMTYPE             OBS  TRACKS PROT
QTR1YR1  /DATA               101       1
```

Adding Data Sets to the Library

Example: Store the April data in a SAS data set in the SAS data library.

```
DATA FLIGHT.APR;
   INFILE APRIL RECFM=FB LRECL=80 BLKSIZE=9040;
   INPUT FLIGHTNO 1-3 DATE MMDDYY8. ORIGIN $ 12-14
         DEST $ 15-17 COUNT 18-19 REVENUE 20-24;
```



Copying a Library

Example: Use another SAS program to create a backup of the library.

VSE batch

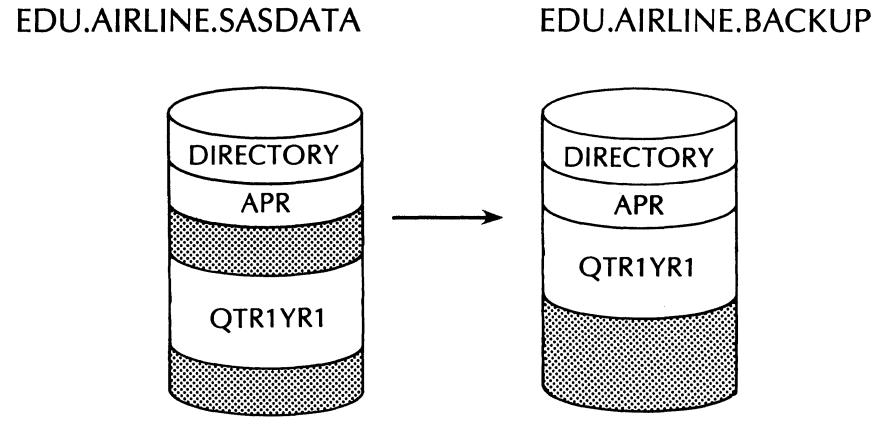
```
// DLBL IOLD,'EDU.AIRLINE.SASDATA'
// EXTENT SYS055,DOSRES
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
// DLBL ONEW,'EDU.AIRLINE.BACKUP',99/365
// EXTENT SYS055,DOSRES,1,0,9970,15
// ASSGN SYS055,DISK,VOL=DOSRES,SHR
```

ICCF

```
/FILE NAME=IOLD,ID='EDU.AIRLINE.SASDATA',UNITS=SYS055,
/ SERIAL=DOSRES
/FILE NAME=ONEW,ID='EDU.AIRLINE BACKUP',UNITS=SYS055,
/ SERIAL=DOSRES,DATE=99/365,LOC=9970,15
```

Note: under ICCF, the logical unit SYS055 must be assigned to DOSRES in the ICCF start-up job.

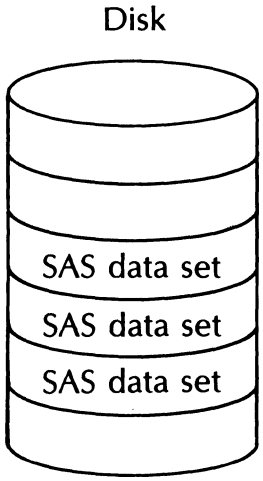
```
PROC COPY IN=IOLD OUT=ONEW;
```



14.4 Minicomputer Environments

SAS Data Libraries on Disk

In the minicomputer environment, a SAS data library is a collection of SAS data sets located in the same directory.



Two-Level Data Set Names

Every SAS data set has a two-level name:

libref.SASdataset

libref (short for SAS data library reference) is a name that is associated with a directory.

The libref is associated with the appropriate directory through a LIBNAME statement.

SASdataset refers to the SAS data set that is being created (or read) (for example, CLASS or PAYROLL).

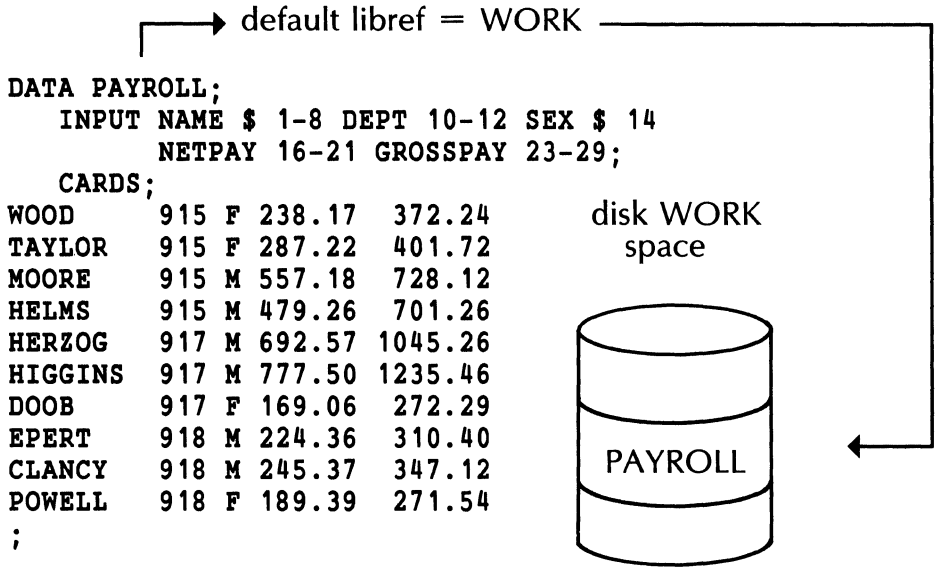
General form of the LIBNAME statement:

LIBNAME *libref 'directory'*;

If a libref is not specified, the default libref WORK is used.

WORK data sets are stored in temporary work space and exist only for the duration of the job or session.

Temporary SAS Data Sets



The note in the SAS log for this DATA step would read:

```
NOTE: DATA SET WORK.PAYROLL HAS 10 OBSERVATIONS AND 5 VARIABLES.
```

The data set WORK.PAYROLL has been stored in WORK space and will be deleted at the end of the job or session.

The data set may be referred to as PAYROLL or WORK.PAYROLL.

```
PROC PRINT DATA=WORK.PAYROLL;
```

is equivalent to

```
PROC PRINT DATA=PAYROLL;
```

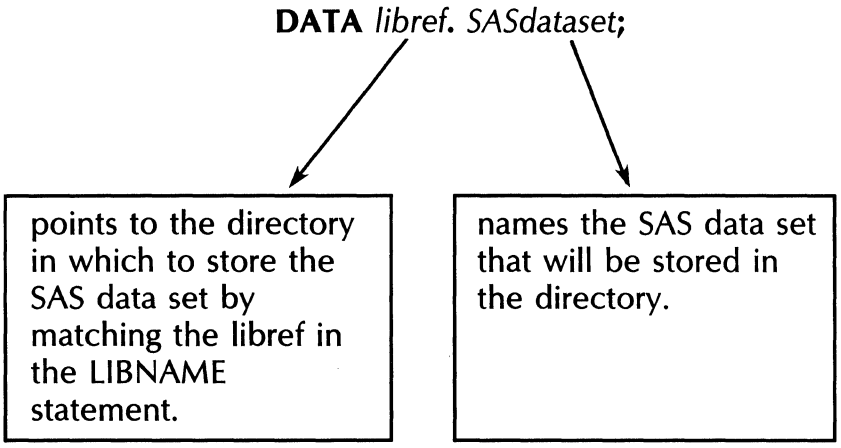
Why Store SAS Data Sets?

There are several advantages to storing data in SAS data sets:

- SAS data sets are self-documenting.
- The data can be directly accessed by SAS procedures.
- The data can be directly accessed by SET, MERGE, and UPDATE statements (in other words, an INPUT statement is not needed).
- Data conversion is not necessary (computer time is saved since the data are stored in the form that the SAS System requires).

Permanently Storing SAS Data Sets

- A SAS data set can be permanently stored by specifying a libref other than WORK when the data set is created.



- A LIBNAME statement with the same libref must be specified to point to the appropriate directory.

Permanently Storing SAS Data Sets

Example: An airline company stores its accounting information in raw data files for each month.

Store January and February data in separate SAS data sets within the same directory.

Use FILENAME statements to point to the raw data files and a LIBNAME statement to point to the directory in which the SAS data sets will be stored.

VMS

```
FILENAME DATA1   '[AIRLINE.FLIGHT]JAN.DAT'
         DATA2   '[AIRLINE.FLIGHT]FEB.DAT';
LIBNAME  FLIGHT '[AIRLINE.SASDATA]';
```

AOS/VS

```
FILENAME DATA1   ':UDD:AIRLINE:FLIGHT:JAN.DAT'
         DATA2   ':UDD:AIRLINE:FLIGHT:FEB.DAT';
LIBNAME  FLIGHT ':UDD:AIRLINE:SASDATA';
```

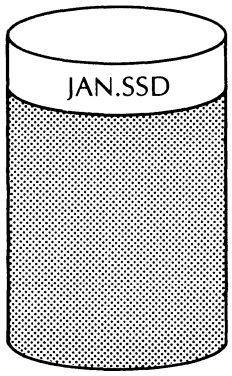
PRIMOS

```
FILENAME DATA1   '<MFD>AIRLINE>FLIGHT>JAN.DAT'
         DATA2   '<MFD>AIRLINE>FLIGHT>FEB.DAT';
LIBNAME  FLIGHT '<MFD>AIRLINE>SASDATA';
```

Permanently Storing SAS Data Sets

Example: Read the January raw data from an external file and store the SAS data set in the SAS data library.

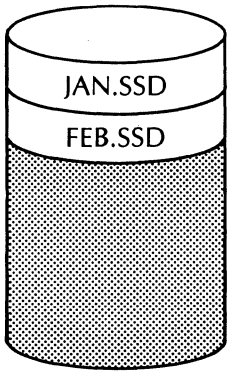
```
DATA FLIGHT.JAN;
   INFILE DATA1;
   INPUT FLIGHTNO 1-3 DATE MMDDYY8. ORIGIN $ 12-14
         DEST $ 15-17 COUNT 18-19 REVENUE 20-24;
```



Permanently Storing SAS Data Sets

Example: Read the February raw data file and store the SAS data set in the SAS data library.

```
DATA FLIGHT.FEB;
   INFILE DATA2;
   INPUT FLIGHTNO 1-3 DATE MMDDYY8. ORIGIN $ 12-14
         DEST $ 15-17 COUNT 18-19 REVENUE 20-24;
```



Permanently Storing SAS Data Sets

Example: Use another SAS program to place March flight data in the SAS data library and to concatenate the first-quarter data.

VMS

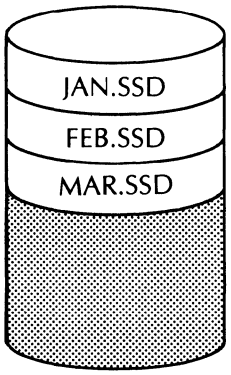
```
FILENAME MARCH  '[AIRLINE.FLIGHT]MARCH.DAT';
LIBNAME  FLIGHT '[AIRLINE.SASDATA]';
```

AOS/VS

```
FILENAME MARCH  ':UDD:AIRLINE:FLIGHT:MARCH.DAT';
LIBNAME  FLIGHT ':UDD:AIRLINE:SASDATA';
```

PRIMOS

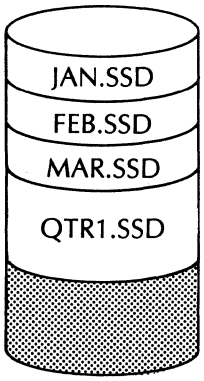
```
FILENAME MARCH  '<MFD>AIRLINE>FLIGHT>MARCH.DAT';
LIBNAME  FLIGHT '<MFD>AIRLINE>SASDATA';


DATA FLIGHT.MAR;
   INFILE MARCH;
   INPUT FLIGHTNO 1-3 DATE MMDDYY8. ORIGIN $ 12-14
         DEST $ 15-17 COUNT 18-19 REVENUE 20-24;
```



Permanently Storing SAS Data Sets

Example: Concatenate the three SAS data sets into one SAS data set.

```
DATA FLIGHT.QTR1;
   SET FLIGHT.JAN FLIGHT.FEB FLIGHT.MAR;
```



SAS Utility Procedures

The SAS System provides several utility procedures for managing SAS data libraries.

CONTENTS prints descriptions of the contents of one or more data sets in a SAS data library.

COPY copies an entire SAS data library or selected members of the library.

DATASETS deletes and renames SAS data sets stored in a SAS data library on disk.

General form of the PROC CONTENTS statement:

PROC CONTENTS *options*;

Selected options:

DATA=*libref.member*
refers to an entire library or a single data set in the library.

member can be

- *SASdataset* to refer to a specific SAS data set in the library
- _ALL_ to refer to all SAS data sets in the library.

NODS
suppresses printing the contents of individual SAS data sets when _ALL_ is specified.

SAS Utility Procedures

General form of the PROC COPY statement:

PROC COPY IN=*libref* **OUT=***libref options*;

Selected statements used with PROC COPY:

SELECT *SASdataset . . . / options*;

EXCLUDE *SASdataset . . . / options*;

Note: if neither a SELECT nor EXCLUDE statement is used, the entire library is copied.

General form of the PROC DATASETS statement:

PROC DATASETS LIBRARY=*libref options*;

Selected statements used with PROC DATASETS:

DELETE *SASdataset . . . / options*;

SAVE *SASdataset . . . / options*;

CHANGE *oldname=newname . . . / options*;

Note: PROC DATASETS can be executed in line mode or full-screen mode.

The NOFS option in the PROC DATASETS statement puts the procedure in line mode.

Displaying the Contents of a Library

Example: Display the names of all the SAS data sets in the library, but do not show any individual data set information.

VMS

```
LIBNAME FLIGHT '[AIRLINE.SASDATA]';
```

AOS/VS

```
LIBNAME FLIGHT ':UDD:AIRLINE:SASDATA';
```

PRIMOS

```
LIBNAME FLIGHT '<MFD>AIRLINE>SASDATA';

PROC CONTENTS DATA=FLIGHT._ALL_ NODS;
```

Displaying the Contents of a Library

```
                                SAS

                           CONTENTS PROCEDURE
*****LISTING OF SAS DATA LIBRARY MEMBER NAMES*****
     LIBNAME=FLIGHT
FEB
JAN
MAR
QTR1
```

Displaying the Contents of a Library

Example: Display the contents of the QTR1 data set.

```
PROC CONTENTS DATA=FLIGHT.QTR1;
```

```
                                SAS

                          CONTENTS PROCEDURE
                    CONTENTS OF SAS MEMBER FLIGHT.QTR1

NUMBER OF OBSERVATIONS: 101              NUMBER OF VARIABLES: 6
MEMTYPE:             DATA

          ----ALPHABETIC LIST OF VARIABLES AND ATTRIBUTES-----
       #  VARIABLE TYPE   LENGTH POSITION FORMAT       INFORMAT     LABEL
       5  COUNT    NUM         8       22
       2  DATE     NUM         8        8
       4  DEST     CHAR        3       19
       1  FLIGHTNO NUM         8        0
       3  ORIGIN   CHAR        3       16
       6  REVENUE  NUM         8       30
```

Deleting and Renaming SAS Data Sets

Example: Delete the three monthly data sets and rename the first-quarter data set.

VMS

```
FILENAME APRIL  '[AIRLINE.FLIGHT]APRIL.DAT';
LIBNAME  FLIGHT '[AIRLINE.SASDATA]';
```

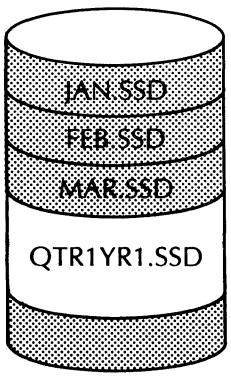
AOS/VS

```
FILENAME APRIL  ':UDD:AIRLINE:FLIGHT:APRIL.DAT';
LIBNAME  FLIGHT ':UDD:AIRLINE:SASDATA';
```

PRIMOS

```
FILENAME APRIL  '<MFD>AIRLINE>FLIGHT>APRIL.DAT';
LIBNAME  FLIGHT '<MFD>AIRLINE>SASDATA';
```

```
PROC DATASETS LIBRARY=FLIGHT;
   DELETE JAN FEB MAR;
   CHANGE QTR1=QTR1YR1;
```



Deleting and Renaming SAS Data Sets

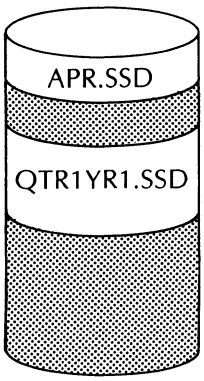
Partial SAS Log

```
LIST OF MEMBERS BEFORE UPDATE OF DIRECTORY.
NAME     MEMTYPE
FEB      /DATA
JAN      /DATA
MAR      /DATA
QTR1     /DATA
LIST OF MEMBERS AFTER UPDATE OF DIRECTORY.
NAME     MEMTYPE
QTR1YR1  /DATA
```

Deleting and Renaming SAS Data Sets

Example: Store the APRIL data in a SAS data set in the SAS data library.

```
DATA FLIGHT.APR;
   INFILE APRIL;
   INPUT FLIGHTNO 1-3 DATE MMDDYY8. ORIGIN $ 12-14
         DEST $ 15-17 COUNT 18-19 REVENUE 20-24;
```



Copying a SAS Data Library

Example: Use another SAS program to create a backup of the library.

VMS

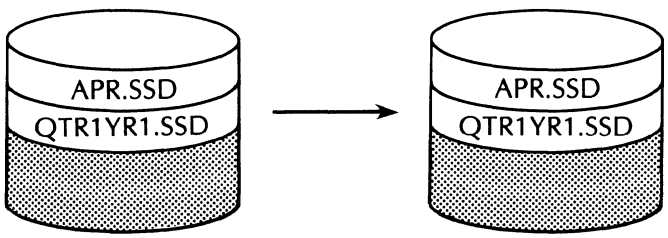
```
LIBNAME BACKUP '[AIRLINE.BACKUP]'
        FLIGHT '[AIRLINE.SASDATA]';
```

AOS/VS

```
LIBNAME BACKUP ':UDD:AIRLINE:BACKUP'
        FLIGHT ':UDD:AIRLINE:SASDATA';
```

PRIMOS

```
LIBNAME BACKUP '<MFD>AIRLINE>BACKUP'
        FLIGHT '<MFD>AIRLINE>SASDATA';


PROC COPY IN=FLIGHT OUT=BACKUP;
```



14.5 Exercises

14.1 The National Safety Council annually compiles information on accidental deaths in the home. Each record of information represents one year of data. The record layout is given in the table below.

Field Description	Field Location	Variable Type
Year	1-4	numeric
Falls	5-9	numeric
Fires, burns	10-14	numeric
Suffocation, ingested	15-19	numeric
Suffocation, mechanical	20-24	numeric
Poison, solid or liquid	25-29	numeric
Poison by gas	30-34	numeric
Firearms	35-39	numeric
Other	40-44	numeric

Assume that the data file has been referred to with a system command with fileref=ACCIDENT.

a. Create and permanently store a SAS data set named ACCIDNTS from the raw data file. Include the variables listed on the next page. Assume that the space for the file that will contain the SAS data set has already been provided through a system command or LIBNAME statement with libref=HOME. (CMS users should assume that no system command was used. However, the filetype of the stored data should be HOME.)

Exercises

14.1

Variable Name	Description (cause of death)
YEAR	year
FALLS	falls
BURNS	fires or burns
INGEST	suffocation, ingested
MECH	suffocation, mechanical
SOLID	poison, solid or liquid
GAS	poison by gas
GUNS	firearms
OTHER	other
TOTAL	total deaths by all causes

b. Print the descriptive information in the data set.

Exercises

14.2 One week later, the data set created in Exercise 14.1 is needed for a report. Assume that the file containing the SAS data set has been referenced through a system command or LIBNAME statement with libref=ACCID (assume no system command was used if under CMS).

a. Print the data set.

b. Compute the average number of deaths over the years for each cause and for all causes.

Exercises

14.3 The National Center for Health Statistics keeps records on the numbers of marriages and divorces for each state for each year. The data are stored on a disk file with the following record layout.

Field Description	Field Location	Variable Type
Year	1-4	numeric
State	5-19	character
Number of marriages	20-25	numeric
Number of divorces	26-31	numeric

a. Create a SAS data set that contains the data stored in the file. Choose appropriate variable names. Assume that the file has been referenced through a system command with fileref=COUPLES. Store the SAS data set in the same file that contains the SAS data set created in Exercise 14.1. Assume that the file that contains the SAS data set has been referenced through a system command or LIBNAME statement with libref=MOREDATA. (CMS users should assume that no system command was used. However, the filetype for this data set should be the same as that for the data set ACCIDNTS created in Exercise 14.1.)

b. Compute the total number of marriages and total number of divorces in the United States for each year.

c. Compute the average number of marriages and average number of divorces over the years for each state.

Exercises

14.4 Create a new SAS data set named HOMACCID by sorting the ACCIDNTS data set created in Exercise 14.1 by TOTAL. Assume the file has been referenced through a system command or LIBNAME statement with libref=HOME. The HOMACCID data set should be permanently stored in the same file (CMS users should use the same filetype).

14.5 Data on births by state in the United States have been recorded yearly. The data are on disk with the following record format.

Variable Name	Field Location	Variable Type
YEAR	1-4	numeric
STATE (postal code)	5-6	character
MALES	7-13	numeric
FEMALES	14-20	numeric

Assume the file has been referenced through a system command with fileref=BORN.

a. Create a SAS data set with the variables listed above and add a variable called TOTAL that equals the male and female births combined. Store this data set with the data sets created in the previous exercises. Select an appropriate libref for the system command.

b. Superimpose a plot of male births versus year and a plot of female births versus year on the same set of axes. Use the letter 'M' as the plotting symbol for males and 'F' for females.

15. Introduction to Report Writing

15.1 Writing Reports with PROC PRINT

15.2 Customized Report Writing

15.3 Advanced Report Writing Examples

15.4 Unusual Reports

15.5 Exercises

15.1 Writing Reports with PROC PRINT

PROC PRINT Report Writing Features

Several statements and options are available with PROC PRINT to produce simple, structured reports.

- A **VAR statement** names and orders the variable(s) to be printed.
- An **ID statement** names the variable(s) to be printed at the beginning of each line.
- A **BY statement** separates groups for subtotaling and page ejects.
- A **SUM statement** identifies the numeric variables whose values are to be totaled.
- The **PAGEBY statement** begins printing a new page at the start of specified BY groups.
- A **FORMAT statement** can be used to associate specific formats with selected variables.
- **TITLE statements** can be used to specify up to ten title lines.
- **FOOTNOTE statements** can be used to specify up to ten footnote lines.
- The **LABEL** and **SPLIT=** options specify that the PRINT procedure should use labels that have been defined in LABEL statements.

A Simple Report with PROC PRINT

Example: Use the PRINT procedure to generate a simple first quarter sales report for Suny Stereo, Inc.

Create a SAS data set from the raw data. The fields from left to right on the records are January sales, February sales, March sales, location of sale (city), and product sold.

Create a variable that represents year-to-date sales.

```
DATA SALES;
   INPUT JAN 8. FEB 8. MAR 8. +1 CITY $6. PRODUCT $10.;
   YTD=SUM(JAN,FEB,MAR);
   CARDS;

data lines
```

A Simple Report with PROC PRINT

Example: Arrange the data so that the cities are in alphabetical order and the products are arranged in alphabetical order within cities.

Print the data so that the variables listed from left to right are city, product, January sales, February sales, March sales, and year-to-date sales.

Suppress the observation number.

```
PROC SORT;
   BY CITY PRODUCT;
PROC PRINT;
   ID CITY;
   VAR PRODUCT JAN FEB MAR YTD;
```

CITY	PRODUCT	JAN	FEB	MAR	YTD
BOSTON	AMPLIFIER	19321.5	18417.5	21562.4	59301.4
BOSTON	RECEIVERS	24217.1	21015.3	19218.3	64450.7
BOSTON	SPEAKERS	30548.1	32847.2	29430.4	92825.7
BOSTON	TAPE DECK	31072.7	31243.9	32945.3	95261.9
BOSTON	TURNTABLE	12157.1	15378.6	13582.3	41118.0
DALLAS	AMPLIFIER	24672.5	19143.0	16588.1	60403.6
DALLAS	RECEIVERS	23476.5	24981.8	20611.7	69070.0
DALLAS	SPEAKERS	25934.5	28316.0	27186.1	81436.6
DALLAS	TAPE DECK	34145.4	31427.4	30661.8	96234.6
DALLAS	TURNTABLE	19161.5	19445.7	18892.3	57499.5
MIAMI	AMPLIFIER	24519.2	27210.5	21546.0	73275.7
MIAMI	RECEIVERS	28349.3	24318.1	29783.5	82450.9
MIAMI	SPEAKERS	21926.0	22905.0	24936.9	69767.9
MIAMI	TAPE DECK	29816.0	31233.2	26984.5	88033.7
MIAMI	TURNTABLE	16283.8	18122.3	19323.8	53729.9

Enhance the Report

Example: Print the data in groups by city.

Place titles on the output.

Generate city subtotals and a grand total for product sales.

```
PROC PRINT;
   ID CITY;
   BY CITY;
   VAR PRODUCT JAN FEB MAR YTD;
   SUM JAN FEB MAR YTD;
   TITLE 'SUNY STEREO, INC.';
   TITLE2 'FIRST QUARTER SALES';
```

Enhance the Report

SUNY STEREO, INC.
FIRST QUARTER SALES

CITY	PRODUCT	JAN	FEB	MAR	YTD
BOSTON	AMPLIFIER	19322	18418	21562	59301
	RECEIVERS	24217	21015	19218	64451
	SPEAKERS	30548	32847	29430	92826
	TAPE DECK	31073	31244	32945	95262
	TURNTABLE	12157	15379	13582	41118
BOSTON		117316	118902	116739	352958
DALLAS	AMPLIFIER	24673	19143	16588	60404
	RECEIVERS	23477	24982	20612	69070
	SPEAKERS	25935	28316	27186	81437
	TAPE DECK	34145	31427	30662	96235
	TURNTABLE	19162	19446	18892	57499
DALLAS		127390	123314	113940	364644
MIAMI	AMPLIFIER	24519	27211	21546	73276
	RECEIVERS	28349	24318	29784	82451
	SPEAKERS	21926	22905	24937	69768
	TAPE DECK	29816	31233	26985	88034
	TURNTABLE	16284	18122	19324	53730
MIAMI		120894	123789	122575	367258
		365601	366005	353253	1084860

Final Report

Example: Change the column headings for JAN, FEB, and MAR to JANUARY, FEBRUARY, and MARCH, respectively.

Change the column heading for YTD to

TOTAL
TO DATE.

Print all numbers with commas and zero decimal places.

```
PROC PRINT SPLIT='+' DATA=SALES;
   LABEL JAN='JANUARY' FEB='FEBRUARY' MAR='MARCH'
         YTD=' TOTAL+TO DATE';
   ID CITY;
   BY CITY;
   VAR PRODUCT JAN FEB MAR YTD;
   SUM JAN FEB MAR YTD;
   FORMAT JAN FEB MAR YTD COMMA12.;
   TITLE 'SUNY STEREO, INC.';
   TITLE2 'FIRST QUARTER SALES';
```

Final Report

SUNY STEREO, INC.
FIRST QUARTER SALES

CITY	PRODUCT	JANUARY	FEBRUARY	MARCH	TOTAL TO DATE
BOSTON	AMPLIFIER	19,322	18,418	21,562	59,301
	RECEIVERS	24,217	21,015	19,218	64,451
	SPEAKERS	30,548	32,847	29,430	92,826
	TAPE DECK	31,073	31,244	32,945	95,262
	TURNTABLE	12,157	15,379	13,582	41,118
BOSTON		117,316	118,902	116,739	352,958
DALLAS	AMPLIFIER	24,673	19,143	16,588	60,404
	RECEIVERS	23,477	24,982	20,612	69,070
	SPEAKERS	25,935	28,316	27,186	81,437
	TAPE DECK	34,145	31,427	30,662	96,235
	TURNTABLE	19,162	19,446	18,892	57,499
DALLAS		127,390	123,314	113,940	364,644
MIAMI	AMPLIFIER	24,519	27,211	21,546	73,276
	RECEIVERS	28,349	24,318	29,784	82,451
	SPEAKERS	21,926	22,905	24,937	69,768
	TAPE DECK	29,816	31,233	26,985	88,034
	TURNTABLE	16,284	18,122	19,324	53,730
MIAMI		120,894	123,789	122,575	367,258
		365,601	366,005	353,253	1,084,860

15.2 Customized Report Writing

PUT and FILE Statement Features

The PUT statement can be used to write detailed reports or reports printed on special forms.

The FILE statement defines the destination of lines written with PUT statements and provides options that are useful for writing reports.

- The special fileref **PRINT** on the FILE statement directs lines written by PUT statements to the standard SAS print file.
- The **NOTITLES** option on the FILE statement suppresses any currently defined titles.
- The **HEADER=** option on the FILE statement provides a facility for automatically writing user-defined headings whenever you start a new page.
- The **N=PAGESIZE** (or **N=PS**) option on the FILE statement allows the user to access a whole page at a time.
- Since PUT and FILE statements are used in the DATA step, the full power of SAS programming statements is available for writing reports.
- The following pointer controls can be used on the PUT statement:

  @*n* go to column *n*

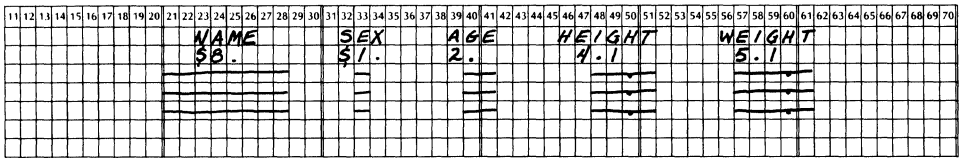
  +*n* move the pointer *n* positions

  / move the pointer to column 1 of the next line

  #*n* move the pointer to column 1 of line number *n*.

Write a Report with a PUT Statement

Example: Use PUT and FILE statements to produce a listing of the data in the CLASS data set.



Write a Report with a PUT Statement

```
DATA _NULL_;
   SET CLASS;
   FILE PRINT;
   PUT @21 NAME $8. @33 SEX $1. @40 AGE 2.
       @48 HEIGHT 4.1 @57 WEIGHT 5.1;
```

```
                    SAS
JOHN        M       12      59.0      99.5
JAMES       M       12      57.3      83.0
ALFRED      M       14      69.0     112.5
WILLIAM     M       15      66.5     112.0
JEFFREY     M       13      62.5      84.0
RONALD      M       15      67.0     133.0
THOMAS      M       11      57.5      85.0
PHILIP      M       16      72.0     150.0
ROBERT      M       12      64.8     128.0
HENRY       M       14      63.5     102.5
JANET       F       15      62.5     112.5
JOYCE       F       11      51.3      50.5
JUDY        F       14      64.3      90.0
CAROL       F       14      62.8     102.5
JANE        F       12      59.8      84.5
LOUISE      F       12      56.3      77.0
BARBARA     F       13      65.3      98.0
MARY        F       15      66.5     112.0
ALICE       F       13      56.5      84.0
```

Note: the default title, SAS, is turned off in display manager output in some operating systems.

Enhance the Report

Example: Suppress the default title (NOTITLES option).

Add column headings (HEADER= option).

```
DATA _NULL_;
   SET CLASS;
   FILE PRINT NOTITLES HEADER=COLHEAD;
   PUT @21 NAME $8. @33 SEX $1. @40 AGE 2.
       @48 HEIGHT 4.1 @57 WEIGHT 5.1;
   RETURN;
 COLHEAD:
   PUT @21 'NAME' @32 'SEX' @39 'AGE' @46 'HEIGHT'
       @56 'WEIGHT';
   RETURN;
```

NAME	SEX	AGE	HEIGHT	WEIGHT
JOHN	M	12	59.0	99.5
JAMES	M	12	57.3	83.0
ALFRED	M	14	69.0	112.5
WILLIAM	M	15	66.5	112.0
JEFFREY	M	13	62.5	84.0
RONALD	M	15	67.0	133.0
THOMAS	M	11	57.5	85.0
PHILIP	M	16	72.0	150.0
ROBERT	M	12	64.8	128.0
HENRY	M	14	63.5	102.5
JANET	F	15	62.5	112.5
JOYCE	F	11	51.3	50.5
JUDY	F	14	64.3	90.0
CAROL	F	14	62.8	102.5
JANE	F	12	59.8	84.5
LOUISE	F	12	56.3	77.0
BARBARA	F	13	65.3	98.0
MARY	F	15	66.5	112.0
ALICE	F	13	56.5	84.0

Final Report

Example: Add a column for the observation number.

Skip a line between the heading and the first observation.

```
DATA _NULL_;
   SET CLASS;
   FILE PRINT NOTITLES HEADER=COLHEAD;
   PUT @15 _N_ 2. @21 NAME $8. @33 SEX $1.
       @40 AGE 2. @48 HEIGHT 4.1 @57 WEIGHT 5.1;
   RETURN;
 COLHEAD:
   PUT @14 'OBS' @21 'NAME' @32 'SEX' @39 'AGE'
       @46 'HEIGHT' @56 'WEIGHT' /;
   RETURN;
```

Final Report

OBS	NAME	SEX	AGE	HEIGHT	WEIGHT
1	JOHN	M	12	59.0	99.5
2	JAMES	M	12	57.3	83.0
3	ALFRED	M	14	69.0	112.5
4	WILLIAM	M	15	66.5	112.0
5	JEFFREY	M	13	62.5	84.0
6	RONALD	M	15	67.0	133.0
7	THOMAS	M	11	57.5	85.0
8	PHILIP	M	16	72.0	150.0
9	ROBERT	M	12	64.8	128.0
10	HENRY	M	14	63.5	102.5
11	JANET	F	15	62.5	112.5
12	JOYCE	F	11	51.3	50.5
13	JUDY	F	14	64.3	90.0
14	CAROL	F	14	62.8	102.5
15	JANE	F	12	59.8	84.5
16	LOUISE	F	12	56.3	77.0
17	BARBARA	F	13	65.3	98.0
18	MARY	F	15	66.5	112.0
19	ALICE	F	13	56.5	84.0

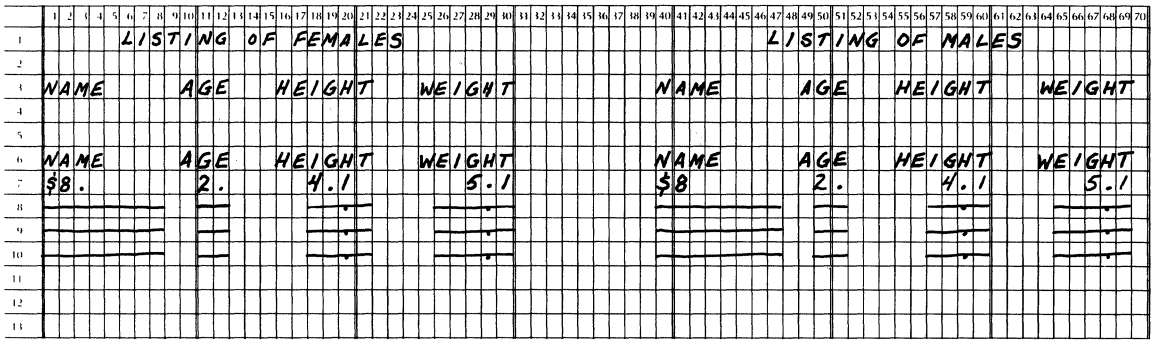
Controlling an Entire Page

Example: Generate a multicolumn report from the CLASS data set.

Put females on the left side of the page.

Put males on the right side of the same page.

Use appropriate column headings.



Controlling an Entire Page

Example: Use the N=PS option to access a whole page.

```
DATA _NULL_;
   RETAIN FLINE MLINE 5;
   SET CLASS;
   FILE PRINT NOTITLES HEADER=H N=PS;
   IF SEX='F' THEN
      DO;
         FLINE+1;
         PUT #FLINE @1 NAME $8. +2 AGE 2. +5 HEIGHT 4.1
             +4 WEIGHT 5.1;
      END;
   ELSE
      DO;
         MLINE+1;
         PUT #MLINE @40 NAME $8. +2 AGE 2. +5 HEIGHT 4.1
             +4 WEIGHT 5.1;
      END;
   RETURN;
 H:
   PUT @6 'LISTING OF FEMALES' @47 'LISTING OF MALES' //
       @1 'NAME' @10 'AGE' @16 'HEIGHT' @25 'WEIGHT'
       @40 'NAME' @49 'AGE' @55 'HEIGHT' @64 'WEIGHT';
   RETURN;
```

Generated Report

LISTING OF FEMALES

NAME	AGE	HEIGHT	WEIGHT
JANET	15	62.5	112.5
JOYCE	11	51.3	50.5
JUDY	14	64.3	90.0
CAROL	14	62.8	102.5
JANE	12	59.8	84.5
LOUISE	12	56.3	77.0
BARBARA	13	65.3	98.0
MARY	15	66.5	112.0
ALICE	13	56.5	84.0

LISTING OF MALES

NAME	AGE	HEIGHT	WEIGHT
JOHN	12	59.0	99.5
JAMES	12	57.3	83.0
ALFRED	14	69.0	112.5
WILLIAM	15	66.5	112.0
JEFFREY	13	62.5	84.0
RONALD	15	67.0	133.0
THOMAS	11	57.5	85.0
PHILIP	16	72.0	150.0
ROBERT	12	64.8	128.0
HENRY	14	63.5	102.5

15.3 Advanced Report Writing Examples

Data for Upcoming Examples

Example: Create a SAS data set named PAYROLL and print it.

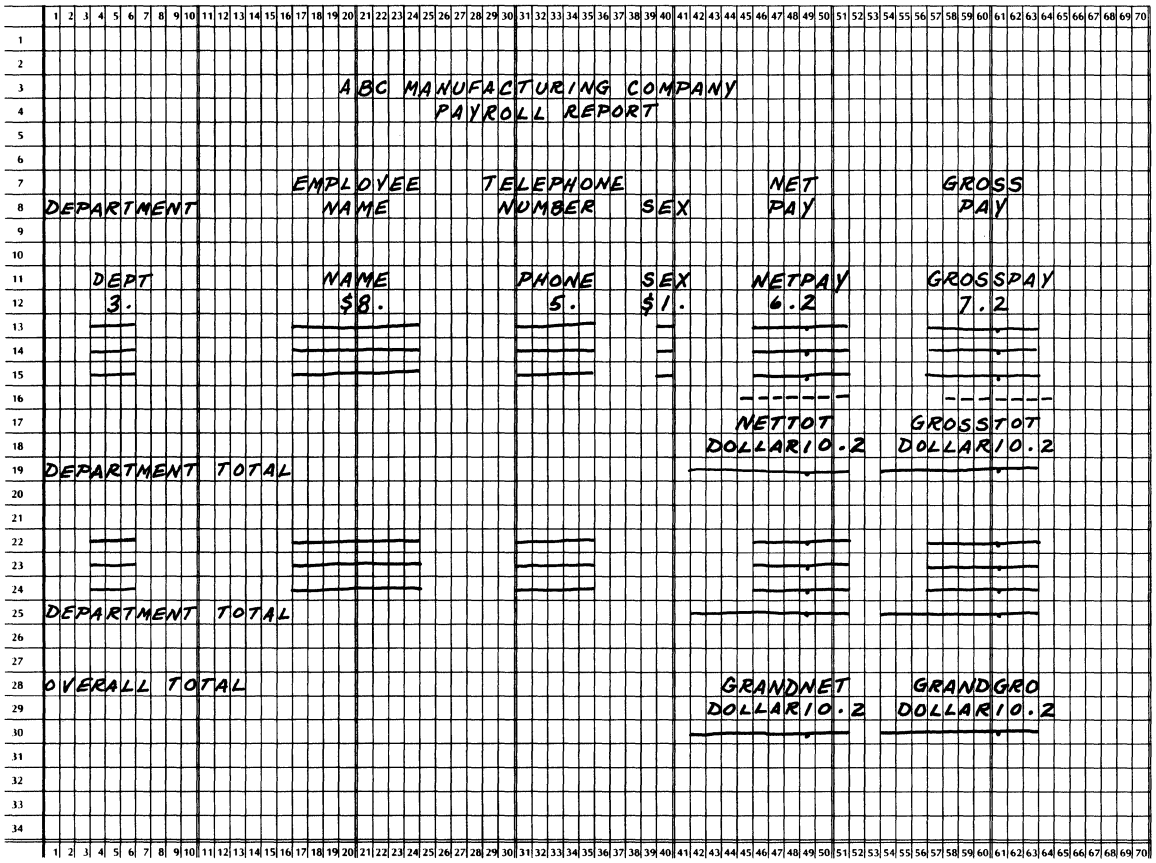
```
DATA PAYROLL;
   INPUT NAME $ 1-8 DEPT 10-12 SEX $ 14
         NETPAY 16-21 GROSSPAY 23-29 PHONE 31-34;
   CARDS;

data lines

PROC PRINT DATA=PAYROLL;
```

Data for Upcoming Examples

OBS	NAME	DEPT	SEX	NETPAY	GROSSPAY	PHONE
1	YOUNG	911	M	229.69	313.60	6263
2	REYNOLDS	911	F	134.03	174.15	1727
3	STRIDE	911	M	272.53	383.40	7281
4	GREEN	911	M	238.04	365.60	2182
5	SMOTHERS	911	M	202.43	315.20	8272
6	KRAUSE	911	F	182.09	242.40	7222
7	POWELL	911	M	167.53	243.95	2187
8	LARSON	914	F	215.47	283.92	5262
9	ARNOLD	914	M	356.87	445.50	6473
10	MANHART	914	M	250.34	344.78	8273
11	CURCI	914	M	215.31	376.00	2638
12	GRECO	914	M	685.23	1004.00	2723
13	JOHNSON	914	M	135.23	180.72	2765
14	THOMPSON	914	M	221.09	297.56	2343
15	CORNING	915	F	103.43	146.16	1273
16	WOOD	915	F	238.17	372.24	1734
17	GREEN	915	M	725.16	1124.51	2156
18	TAYLOR	915	F	287.22	401.72	8374
19	CHAPMAN	915	M	264.31	385.47	2154
20	MOORE	915	M	557.18	728.12	2358
21	FARLOW	916	M	527.19	842.03	3455
22	SHIRES	916	F	187.12	279.22	4353
23	RICHARDS	916	M	219.27	352.84	7787
24	SMITH	916	F	147.09	201.88	2232
25	MURPHY	916	M	272.66	385.11	2099
26	HERZOG	917	M	692.57	1045.26	7440
27	TALL	917	M	355.19	492.26	3830
28	HIGGINS	917	M	777.50	1235.46	5667
29	MCKAY	917	M	821.44	1392.27	3748
30	SANFORD	917	F	284.06	405.37	3321
31	BRANDON	918	M	554.31	804.64	4043
32	EPERT	918	M	224.36	310.40	3900
33	CLANCY	918	M	245.37	347.12	2604
34	POWELL	918	F	189.39	271.54	3017

Generating a Payroll Report

Example: Use PUT and FILE statements to generate a payroll report.



Generating a Payroll Report

```
PROC SORT DATA=PAYROLL;
   BY DEPT;
DATA _NULL_;
   SET PAYROLL END=EOF;
   BY DEPT;
   FILE PRINT HEADER=H NOTITLES LINESLEFT=LL;
   IF FIRST.DEPT THEN
      DO;
         NETTOT=0;  GROSSTOT=0;
         IF LL<14 THEN PUT _PAGE_;
      END;
   PUT @4 DEPT 3. @17 NAME $8. @31 PHONE 5.
       @40 SEX $1. @46 NETPAY 6.2
       @57 GROSSPAY 7.2;
   NETTOT+NETPAY;
   GROSSTOT+GROSSPAY;
   IF LAST.DEPT THEN
      DO;
         PUT @45 '-------' @58 '-------' /
             'DEPARTMENT TOTAL'
             @42 NETTOT DOLLAR10.2
             @54 GROSSTOT DOLLAR10.2 //;
         GRANDNET+NETTOT;
         GRANDGRO+GROSSTOT;
      END;
   IF EOF THEN PUT 'OVERALL TOTAL'
      @42 GRANDNET DOLLAR10.2
      @54 GRANDGRO DOLLAR10.2;
   RETURN;
 H:
   PUT // @20 'ABC MANUFACTURING COMPANY' /
      @26 'PAYROLL REPORT' /// @17 'EMPLOYEE'
      @29 'TELEPHONE' @47 'NET' @58 'GROSS' /
      @1 'DEPARTMENT' @19 'NAME' @30 'NUMBER'
      @39 'SEX' @47 'PAY' @59 'PAY' //;
   RETURN;
```

Generating a Payroll Report

ABC MANUFACTURING COMPANY
PAYROLL REPORT

DEPARTMENT	EMPLOYEE NAME	TELEPHONE NUMBER	SEX	NET PAY	GROSS PAY
911	YOUNG	6263	M	229.69	313.60
911	REYNOLDS	1727	F	134.03	174.15
911	STRIDE	7281	M	272.53	383.40
911	GREEN	2182	M	238.04	365.60
911	SMOTHERS	8272	M	202.43	315.20
911	KRAUSE	7222	F	182.09	242.40
911	POWELL	2187	M	167.53	243.95
DEPARTMENT TOTAL				$1,426.34	$2,038.30
914	LARSON	5262	F	215.47	283.92
914	ARNOLD	6473	M	356.87	445.50
914	MANHART	8273	M	250.34	344.78
914	CURCI	2638	M	215.31	376.00
914	GRECO	2723	M	685.23	1004.00
914	JOHNSON	2765	M	135.23	180.72
914	THOMPSON	2343	M	221.09	297.56
DEPARTMENT TOTAL				$2,079.54	$2,932.48
915	CORNING	1273	F	103.43	146.16
915	WOOD	1734	F	238.17	372.24
915	GREEN	2156	M	725.16	1124.51
915	TAYLOR	8374	F	287.22	401.72
915	CHAPMAN	2154	M	264.31	385.47
915	MOORE	2358	M	557.18	728.12
DEPARTMENT TOTAL				$2,175.47	$3,158.22

Generating a Payroll Report

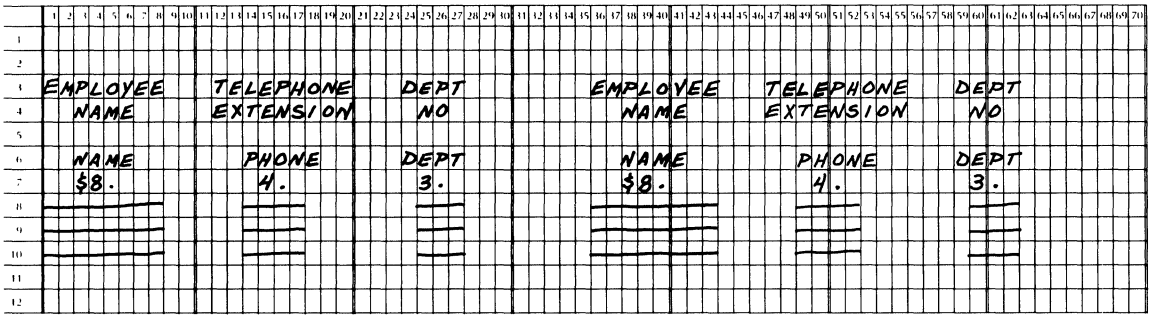
ABC MANUFACTURING COMPANY
PAYROLL REPORT

DEPARTMENT	EMPLOYEE NAME	TELEPHONE NUMBER	SEX	NET PAY	GROSS PAY
916	FARLOW	3455	M	527.19	842.03
916	SHIRES	4353	F	187.12	279.22
916	RICHARDS	7787	M	219.27	352.84
916	SMITH	2232	F	147.09	201.88
916	MURPHY	2099	M	272.66	385.11
DEPARTMENT TOTAL				$1,353.33	$2,061.08
917	HERZOG	7440	M	692.57	1045.26
917	TALL	3830	M	355.19	492.26
917	HIGGINS	5667	M	777.50	1235.46
917	MCKAY	3748	M	821.44	1392.27
917	SANFORD	3321	F	284.06	405.37
DEPARTMENT TOTAL				$2,930.76	$4,570.62
918	BRANDON	4043	M	554.31	804.64
918	EPERT	3900	M	224.36	310.40
918	CLANCY	2604	M	245.37	347.12
918	POWELL	3017	F	189.39	271.54
DEPARTMENT TOTAL				$1,213.43	$1,733.70
OVERALL TOTAL				$11,178.87	$16,494.40

Generating an Employee Directory

Example: Generate an employee directory that displays the employee's name, telephone extension, and department number.

Alphabetically list twenty-five names on the left side of the page followed by nine names on the right side of the page.




```
PROC SORT DATA=PAYROLL;
   BY NAME;
   TITLE 'ABC MANUFACTURING COMPANY';
   TITLE3 'EMPLOYEE TELEPHONE EXTENSION';
DATA _NULL_;
   FILE PRINT N=PS;
   DO C=1,36;
      PUT #3 @C 'EMPLOYEE' +3 'TELEPHONE' +3
          'DEPT' / @C +2 'NAME' +5
          'EXTENSION' +4 'NO';
      DO L=6 TO 30;
         SET PAYROLL;
         PUT #L @C NAME $8. +5 PHONE 4. +7
             DEPT 3.;
      END;
   END;
   PUT _PAGE_;
```

Generating an Employee Directory

ABC MANUFACTURING COMPANY

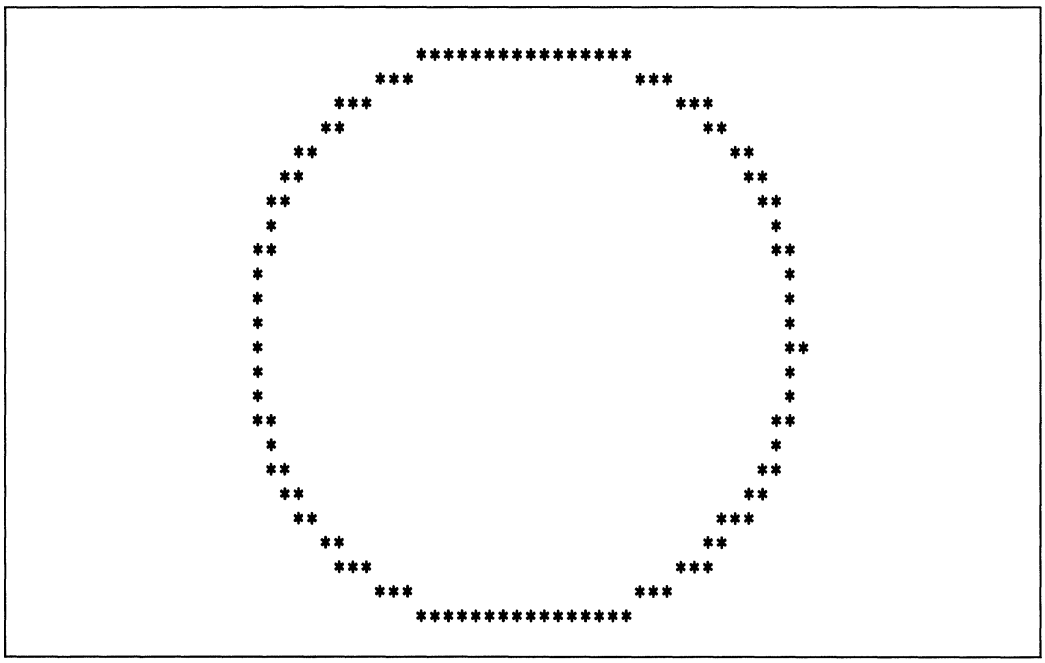
EMPLOYEE TELEPHONE EXTENSION

EMPLOYEE NAME	TELEPHONE EXTENSION	DEPT NO	EMPLOYEE NAME	TELEPHONE EXTENSION	DEPT NO
ARNOLD	6473	914	SHIRES	4353	916
BRANDON	4043	918	SMITH	2232	916
CHAPMAN	2154	915	SMOTHERS	8272	911
CLANCY	2604	918	STRIDE	7281	911
CORNING	1273	915	TALL	3830	917
CURCI	2638	914	TAYLOR	8374	915
EPERT	3900	918	THOMPSON	2343	914
FARLOW	3455	916	WOOD	1734	915
GRECO	2723	914	YOUNG	6263	911
GREEN	2182	911			
GREEN	2156	915			
HERZOG	7440	917			
HIGGINS	5667	917			
JOHNSON	2765	914			
KRAUSE	7222	911			
LARSON	5262	914			
MANHART	8273	914			
MCKAY	3748	917			
MOORE	2358	915			
MURPHY	2099	916			
POWELL	2187	911			
POWELL	3017	918			
REYNOLDS	1727	911			
RICHARDS	7787	916			
SANFORD	3321	917			

15.4 Unusual Reports

Draw a Circle

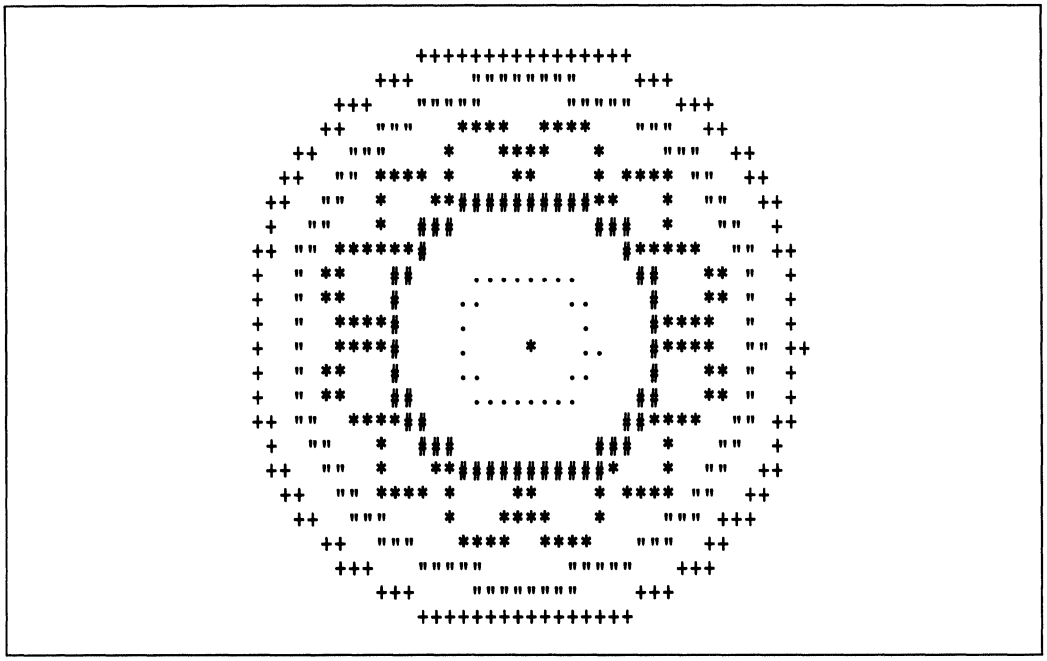
```
DATA _NULL_;
   RADIUS=20;
   FILE PRINT N=PS;
   DO ANGLE=0 TO 2*3.1416 BY .02;
      X=RADIUS*COS(ANGLE)+37;
      Y=26-RADIUS*SIN(ANGLE)*.6;
      PUT #Y @X '*';
   END;
```



Draw a Snowflake

```
DATA _NULL_;
   FILE PRINT N=PS;
   A=5;
   DO ANGLE=0 TO 2*3.1416 BY .02;
      RADI=5;      C='.';  LINK DRAW;
      RADI=10;     C='#';  LINK DRAW;
      RADI=A*SIN(6*ANGLE);
      RADI=RADI-10*(RADI<0)+10*(RADI>0);
      C='*'; LINK DRAW;
      RADI=A+12;       C='"';  LINK DRAW;
      RADI=A+15;       C='+';  LINK DRAW;
   END;
   RETURN;
 DRAW:
   X=RADI*COS(ANGLE)+37;
   Y=26-RADI*SIN(ANGLE)*.6;
   PUT #Y @X C;
   RETURN;
```

Draw a Snowflake



Define a Data Set for the Next Example

Example: Several students were classified into groups according to age and sex. The number of students in each group and the average height and weight for each group were determined and stored on cards.

```
OPTIONS LS=72;
DATA STUDENTS;
   INPUT AGE 1-2 SEX $ 4 N 6-7
         HEIGHT 9-12 WEIGHT 14-18;
   CARDS;
11 F 16 57.1  80.5
12 F 30 59.5  95.8
13 F 16 60.2  94.2
14 F 20 62.4 104.7
15 F 24 62.3 108.7
16 F  3 62.0 116.0
11 M 14 57.0  83.7
12 M 33 58.9  92.7
13 M 30 62.4 102.8
14 M 25 63.9 110.6
15 M 15 66.0 117.8
16 M  7 67.6 129.6
;
PROC PRINT;
```

Define a Data Set for the Next Example

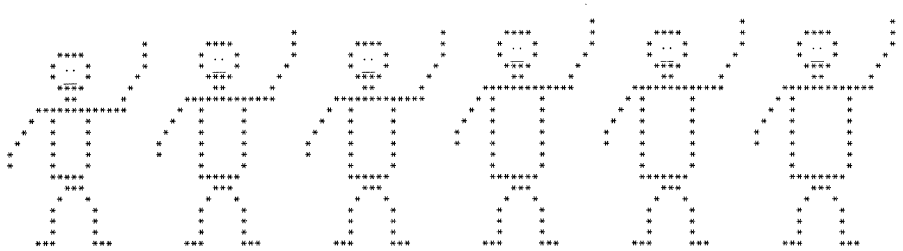
OBS	AGE	SEX	N	HEIGHT	WEIGHT
1	11	F	16	57.1	80.5
2	12	F	30	59.5	95.8
3	13	F	16	60.2	94.2
4	14	F	20	62.4	104.7
5	15	F	24	62.3	108.7
6	16	F	3	62.0	116.0
7	11	M	14	57.0	83.7
8	12	M	33	58.9	92.7
9	13	M	30	62.4	102.8
10	14	M	25	63.9	110.6
11	15	M	15	66.0	117.8
12	16	M	7	67.6	129.6

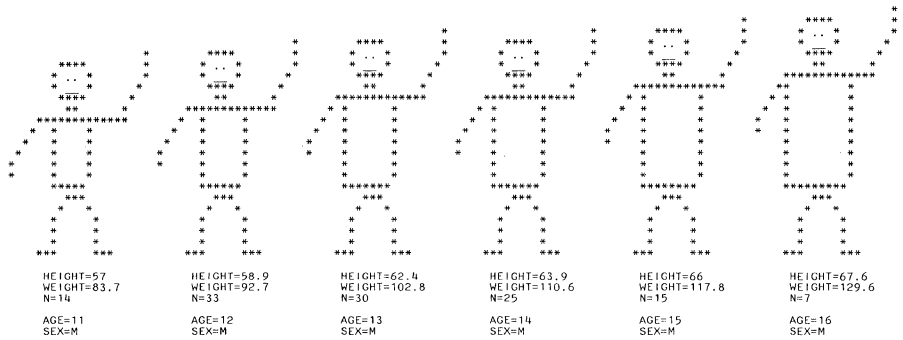
Scaling Figures

Example: Draw a stick person with the body scaled to the average height and weight of the group.

```
OPTIONS LS=132 PS=60;
DATA _NULL_;
   RETAIN C 1;
   FILE PRINT N=PS;
   DO C=1 TO 112 BY 21;
      SET STUDENTS;
      WIDE=(WEIGHT/HEIGHT-1)*8+1; HIGH=(HEIGHT-40)/3;
      OVER=9-WIDE/2; LINE=20-HIGH;
      PUT #LINE / @C '                    *'
                / @C '       ****         *'
                / @C '      * .. *        *'
                / @C '      * __ *       *'
                / @C '       ****       *'
                / @C '         **      *'
                / @C '    ************'
                / @C '   *'
                / @C '  *'
                / @C ' *'
                / @C '*'
                / @C '*';
      PUT #LINE //////;
      DO HIGH=HIGH TO 0 BY -1;
         PUT @C +OVER '*' +WIDE '*';
      END;
      PUT @C +OVER @;
      DO WIDE=WIDE TO 0 BY -1;
         PUT '*' @;
      END;
      PUT / @C '         ***'
          / @C '        *   *'
          / @C '       *     *'
          / @C '       *     *'
          / @C '       *     *'
          / @C '    ***     ***'
          //@C +5  HEIGHT= / @C +5  WEIGHT= / @C +5  N=
          //@C +5  AGE= / @C +5  SEX=;
   END;
   PUT _PAGE_; C=1;
```

Scaling Figures





15.5 Exercises

15.1 Create a SAS data set named PAYROLL from a raw data file with the following record layout. Assume the data are in a disk file that has been assigned fileref=SALARY.

Variable Name	Field Description	Field Location	Variable Type
SSN	social security number	1-9	numeric
NAME	employee's name	10-24	character
SEX	sex	25	character
AGE	age in years	26-27	numeric
START	start date of employment	28-33	MMDDYY6.
SALARY	annual salary	34-38	numeric
DEPT	department number	39-41	numeric

Use PROC PRINT to write a simple report with the specifications shown on the next page.

Exercises

15.1

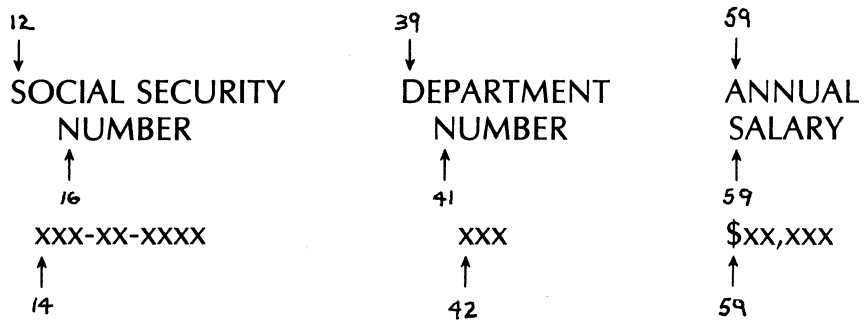
- The column headings from left to right should be:

EMPLOYEE'S NAME	SOCIAL SECURITY NUMBER	ANNUAL SALARY

- Salary data should be printed with dollar signs, commas, and zero decimal places.
- Social security numbers should be written with hyphens in the appropriate places.
- The data should be grouped by department.
- Print the total salary for each department and over all departments.
- Do not print observation numbers.

Exercises

15.2 Use the PAYROLL data set described in the previous problem to write a customized report with the following layout.

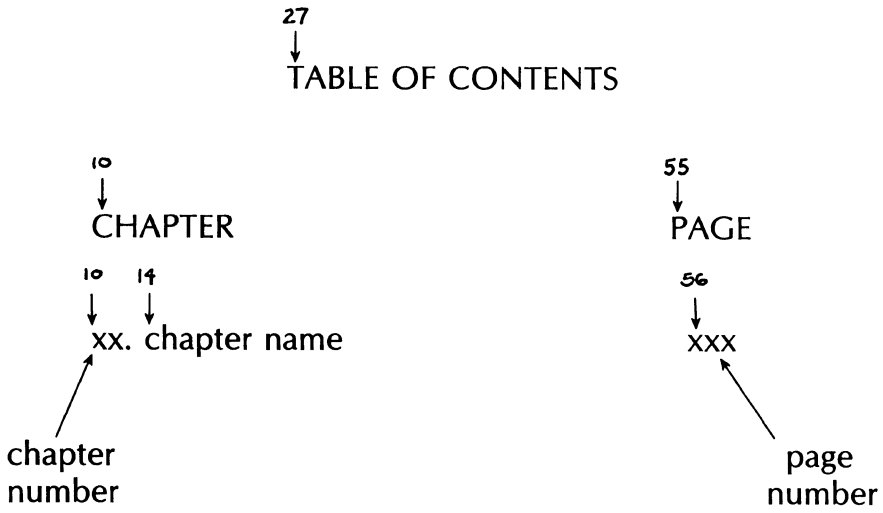


Exercises

15.3 Use PUT and FILE statements to write a table of contents for a book. The information for the table is stored in a file with the following record format. Assume the file has been assigned fileref=TABLE.

Variable Name	Field Description	Field Location	Variable Type
CHAP	chapter number	1-2	numeric
NAME	chapter name	3-25	character
PAGE	page number	26-28	numeric

The layout for the table is shown below. Skip one line between each chapter's information.



Appendix A: Noninteractive Program Execution

A SAS Program

Example: Write a program that will create a SAS data set named PAYROLL and print the data set.

Execute the program in noninteractive mode.

```
OPTIONS TLS=64 LS=64;
DATA PAYROLL;
   INPUT DEPT 1-3 NAME $ 5-12 SEX $ 14
         NETPAY 16-22 GROSSPAY 24-30;
   CARDS;
917 DOBBS    F  169.06  272.29
918 EVANS    M  224.36  310.40
917 HIGGINS  M  777.50 1235.46
914 LARSON   F  415.47  483.92
918 POWELL   F  189.39  271.54
916 RICHARDS M  219.27  352.84
914 RYAN     M  291.56  399.20
;
PROC PRINT;
   TITLE 'LISTING OF PAYROLL DATA SET';
```

Note: this program is referred to as the PAYROLL program in the remainder of this appendix.

TSO Environment

Under TSO, to execute a SAS program in noninteractive mode,

- use an editor under TSO (EDIT, QED, SPF, or some other editor) to store SAS statements in a text data file
- use the INPUT operand to identify the file that contains the SAS statements when you invoke the SAS CLIST.

General form of the command to invoke the SAS CLIST:

SASname [*operands*]

where

SASname is the name used to invoke the CLIST (typically the name is SAS).

operands provide information on how the CLIST is to operate.

TSO Environment

Selected operands:

INPUT('*'OSdatasetname'*'**)**
specifies the name of a text data file (sequential or member of a partitioned data set) that contains the SAS program.

LOG('*'OSdatasetname'*'**)**
specifies the name of a permanent sequential data set to contain the SAS log. This file must already exist.

PRINT('*'OSdatasetname'*'**)**
specifies the name of a permanent sequential data set to contain the output. If the file does not already exist, it will be created. If it does exist, it is allocated with a disposition of MOD.

Note: in all of the above operands, if the first-level qualifier of the data set name is the same as the userid of your current session, the name **does not** have to be enclosed in quotes.

These operands are defined in the CLIST shipped by SAS Institute. They might not be available at sites that choose not to install this CLIST.

TSO Environment

Example: Use an editor to store the PAYROLL program in a text file named EDU.SALARY.PGM. (The program could also be stored in a member of a partitioned data set.)

Use the INPUT operand on the SAS command to execute the stored program in noninteractive mode.

```
READY
sas input('''edu.salary.pgm''')

NOTE: COPYRIGHT (C) 1984 SAS INSTITUTE INC., CARY, N.C.  27511, U.S.A.
NOTE: SAS RELEASE 5.XX AT SAS INSTITUTE DATA CENTER (00000000)


NOTE: DATA SET WORK.PAYROLL HAS 7 OBSERVATIONS AND 5 VARIABLES.
504 OBS/TRK

                  LISTING OF PAYROLL DATA SET

      OBS    DEPT    NAME         SEX    NETPAY    GROSSPAY

       1     917     DOBBS         F     169.06      272.29
       2     918     EVANS         M     224.36      310.40
       3     917     HIGGINS       M     777.50     1235.46
       4     914     LARSON        F     415.47      483.92
       5     918     POWELL        F     189.39      271.54
       6     916     RICHARDS      M     219.27      352.84
       7     914     RYAN          M     291.56      399.20

NOTE: SAS INSTITUTE, SAS CIRCLE, BOX 8000, CARY, N.C. 27511-8000
```

Note: by default, the SAS log and output will be displayed on your screen.

TSO Environment

Example: Repeat the previous example except direct the SAS log to a file named EDU.SALARY.LOG and direct the output to a file named EDU.SALARY.LIST.

```
READY
sas input('''edu.salary.pgm''') +
    log('''edu.salary.log''')   +
    print('''edu.salary.list''')
```

Contents of EDU.SALARY.LOG

```
          SAS(R) LOG   OS SAS 5.XX      MVS/XA TSO USER TSOUSER

NOTE: COPYRIGHT (C) 1984 SAS INSTITUTE INC., CARY, N.C.  27511,
U.S.A.
NOTE: SAS RELEASE 5.XX AT SAS INSTITUTE DATA CENTER (00000000)

   1 OPTIONS TLS=64 LS=64;
   2 DATA PAYROLL;
   3    INPUT DEPT 1-3 NAME $ 5-12 SEX $ 14
   4          NETPAY 16-22 GROSSPAY 24-30;
   5    CARDS;
NOTE: DATA SET WORK.PAYROLL HAS 7 OBSERVATIONS AND 5 VARIABLES.
504 OBS/TRK

  13 ;
  14 PROC PRINT;
  15    TITLE 'LISTING OF PAYROLL DATA SET';

NOTE: SAS INSTITUTE, SAS CIRCLE, BOX 8000, CARY, N.C. 27511-8000
```

TSO Environment

Contents of EDU.SALARY.LIST

LISTING OF PAYROLL DATA SET

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	917	DOBBS	F	169.06	272.29
2	918	EVANS	M	224.36	310.40
3	917	HIGGINS	M	777.50	1235.46
4	914	LARSON	F	415.47	483.92
5	918	POWELL	F	189.39	271.54
6	916	RICHARDS	M	219.27	352.84
7	914	RYAN	M	291.56	399.20

CMS Environment

Under CMS, to execute a SAS program in noninteractive mode,

- use an editor under CMS (EDIT, EDGAR, XEDIT, or some other editor) to store SAS statements in a CMS file with filetype=SAS
- specify the filename of the file that contains the SAS statements when you invoke the SAS EXEC with the SAS command.

General form of the SAS command:

SAS *filename*

where

filename is the filename of the file (with filetype=SAS) that contains the SAS statements.

When this method is used, the SAS System normally creates two disk files that are formatted for a printer.

Both files have the same filename as the file of SAS statements.

One has filetype=SASLOG and the other has filetype=LISTING.

filetype=SASLOG
: contains the SAS log.

filetype=LISTING
: contains output from SAS programs.

CMS Environment

Example: Use an editor to store the PAYROLL program in a CMS file named SALARY SAS A.

Specify SALARY in the SAS command to execute the stored program in noninteractive mode.

```
R;
sas salary
```

- The SAS log is stored in a file named SALARY SASLOG A.
- The procedure output is stored in a file named SALARY LISTING A.

CMS Environment

Use the CMS TYPE command to list the contents of the file SALARY SASLOG A on your terminal.

```
R;
type salary saslog a
```

Contents of SALARY SASLOG A

```
           SAS(R) LOG  CMS SAS 5.XX      VM/CMS CMS USER CMSUSER

NOTE: COPYRIGHT (C) 1984 SAS INSTITUTE INC., CARY, N.C.  27511, U.S.A.
NOTE: CMS SAS RELEASE 5.XX AT SAS INSTITUTE DATA CENTER (00000000).

NOTE: CPUID   VERSION = FF  SERIAL = 000000  MODEL = 4381 .
  1 OPTIONS TLS=64 LS=64;
  2 DATA PAYROLL;
  3    INPUT DEPT 1-3 NAME $ 5-12 SEX $ 14
  4          NETPAY 16-22 GROSSPAY 24-30;
  5    CARDS;
NOTE: DATA SET WORK.PAYROLL HAS 7 OBSERVATIONS AND 5 VARIABLES.

 13 ;
 14 PROC PRINT;
 15    TITLE 'LISTING OF PAYROLL DATA SET';

NOTE: SAS INSTITUTE, SAS CIRCLE, BOX 8000, CARY, N.C. 27511-8000
```

CMS Environment

Use the CMS TYPE command to list the contents of the file SALARY LISTING A on your terminal.

```
R;
type salary listing a
```

Contents of SALARY LISTING A

LISTING OF PAYROLL DATA SET

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	917	DOBBS	F	169.06	272.29
2	918	EVANS	M	224.36	310.40
3	917	HIGGINS	M	777.50	1235.46
4	914	LARSON	F	415.47	483.92
5	918	POWELL	F	189.39	271.54
6	916	RICHARDS	M	219.27	352.84
7	914	RYAN	M	291.56	399.20

ICCF Environment

Under ICCF, to execute a SAS program in noninteractive mode,

- use an editor under ICCF to store SAS statements in an ICCF library member
- specify the filename of the file that contains the SAS statements when you invoke the SAS System.

General form for invoking the SAS System:

SAS *membername*

where

SAS is the name of an ICCF library member that contains an ICCF procedure provided by SAS Institute. (This ICCF procedure is usually in a common ICCF library.)

membername is the name of the ICCF library member that contains the SAS statements.

If you use the procedure supplied by SAS Institute, the SAS System writes

- the SAS log to your terminal
- the output to a member named SASLST.P in your ICCF library.

Note: the ICCF procedure provided by SAS Institute is documented in the *SAS Companion for the VSE Operating System*.

ICCF Environment

Example: Use an editor to store the PAYROLL program in an ICCF library member named SALARY.

Specify SALARY when you invoke the SAS System to execute the stored program in noninteractive mode.

```
sas salary
```

- The SAS log is written to the terminal.
- The output is stored in a library member named SASLST.P.

ICCF Environment

SAS Log

```
NOTE:  SAS/VSE 5.XX BETA RELEASE SYSTEM.
          SAS(R) LOG  VSE SAS 5.XX  VSE 1.3.5 ICCF USER USER1
NOTE: COPYRIGHT (C) 1984 SAS INSTITUTE INC., CARY, N.C.  27511, U.S.A.
NOTE: SAS RELEASE 5.XX AT SAS/VSE 5.XX RELEASE SYSTEM (00000000).

1          OPTIONS TLS=64 LS=64;
2          DATA PAYROLL;
3             INPUT DEPT 1-3 NAME $ 5-12 SEX $ 14
4                   NETPAY 16-22 GROSSPAY 24-30;
5             CARDS;

NOTE: DATA SET WORK.PAYROLL HAS 7 OBSERVATIONS AND 5 VARIABLES.
221 OBS/TRK

13         ;
14         PROC PRINT;
15            TITLE 'LISTING OF PAYROLL DATA SET';

NOTE: THE PROCEDURE PRINT PRINTED PAGE 1.
NOTE: SAS INSTITUTE INC.
      SAS CIRCLE
      PO BOX 8000
      CARY, N.C. 27511-8000
NOTE:  SAS/VSE COMPLETED NORMALLY.
```

ICCF Environment

Use the /LIST command to list the contents of the file SASLST.P on your terminal.

```
/list saslst.p
```

Contents of SASLST.P

LISTING OF PAYROLL DATA SET

OBS	DEPT	NAME	SEX	NETPAY	GROSSPAY
1	917	DOBBS	F	169.06	272.29
2	918	EVANS	M	224.36	310.40
3	917	HIGGINS	M	777.50	1235.46
4	914	LARSON	F	415.47	483.92
5	918	POWELL	F	189.39	271.54
6	916	RICHARDS	M	219.27	352.84
7	914	RYAN	M	291.56	399.20

Appendix B: Solutions to Exercises

Chapter 3 Solutions

3.1 Given the following two raw data records:

```
Column      1234567890
            452 857
            763 943
```

a.
```
DATA; INPUT X Y;
    CARDS;
```

b.
```
DATA; INPUT X 1-3 Y 5-7;
    CARDS;
```

c.
```
DATA; INPUT X 1-2 .1 Y 6-7;
    CARDS;
```

d.
```
DATA; INPUT X $ Y $;
    CARDS;
```

e.
```
DATA; INPUT X $ 1-7 Y $ 2-5;
    CARDS;
```

f.
```
DATA; INPUT X 3. Y 3.;
    CARDS;
```

g.
```
DATA; INPUT X 3.2 @6 Y 3.;
    CARDS;
```

h.
```
DATA; INPUT X/Y;
    CARDS;
```

SAS Data Sets

X	Y
452	857
763	943
452	857
763	943
4.5	57
7.6	43
"452"	"857"
"763"	"943"
"452 857"	"52 8"
"763 943"	"63 9"
452	85
763	94
4.52	57
7.63	43
452	763

Chapter 3 Solutions

3.2 a. list input

```
INPUT STATE $ COUNTY $ SQMILE REGION TRACT
      CODE $ RAINFALL TEMP TEMPTYPE $;
```

3.2 b. column input

```
INPUT STATE $ 1-3 COUNTY $ 5-12 SQMILE 14-19
      REGION 23-25 TRACT 29-31 CODE $ 34-35
      RAINFALL 40-46 TEMP 50-54 TEMPTYPE $ 59;
```

3.2 c. formatted input

```
INPUT STATE $3. @5 COUNTY $8. @14 SQMILE 6.
      @23 REGION 3. @29 TRACT 3. @34 CODE $2.
      @40 RAINFALL 7. @50 TEMP 5.
      @59 TEMPTYPE $1.;
```

3.3

```
DATA AIRLINE;
   INPUT CODE $ REVENUE NUMBER FLIGHT FUEL;
   CARDS;

data lines
```

3.4

```
DATA NUCLEAR;
   INPUT REGION $ 1-20 PLANTS 21-25
         PROXIMIT 26-30 LOCATION $ 31-40
         #2 ACCIDENT 26-30 ENERGY 31-35
         FAVOR $ 36 #3 QUEST1 $ 1 QUEST2 $ 2
         QUEST3 $ 3 QUEST4 $ 4;
   CARDS;

data lines
```

Chapter 4 Solutions

4.1 a.

```
DATA SALES;
   INPUT CLERK $5. DEPT $8. COST 5.;
   CARDS;
SMITHCLOTHING19.28
SMITHTOYS     6.74
JONESTOYS     3.72
ASHLYTOYS    15.83
JONESCLOTHING21.92
SMITHTOYS     6.97
JONESCLOTHING11.58
SMITHCLOTHING12.76
SMITHCLOTHING15.49
ASHLYTOYS     5.45
ASHLYCLOTHING15.90
ASHLYTOYS    17.93
JONESCLOTHING 9.04
SMITHTOYS    19.29
SMITHCLOTHING12.67
ASHLYCLOTHING16.65
;
```

4.1 b.

```
PROC PRINT;
```

4.1 c.

```
PROC PRINT NOOBS;
   VAR CLERK COST;
```

4.1 d.

```
PROC SORT;
   BY CLERK DESCENDING COST;
PROC PRINT;
```

Chapter 4 Solutions

4.1 e.

```
PROC PRINT;
   BY CLERK;
   PAGEBY CLERK;
```

4.1 f.

```
PROC PRINT;
   BY CLERK;
   SUM COST;
```

4.2 a.

```
DATA WATER;
   INPUT ACCOUNT $ 1-8 PAST 9-14
         PRESENT 15-20 MONTH $ 21-23
         YEAR 24-27 COST 28-32;
   CARDS;

data lines
```

4.2 b.

```
PROC SORT;
   BY ACCOUNT;
PROC PRINT NOOBS;
```

4.2 c.

```
PROC PRINT DOUBLE NOOBS;
   VAR ACCOUNT COST;
```

4.2 d.

```
PROC PRINT N NOOBS;
   VAR YEAR MONTH COST;
   SUM COST;
```

Chapter 5 Solutions

5.1 a.

```
DATA NCSC;
   INPUT STATE $ 1-2 MONTH $ 3-5 RAIN 6-10
         MAXTEMP 11-15 MINTEMP 16-20 MEANTEMP 21-25;
   LENGTH PRECIP $ 3 TEMP $ 4;
```

5.1 b.

```
   DIFF=MAXTEMP-MINTEMP;
```

5.1 c.

```
   IF RAIN < 2 THEN PRECIP='DRY';
   ELSE PRECIP='WET';
   IF MEANTEMP > 80 THEN TEMP='HOT';
   ELSE IF 40 < MEANTEMP <= 80 THEN TEMP='MILD';
   ELSE IF MEANTEMP <= 40 THEN TEMP='COLD';
   CARDS;

data lines
```

5.1 d.

```
PROC PRINT;
```

Chapter 5 Solutions

5.2

```
DATA NCSC;
   INPUT STATE $ 1-2 MONTH $ 3-5 RAIN 6-10
         MAXTEMP 11-15 MINTEMP 16-20 MEANTEMP 21-25;
   LENGTH PRECIP $ 3 TEMP $ 4;
   IF RAIN < 2 THEN PRECIP='DRY';
   ELSE PRECIP='WET';
   IF MEANTEMP > 80 THEN TEMP='HOT';
   ELSE IF 40 < MEANTEMP <= 80 THEN TEMP='MILD';
   ELSE IF MEANTEMP <= 40 THEN TEMP='COLD';
   MAXTEMP=5*(MAXTEMP-32)/9;
   MINTEMP=5*(MINTEMP-32)/9;
   MEANTEMP=5*(MEANTEMP-32)/9;
   DIFF=MAXTEMP-MINTEMP;
   CARDS;

data lines

PROC PRINT;
```

Chapter 5 Solutions

5.3

```
DATA JALOPY;
   INPUT DAYS DAYRATE MILES MILERATE DISCRATE INSURE;
   IF DAYS < 1 THEN RENTCOST = DAYRATE;
   ELSE RENTCOST = DAYS*DAYRATE;
   MILECOST=MILES*MILERATE;
   IF INSURE=1 THEN NSURCOST=6*CEIL(DAYS);
   ELSE NSURCOST=0;
   DISCOUNT=DISCRATE*(RENTCOST+MILECOST);
   SUBTOTAL=RENTCOST+MILECOST+NSURCOST-DISCOUNT;
   TAX=.06*SUBTOTAL;
   TOTAL=SUBTOTAL+TAX;
   CARDS;

data lines

PROC PRINT;
```

Chapter 6 Solutions

6.1

Column 1234567

```
 4   0
10  20
 6
```

a.
```
DATA; INPUT X Y;
    Z=X/Y;
    CARDS;
```

b.
```
DATA; INPUT X Y;
    Z=X+Y;
    CARDS;
```

c.
```
DATA; INPUT X Y;
    Z=MEAN(X,Y);
    CARDS;
```

d.
```
DATA; INPUT X Y;
    IF X<Y THEN Z='LOSS';
    CARDS;
```

e.
```
DATA; INPUT X Y;
    Z=X-Y;
    IF Z<0 THEN Z=0;
    CARDS;
```

SAS Data Sets

X	Y	Z
4	0	.
10	20	.5
6	.	.
4	0	4
10	20	30
6	.	.
4	0	2
10	20	15
6	.	6
4	0	
10	20	LOSS
6	.	
4	0	4
10	20	0
6	.	0

Chapter 7 Solutions

7.1 a.

```
DATA SALES;
   INPUT CLERK $ 1-5 DEPT $ 6-13 COST 14-18;
   IF DEPT='TOYS' THEN COM=.075*COST;
   ELSE COM=.05*COST;
   CARDS;
SMITHCLOTHING19.28
SMITHTOYS     6.74
JONESTOYS     3.72
ASHLYTOYS    15.83
JONESCLOTHING21.92
SMITHTOYS     6.97
JONESCLOTHING11.58
SMITHCLOTHING12.76
SMITHCLOTHING15.49
ASHLYTOYS     5.45
ASHLYCLOTHING15.90
ASHLYTOYS    17.93
JONESCLOTHING 9.04
SMITHTOYS    19.29
SMITHCLOTHING12.67
ASHLYCLOTHING16.65
;
```

7.1 b.

```
PROC PRINT;
```

7.1 c.

```
PROC FREQ;
   TABLES CLERK;
```

7.1 d.

```
PROC FREQ;
   TABLES CLERK*DEPT;
```

Chapter 7 Solutions

7.1 e.

```
PROC MEANS MEAN MIN MAX;
   VAR COST;
```

7.1 f.

```
PROC SORT;
   BY CLERK;
PROC MEANS N MEAN SUM;
   VAR COST;
   BY CLERK;
```

7.2 a.

```
DATA WATER;
   INPUT ACCOUNT $ 1-8 PAST 9-14 PRESENT 15-20
         MONTH $ 21-23 YEAR 24-27;
   GALLONS=PRESENT-PAST;
   WATER=.58*GALLONS/100;
   SEWER=.8*WATER;
   TAX=.04*(WATER+SEWER);
   BILL=WATER+SEWER+TAX;
   CARDS;

data lines
```

7.2 b.

```
PROC SORT;
   BY ACCOUNT;
PROC PRINT NOOBS;
```

Chapter 7 Solutions

7.2 c.

```
PROC MEANS MEAN;
   VAR WATER SEWER BILL;
```

7.2 d.

```
PROC SORT;
   BY MONTH;
PROC MEANS MEAN MIN MAX;
   VAR WATER SEWER BILL;
   BY MONTH;
```

7.2 e.

```
PROC SORT;
   BY YEAR ACCOUNT;
PROC MEANS SUM;
   VAR WATER SEWER BILL;
   BY YEAR ACCOUNT;
```

7.3 a.

```
DATA HOUSES;
   INPUT ADDRESS $ 1-25 STYLE $ 26-33
         SQFEET 34-37 BR 38 BATHS 39
         AGE 40-41 FIRE 42 BASE 43
         GARAGE 44 ASSUME 45 PRICE 46-51;
   CARDS;

data lines
```

7.3 b.

```
PROC PRINT;
```

Chapter 7 Solutions

7.3 c.

```
PROC SORT;
   BY PRICE;
PROC PRINT;
```

7.3 d.

```
PROC FREQ;
   TABLES STYLE;
```

7.3 e.

```
PROC FREQ;
   TABLES BR*BATHS;
```

7.3 f.

```
PROC MEANS MEAN MIN MAX;
   VAR PRICE;
```

7.3 g.

```
PROC SORT;
   BY STYLE;
PROC MEANS N MEAN;
   VAR PRICE;
   BY STYLE;
```

Chapter 7 Solutions

7.4 a.

```
DATA INSURE;
   INPUT POLICY $ 1-10 CUSTOMER $ 11-35 AGENT 36-40
         BASEPREM 41-47 PLAN 48 TYPE $ 49-54;
   IF PLAN=1 THEN
      DO;
         YEARPREM=BASEPREM;
         PAYMENT=BASEPREM;
      END;
   ELSE IF PLAN=2 THEN
      DO;
         YEARPREM=1.05*BASEPREM;
         PAYMENT=YEARPREM/4;
      END;
   ELSE IF PLAN=3 THEN
      DO;
         YEARPREM=1.10*BASEPREM;
         PAYMENT=YEARPREM/12;
      END;
   FEE=YEARPREM-BASEPREM;
   CARDS;

data lines
```

7.4 b.

```
PROC MEANS SUM;
   VAR BASEPREM YEARPREM FEE;
```

Chapter 7 Solutions

7.4 c.

```
PROC SORT;
   BY TYPE;
PROC MEANS N SUM;
   VAR YEARPREM;
   BY TYPE;
```

7.4 d.

```
PROC SORT;
   BY AGENT;
PROC MEANS SUM;
   VAR BASEPREM YEARPREM;
   BY AGENT;
```

7.4 e.

```
PROC FREQ;
   TABLES PLAN;
```

7.4 f.

```
PROC FREQ;
   TABLES AGENT*TYPE;
```

Chapter 8 Solutions

8.1 a.

```
DATA HOUSES;
   INPUT ADDRESS $ 1-25 STYLE $ 26-33
         SQFEET 34-37 BR 38 BATHS 39
         AGE 40-41 FIRE 42 BASE 43
         GARAGE 44 ASSUME 45 PRICE 46-51;
   CARDS;

data lines
```

8.1 b.

```
PROC PRINT;
```

8.1 c.

```
DATA UNDER70;
   SET HOUSES;
   IF PRICE<=70000;
PROC PRINT;
```

8.1 d.

```
DATA RANCH SPLIT TWOSTORY;
   SET HOUSES;
   IF STYLE='RANCH' THEN OUTPUT RANCH;
   ELSE IF STYLE='SPLIT' THEN OUTPUT SPLIT;
   ELSE OUTPUT TWOSTORY;
PROC PRINT DATA=RANCH;
   TITLE 'LISTING OF RANCH STYLE HOUSES';
PROC PRINT DATA=SPLIT;
   TITLE 'LISTING OF SPLIT LEVEL HOUSES';
PROC PRINT DATA=TWOSTORY;
   TITLE 'LISTING OF TWO-STORY HOUSES';
```

Chapter 8 Solutions

8.1 e.

```
DATA BUYER1 BUYER2 BUYER3;
   SET HOUSES;
   IF BR=3 & BATHS=2 THEN OUTPUT BUYER1;
   IF BR=4 & BATHS=2 & PRICE<90000 THEN OUTPUT BUYER2;
   IF BR=3 & BATHS=2 & FIRE=1 THEN OUTPUT BUYER3;
PROC PRINT DATA=BUYER1;
   TITLE '3-BEDROOM, 2-BATH HOUSES';
PROC PRINT DATA=BUYER2;
   TITLE '4-BEDROOM, 2-BATH HOUSES';
   TITLE2 'FOR UNDER $90,000';
PROC PRINT DATA=BUYER3;
   TITLE '3-BEDROOM, 2-BATH HOUSES';
   TITLE2 'WITH A FIREPLACE';
```

8.1 f.

```
DATA GARAGE NOGARAGE;
   SET BUYER3;
   KEEP ADDRESS PRICE;
   IF GARAGE=1 THEN OUTPUT GARAGE;
   ELSE OUTPUT NOGARAGE;
PROC PRINT DATA=GARAGE;
   TITLE '3 BEDROOM-2 BATH HOUSES';
   TITLE2 'WITH A FIREPLACE AND GARAGE';
PROC PRINT DATA=NOGARAGE;
   TITLE '3 BEDROOM-2 BATH HOUSES';
   TITLE2 'WITH A FIREPLACE AND NO GARAGE';
```

Chapter 8 Solutions

8.1 g.

```
DATA UNDER75(DROP=FIRE BASE BR BATHS GARAGE)
     OVER75(DROP=BR BATHS AGE BASE STYLE);
   SET BUYER2;
   IF PRICE<75000 THEN OUTPUT UNDER75;
   ELSE OUTPUT OVER75;
PROC PRINT DATA=UNDER75;
   TITLE '4 BEDROOM-2 BATH HOUSES';
   TITLE2 'FOR UNDER $75,000';
PROC PRINT DATA=OVER75;
   TITLE '4 BEDROOM-2 BATH HOUSES';
   TITLE2 'FOR $75,000 TO $90,000';
```

8.2 a.

```
DATA DIRECT;
   INPUT NUMBER 1-7 EXCHANGE 1-3 NAME $ 8-35
         ADDRESS $ 36-55 ZIP 56-60;
   CARDS;

data lines

PROC PRINT;
```

8.2 b.

```
DATA WRONG;
   SET DIRECT;
   IF ZIP = 28007 & EXCHANGE ¬= 862 OR
      ZIP = 28008 & EXCHANGE = 862;
PROC PRINT;
```

Chapter 8 Solutions

8.2 c.

```
DATA EX862 EX863 EX864;
   SET DIRECT;
   IF EXCHANGE = 862 THEN OUTPUT EX862;
   ELSE IF EXCHANGE = 863 THEN OUTPUT EX863;
   ELSE IF EXCHANGE = 864 THEN OUTPUT EX864;
PROC PRINT DATA=EX862;
   TITLE 'CUSTOMERS WITH THE 862 EXCHANGE';
PROC PRINT DATA=EX863;
   TITLE 'CUSTOMERS WITH THE 863 EXCHANGE';
PROC PRINT DATA=EX864;
   TITLE 'CUSTOMERS WITH THE 864 EXCHANGE';
```

8.3 a.

```
DATA EMPLOYEE;
   INPUT SSN 1-9 START $ 10-15 SALARY 16-20
         DEPT 21-23;
   CARDS;

data lines
```

8.3 b.

```
DATA DEPT121;
   SET EMPLOYEE(DROP=START);
   IF DEPT=121;
```

Chapter 8 Solutions

8.3 c.

```
DATA DEPT121 DEPT131 DEPT141;
   SET EMPLOYEE;
   IF DEPT=121 THEN OUTPUT DEPT121;
   ELSE IF DEPT=131 THEN OUTPUT DEPT131;
   ELSE IF DEPT=141 THEN OUTPUT DEPT141;
```

8.3 d.

```
DATA ALL
     DEPT121(DROP=DEPT)
     DEPT131(DROP=DEPT)
     DEPT141(DROP=DEPT);
   INPUT SSN 1-9 SALARY 16-20 DEPT 21-23;
   OUTPUT ALL;
   IF DEPT=121 THEN OUTPUT DEPT121;
   ELSE IF DEPT=131 THEN OUTPUT DEPT131;
   ELSE IF DEPT=141 THEN OUTPUT DEPT141;
   CARDS;

data lines
```

Chapter 9 Solutions

9.1 a.

```
DATA PAYROLL;
   INPUT DEPT 1-3 NAME $ 5-12 NUMBER 14-18
         SEX $ 20 NETPAY 22-28 GROSSPAY 30-36;
   CARDS;
914 LARSON   11357 F  265.47  383.92
914 RYAN     10961 M  291.56  399.20
914 JOHNSON  10546 M  435.23  618.72
914 THOMPSON 11584 M  221.09  297.56
915 WOOD      8113 F  238.17  372.24
915 TAYLOR    2943 F  287.22  401.72
915 MOORE    12649 M  557.18  728.12
915 HELMS     3658 M  479.26  701.26
916 SHIRES   10052 F  487.12  729.22
916 RICHARDS  5781 M  319.27  472.84
916 SMITH     6237 F  207.09  345.88
916 MURPHY    6927 M  272.66  385.11
916 FAIRLEY   8846 M  521.66  834.80
917 HERZOG    6433 M  692.57 1045.26
917 TALL     11931 M  355.19  492.26
917 HIGGINS  11658 M  777.50 1235.46
917 DOOB     10554 F  669.06  972.29
918 EPERT     7716 M  224.36  310.40
918 CLANCY    6648 M  245.37  347.12
918 POWELL    8123 F  789.39 1271.54
;
```

9.1 b.

```
PROC PRINT;
```

9.1 c.

```
PROC PRINT;
   FORMAT NETPAY GROSSPAY DOLLAR9.2;
```

Chapter 9 Solutions

9.1 d.

```
PROC FORMAT;
   VALUE $SEXFMT 'F'='FEMALE' 'M'='MALE';
   VALUE DEPTFMT 914='MARKETING'
                 915='MAINTENANCE'
                 916='RESEARCH & DEVELOPMENT'
                 917='ADMINISTRATION'
                 918='EDUCATION';
PROC PRINT;
   FORMAT SEX $SEXFMT. DEPT DEPTFMT.
          NETPAY GROSSPAY DOLLAR9.2;
```

9.1 e.

```
PROC MEANS MEAN;
   VAR NETPAY;
   LABEL NETPAY='NET PAY OF EMPLOYEE';
```

9.1 f.

```
PROC SORT;
   BY DEPT;
PROC MEANS MEAN;
   VAR NETPAY;
   BY DEPT;
   LABEL NETPAY='NET PAY OF EMPLOYEE'
           DEPT='DEPARTMENT';
   FORMAT DEPT DEPTFMT.;
```

Chapter 9 Solutions

9.2 a.

```
DATA HOUSES;
   INPUT ADDRESS $ 1-25 STYLE $ 26-33
         SQFEET 34-37 BR 38 BATHS 39
         AGE 40-41 FIRE 42 BASE 43
         GARAGE 44 ASSUME 45 PRICE 46-51;
   CARDS;

data lines
```

9.2 b.

```
PROC PRINT;
   FORMAT PRICE DOLLAR8. BR BATHS WORDS.;
```

9.2 c.

```
PROC SORT;
   BY STYLE;
PROC MEANS MEAN;
   BY STYLE;
   VAR PRICE;
   LABEL STYLE='STYLE OF THE HOUSE';
```

9.2 d.

```
PROC FORMAT;
   VALUE YESNO 0='NO' 1='YES';
   PICTURE SQFT LOW-HIGH='0000 SQ. FT.';
   PICTURE AGEFMT LOW-HIGH='00 YEARS OLD';
PROC PRINT;
   FORMAT PRICE DOLLAR8. BR BATHS WORDS.
          FIRE BASE GARAGE ASSUME YESNO.
          SQFEET SQFT. AGE AGEFMT.;
```

Chapter 10 Solutions

10.1

```
DATA ORDERS;
   INPUT DATEOUT YYMMDD8. @11 DATEIN YYMMDD6.
         MILES 21-24;
   RATIO=MILES/((DATEIN-DATEOUT)+1);
   CARDS;

data lines

PROC PRINT;
   FORMAT DATEOUT DATEIN WEEKDATE30.;
```

10.2

```
DATA BIRTHDAY;
   INPUT NAME $ 1-20 @24 BORN DATE7.;
   IF MONTH(TODAY())=12 THEN NEXTMON=1;
   ELSE NEXTMON=MONTH(TODAY())+1;
   IF MONTH(BORN)=NEXTMON;
   CARDS;

data lines

PROC PRINT;
   FORMAT BORN WORDDATE20.;
```

Chapter 10 Solutions

10.2 alternate solution

```
DATA BIRTHDAY;
   INPUT NAME $ 1-20 @24 BORN DATE7.;
   IF MONTH(BORN)=MOD(MONTH(TODAY()),12)+1;
   CARDS;

data lines

PROC PRINT;
   FORMAT BORN WORDDATE20.;
```

10.3

```
DATA MATURE;
   INPUT ID 1-10 NAME $ 11-30 AMOUNT 31-38
         RATE 39-43 ISSUE MMDDYY8.
         EXPIRE MMDDYY8.;
   IF EXPIRE=TODAY();
   INTEREST=(EXPIRE-ISSUE)*AMOUNT*RATE/365;
   CARDS;

data lines

PROC PRINT;
   FORMAT ISSUE EXPIRE MMDDYY8.
          AMOUNT INTEREST DOLLAR10.2;
```

Chapter 11 Solutions

11.1 a.

```
DATA CHICAGO;
   INPUT MONTH 1-2 SEASON 3 CITY $ 4-10
         LOW 11-13 HIGH 14-16;
   CARDS;
 14CHICAGO 17 32
 24CHICAGO 20 35
 31CHICAGO 29 45
 41CHICAGO 40 59
 51CHICAGO 50 70
 62CHICAGO 60 81
 72CHICAGO 65 84
 82CHICAGO 64 83
 93CHICAGO 56 76
103CHICAGO 46 65
113CHICAGO 33 48
124CHICAGO 22 35
;
```

11.1 b.

```
PROC PLOT;
   PLOT LOW*MONTH;
```

11.1 c.

```
PROC PLOT;
   PLOT HIGH*MONTH;
```

11.1 d.

```
PROC PLOT;
   PLOT LOW*MONTH='-' HIGH*MONTH='+'/OVERLAY;
```

Chapter 11 Solutions

11.2 a.

```
DATA WEATHER;
   INPUT MONTH 1-2 SEASON 3 CITY $ 4-10
         LOW 11-13 HIGH 14-16;
   RANGE=HIGH-LOW;
   CARDS;
 14CHICAGO 17 32
 24CHICAGO 20 35
 31CHICAGO 29 45
 41CHICAGO 40 59
 51CHICAGO 50 70
 62CHICAGO 60 81
 72CHICAGO 65 84
 82CHICAGO 64 83
 93CHICAGO 56 76
103CHICAGO 46 65
113CHICAGO 33 48
124CHICAGO 22 35
 14PHOENIX 38 65
 24PHOENIX 41 69
 31PHOENIX 45 75
 41PHOENIX 52 84
 51PHOENIX 60 93
 62PHOENIX 68102
 72PHOENIX 78105
 82PHOENIX 76102
 93PHOENIX 69 98
103PHOENIX 57 88
113PHOENIX 45 75
124PHOENIX 39 66
;
```

11.2 b.

```
PROC CHART;
   VBAR CITY/SUMVAR=LOW TYPE=MEAN;
```

Chapter 11 Solutions

11.2 c.

```
PROC CHART;
   BLOCK SEASON/SUMVAR=HIGH TYPE=MEAN DISCRETE
      GROUP=CITY;
```

11.2 d.

```
PROC SORT;
   BY CITY;
PROC CHART;
   PIE SEASON/SUMVAR=HIGH TYPE=MEAN DISCRETE;
   BY CITY;
```

11.2 e.

```
PROC CHART;
   HBAR MONTH/SUMVAR=RANGE DISCRETE
      GROUP=CITY;
```

11.3 a.

```
DATA TV;
   INPUT AGEGROUP $ 1-5 SEX $ 6 TIME 7-10;
   CARDS;
ADULTF37.4
ADULTM28.6
TEEN F25.2
TEEN M22.1
CHILDF26.9
CHILDM26.2
;
PROC CHART;
   HBAR AGEGROUP/GROUP=SEX SUMVAR=TIME
      MIDPOINTS='CHILD' 'TEEN' 'ADULT';
   LABEL AGEGROUP='AGE CATEGORY';
```

Chapter 11 Solutions

11.3 b.

```
DATA TV;
   INPUT AGEGROUP $ 1-5 SEX $ 6 TIME 7-10;
   CARDS;

data lines

PROC FORMAT;
   VALUE $AGEFMT 'ADULT'='OVER 19'
                 'TEEN'='13 TO 19'
                 'CHILD'='UNDER 13';
   VALUE $SEXFMT 'F'='FEMALE' 'M'='MALE';
PROC CHART;
   HBAR AGEGROUP/GROUP=SEX SUMVAR=TIME
      MIDPOINTS='UNDER 13' '13 TO 19' 'OVER 19';
   LABEL AGEGROUP='AGE CATEGORY';
   FORMAT AGEGROUP $AGEFMT. SEX $SEXFMT.;
```

Chapter 11 Solutions

11.4 a.

```
DATA MILEAGE;
   INPUT YEAR 1-4 MPG 5-9;
   CARDS;
196813.79
196913.63
197013.57
197113.57
197213.54
197313.10
197413.43
197513.53
197613.72
197713.94
197814.06
;
PROC PLOT;
   PLOT MPG*YEAR='*'/HAXIS=1968 TO 1978 BY 1;
   LABEL MPG='AVERAGE MILES PER GALLON';
   TITLE 'TREND OF CAR EFFICIENCY';
   TITLE2 'FROM 1968 TO 1978';
```

11.4 b.

```
PROC CHART;
   VBAR YEAR/DISCRETE SUMVAR=MPG;
```

Chapter 12 Solutions

12.1

SAS Data Set A

ID	X	Y
1	12	11
2	15	.

SAS Data Set B

ID	X	Z
1	.	4
3	17	6
3	18	.

a.
```
DATA;
    SET A B;
```

b.
```
DATA;
    SET A B;
    BY ID;
```

c.
```
DATA;
    MERGE A B;
```

d.
```
DATA;
    MERGE A B;
    BY ID;
```

e.
```
DATA;
    UPDATE A B;
    BY ID;
```

SAS Data Sets

ID	X	Y	Z
1	12	11	.
2	15	.	.
1	.	.	4
3	17	.	6
3	18	.	.
1	12	11	.
1	.	.	4
2	15	.	.
3	17	.	6
3	18	.	.
1	.	11	4
3	17	.	6
3	18	.	.
1	.	11	4
2	15	.	.
3	17	.	6
3	18	.	.
1	12	11	4
2	15	.	.
3	18	.	6

Chapter 12 Solutions

12.2 a.

```
DATA MASTER;
   INPUT NAME $ 1-25 HOURS 26-28 CREDITS 29-31;
   CARDS;

data lines

DATA CURRENT;
   INPUT NAME $ 1-25 SHOURS 26-28 SCREDITS 29-31;
   CARDS;

data lines
```

12.2 b.

```
PROC SORT DATA=MASTER;
   BY NAME;
PROC SORT DATA=CURRENT;
   BY NAME;
DATA ALL;
   MERGE MASTER CURRENT;
   DROP HOURS CREDITS SHOURS SCREDITS;
   BY NAME;
   THOURS=HOURS+SHOURS;
   TCREDITS=CREDITS+SCREDITS;
   OLDGPA=CREDITS/HOURS;
   SEMGPA=SCREDITS/SHOURS;
   NEWGPA=TCREDITS/THOURS;
   IF THOURS < 30 THEN CLASS='FR';
   ELSE IF 30 <= THOURS < 60 THEN CLASS='SO';
   ELSE IF 60 <= THOURS < 90 THEN CLASS='JR';
   ELSE IF 90 <= THOURS < 120 THEN CLASS='SR';
   ELSE CLASS='GR';
```

Chapter 12 Solutions

12.2 c.

```
PROC PRINT;
```

12.2 d.

```
PROC SORT;
   BY CLASS;
PROC MEANS MEAN;
   VAR SEMGPA NEWGPA;
   BY CLASS;
```

12.3 a.

```
DATA SALES85;
   INPUT YEAR 1-4 MONTH 6-7 SALES 9-13;
   CARDS;
1985  1 12088
1985  2 10928
1985  3  9384
1985  4  9277
1985  5 10542
1985  6 11342
1985  7 10993
1985  8 11633
1985  9 13783
1985 10 13732
1985 11 13926
1985 12 14579
;
PROC PLOT;
   PLOT SALES*MONTH/HAXIS=1 TO 12;
```

Chapter 12 Solutions

12.3 b.

```
DATA SALES86;
   INPUT YEAR 1-2 MONTH 4-5 SALES 7-11;
   YEAR=YEAR+1900;
   CARDS;
86  1 13882
86  2 13801
86  3 12917
86  4 13277
86  5 14831
86  6 14979
86  7 15003
86  8 15710
86  9 16583
86 10 18435
86 11 17553
86 12 18367
;
PROC PLOT;
   PLOT SALES*MONTH/HAXIS=1 TO 12;
```

12.3 c.

```
DATA BOTH;
   MERGE SALES85(RENAME=(SALES=SALES85))
         SALES86(RENAME=(SALES=SALES86));
   BY MONTH;
   DROP YEAR;
PROC PLOT;
   PLOT SALES85*MONTH='0' SALES86*MONTH='1'/OVERLAY;
```

Chapter 12 Solutions

12.3 d.

```
DATA COMBINE;
   SET SALES85 SALES86;
   BY MONTH;
PROC MEANS MEAN;
   VAR SALES;
   BY MONTH;
```

12.4 a.

```
DATA PATIENT;
   INPUT SSN 1-9 NAME $ 10-40 SEX $ 41
         BORN MMDDYY8.;
   CARDS;

data lines

DATA VISITS;
   INPUT SSN 1-9 DATE MMDDYY8. FEE 18-20;
   CARDS;

data lines

PROC SORT DATA=PATIENT;
   BY SSN;
PROC SORT DATA=VISITS;
   BY SSN;
DATA BOTH;
   MERGE PATIENT VISITS;
   BY SSN;
   IF FIRST.SSN THEN VISIT=0;
   VISIT+1;
PROC PRINT;
   FORMAT SSN SSN11. BORN DATE MMDDYY8.;
```

Chapter 12 Solutions

12.4 b.

```
PROC SORT DATA=PATIENT;
   BY SSN;
PROC SORT DATA=VISITS;
   BY SSN;
DATA BILLS;
   MERGE PATIENT VISITS(IN=V);
   BY SSN;
   IF V;
   IF FIRST.SSN THEN VISIT=0;
   VISIT+1;
PROC PRINT;
   FORMAT SSN SSN11. BORN DATE MMDDYY8.;
```

12.4 c.

```
DATA INVOICE;
   SET BILLS;
   BY SSN;
   IF FIRST.SSN THEN TOTAL=0;
   TOTAL+FEE;
   IF LAST.SSN;
   KEEP SSN NAME TOTAL;
PROC PRINT;
   FORMAT SSN SSN11. TOTAL DOLLAR10.2;
```

Chapter 13 Solutions

13.1 a.

```
DATA SAFETY;
   INFILE ACCIDENT;
   INPUT YEAR 1-4 FALLS 5-9 BURNS 10-14
         INGEST 15-19 MECH 20-24 SOLID 25-29
         GAS 30-34 GUNS 35-39 OTHER 40-44;
   TOTAL=SUM(FALLS,BURNS,INGEST,MECH,SOLID,
             GAS,GUNS,OTHER);
```

13.1 b.

```
PROC PRINT;
```

13.1 c.

```
PROC MEANS MEAN DATA=SAFETY(DROP=YEAR);
```

13.2 a.

```
DATA COUPLES;
   INFILE STATS;
   INPUT YEAR 1-4 STATE $ 5-19 MARRIAGE 20-25
         DIVORCE 26-31;
```

13.2 b.

```
PROC SORT;
   BY YEAR;
PROC MEANS SUM;
   BY YEAR;
   VAR MARRIAGE DIVORCE;
```

Chapter 13 Solutions

13.2 c.

```
PROC SORT;
   BY STATE;
PROC MEANS MEAN;
   BY STATE;
   VAR MARRIAGE DIVORCE;
```

13.3

```
DATA _NULL_;
   SET SAFETY;
   FILE NEWDATA;
   PUT YEAR 1-4 TOTAL 5-9 FALLS 10-14
       BURNS 15-19 INGEST 20-24 MECH 25-29
       SOLID 30-34 GAS 35-39 GUNS 40-44
       OTHER 45-49;
```

The PUT statement above can also be written as follows:

```
PUT YEAR 1-4 (TOTAL FALLS BURNS INGEST
    MECH SOLID GAS GUNS OTHER) (5.);
```

Chapter 13 Solutions

13.4

```
DATA _NULL_;
   INPUT REGION $ 1-20 PLANTS 21-25
         PROXIMIT 26-30 LOCATION $ 31-40
         #2 ACCIDENT 26-30 ENERGY 31-35
         FAVOR $ 36 #3 QUEST1 $ 1 QUEST2 $ 2
         QUEST3 $ 3 QUEST4 $ 4;
   FILE DISKDATA;
   PUT REGION $ 1-20 PLANTS 21-25
       PROXIMIT 26-30 LOCATION $ 31-40
       ACCIDENT 41-45 ENERGY 46-50 FAVOR $ 51
       QUEST1 $ 52 QUEST2 $ 53 QUEST3 $ 54
       QUEST4 $ 55;
   CARDS;

data lines
```

Chapter 14 Solutions

14.1 a.

```
DATA HOME.ACCIDNTS;
   INFILE ACCIDENT;
   INPUT YEAR 1-4 FALLS 5-9 BURNS 10-14
         INGEST 15-19 MECH 20-24 SOLID 25-29
         GAS 30-34 GUNS 35-39 OTHER 40-44;
   TOTAL=SUM(FALLS,BURNS,INGEST,MECH,SOLID,
             GAS,GUNS,OTHER);
```

14.1 b.

```
PROC CONTENTS DATA=HOME.ACCIDNTS;
```

14.2 a. All environments except CMS

```
PROC PRINT DATA=ACCID.ACCIDNTS;
```

CMS

```
PROC PRINT DATA=HOME.ACCIDNTS;
```

14.2 b. All environments except CMS

```
PROC MEANS MEAN DATA=ACCID.ACCIDNTS(DROP=YEAR);
```

CMS

```
PROC MEANS MEAN DATA=HOME.ACCIDNTS(DROP=YEAR);
```

Chapter 14 Solutions

14.3 a. All environments except CMS

```
DATA MOREDATA.SPOUSES;
   INFILE COUPLES;
   INPUT YEAR 1-4 STATE $ 5-19
         MARRIAGE 20-25 DIVORCE 26-31;
```

CMS

```
DATA HOME.SPOUSES;
   INFILE COUPLES;
   INPUT YEAR 1-4 STATE $ 5-19
         MARRIAGE 20-25 DIVORCE 26-31;
```

14.3 b. All environments except CMS

```
PROC SORT DATA=MOREDATA.SPOUSES OUT=TEMP;
   BY YEAR;
PROC MEANS SUM DATA=TEMP;
   VAR MARRIAGE DIVORCE;
   BY YEAR;
```

CMS

```
PROC SORT DATA=HOME.SPOUSES OUT=TEMP;
   BY YEAR;
PROC MEANS SUM DATA=TEMP;
   VAR MARRIAGE DIVORCE;
   BY YEAR;
```

Chapter 14 Solutions

14.3 c. All environments

```
PROC SORT DATA=TEMP;
   BY STATE;
PROC MEANS MEAN DATA=TEMP;
   VAR MARRIAGE DIVORCE;
   BY STATE;
```

14.4

```
PROC SORT DATA=HOME.ACCIDNTS OUT=HOME.HOMACCID;
   BY TOTAL;
```

14.5 a.

```
DATA HOME.BIRTHS;
   INFILE BORN;
   INPUT YEAR 1-4 STATE $ 5-6 MALES 7-13
         FEMALES 14-20;
   TOTAL=MALES+FEMALES;
```

14.5 b.

```
PROC PLOT DATA=HOME.BIRTHS;
   PLOT MALES*YEAR='M' FEMALES*YEAR='F'/
      OVERLAY;
```

Chapter 15 Solutions

15.1

```
DATA PAYROLL;
   INFILE SALARY;
   INPUT SSN 1-9 NAME $ 10-24 SEX $ 25 AGE 26-27
         START MMDDYY6. SALARY 34-38 DEPT 39-41;
PROC SORT;
   BY DEPT;
PROC PRINT SPLIT='+' NOOBS;
   BY DEPT;
   SUM SALARY;
   VAR NAME SSN SALARY;
   LABEL NAME='EMPLOYEE''S+   NAME'
         SSN='SOCIAL SECURITY+    NUMBER'
      SALARY='ANNUAL+SALARY';
   FORMAT SALARY DOLLAR8. SSN SSN11.;
```

15.2

```
DATA _NULL_;
   SET PAYROLL;
   FILE PRINT HEADER=HEAD NOTITLES;
   PUT @14 SSN SSN11. @42 DEPT 3. @59 SALARY DOLLAR7.;
   RETURN;
 HEAD:
   PUT @12 'SOCIAL SECURITY' @39 'DEPARTMENT'
       @59 'ANNUAL' / @16 'NUMBER'
       @41 'NUMBER' @59 'SALARY' /;
   RETURN;
```

Chapter 15 Solutions

15.3

```
DATA _NULL_;
   INFILE TABLE;
   INPUT CHAP 1-2 NAME $ 3-25 PAGE 26-28;
   FILE PRINT NOTITLES HEADER=H;
   PUT @10 CHAP 2. '.' @14 NAME $23.
       @56 PAGE 3. /;
   RETURN;
 H:
   PUT / @27 'TABLE OF CONTENTS' ///
       @10 'CHAPTER' @55 'PAGE' //;
   RETURN;
```

Appendix C: Output from Selected Exercises

Chapter 4 Output

Note: all the output in this appendix was generated with the linesize set to 72. The following OPTIONS statement was used to define the linesize:

```
OPTIONS LS=72;
```

4.1 b.

OBS	CLERK	DEPT	COST
1	SMITH	CLOTHING	19.28
2	SMITH	TOYS	6.74
3	JONES	TOYS	3.72
4	ASHLY	TOYS	15.83
5	JONES	CLOTHING	21.92
6	SMITH	TOYS	6.97
7	JONES	CLOTHING	11.58
8	SMITH	CLOTHING	12.76
9	SMITH	CLOTHING	15.49
10	ASHLY	TOYS	5.45
11	ASHLY	CLOTHING	15.90
12	ASHLY	TOYS	17.93
13	JONES	CLOTHING	9.04
14	SMITH	TOYS	19.29
15	SMITH	CLOTHING	12.67
16	ASHLY	CLOTHING	16.65

Chapter 4 Output

4.1 c.

CLERK	COST
SMITH	19.28
SMITH	6.74
JONES	3.72
ASHLY	15.83
JONES	21.92
SMITH	6.97
JONES	11.58
SMITH	12.76
SMITH	15.49
ASHLY	5.45
ASHLY	15.90
ASHLY	17.93
JONES	9.04
SMITH	19.29
SMITH	12.67
ASHLY	16.65

Chapter 4 Output

4.1 d.

OBS	CLERK	DEPT	COST
1	ASHLY	TOYS	17.93
2	ASHLY	CLOTHING	16.65
3	ASHLY	CLOTHING	15.90
4	ASHLY	TOYS	15.83
5	ASHLY	TOYS	5.45
6	JONES	CLOTHING	21.92
7	JONES	CLOTHING	11.58
8	JONES	CLOTHING	9.04
9	JONES	TOYS	3.72
10	SMITH	TOYS	19.29
11	SMITH	CLOTHING	19.28
12	SMITH	CLOTHING	15.49
13	SMITH	CLOTHING	12.76
14	SMITH	CLOTHING	12.67
15	SMITH	TOYS	6.97
16	SMITH	TOYS	6.74

Chapter 4 Output

4.1 e.

Page 1

```
------------------------------CLERK=ASHLY-------------------------------

                          OBS    DEPT          COST

                           1     TOYS          17.93
                           2     CLOTHING      16.65
                           3     CLOTHING      15.90
                           4     TOYS          15.83
                           5     TOYS           5.45
```

Page 2

```
------------------------------CLERK=JONES-------------------------------

                          OBS    DEPT          COST

                           6     CLOTHING      21.92
                           7     CLOTHING      11.58
                           8     CLOTHING       9.04
                           9     TOYS           3.72
```

Page 3

```
------------------------------CLERK=SMITH-------------------------------

                          OBS    DEPT          COST

                          10     TOYS          19.29
                          11     CLOTHING      19.28
                          12     CLOTHING      15.49
                          13     CLOTHING      12.76
                          14     CLOTHING      12.67
                          15     TOYS           6.97
                          16     TOYS           6.74
```

Chapter 4 Output

4.1 f.

```
------------------------------CLERK=ASHLY-------------------------------

                          OBS    DEPT         COST

                            1    TOYS         17.93
                            2    CLOTHING     16.65
                            3    CLOTHING     15.90
                            4    TOYS         15.83
                            5    TOYS          5.45
                         -----                -----
                         CLERK                71.76


------------------------------CLERK=JONES-------------------------------

                          OBS    DEPT         COST

                            6    CLOTHING     21.92
                            7    CLOTHING     11.58
                            8    CLOTHING      9.04
                            9    TOYS          3.72
                         -----                -----
                         CLERK                46.26


------------------------------CLERK=SMITH-------------------------------

                          OBS    DEPT          COST

                           10    TOYS         19.29
                           11    CLOTHING     19.28
                           12    CLOTHING     15.49
                           13    CLOTHING     12.76
                           14    CLOTHING     12.67
                           15    TOYS          6.97
                           16    TOYS          6.74
                        -----                ------
                        CLERK                 93.20
                                             ======
                                            211.22
```

Chapter 7 Output

7.1 b.

OBS	CLERK	DEPT	COST	COM
1	SMITH	CLOTHING	19.28	0.96400
2	SMITH	TOYS	6.74	0.50550
3	JONES	TOYS	3.72	0.27900
4	ASHLY	TOYS	15.83	1.18725
5	JONES	CLOTHING	21.92	1.09600
6	SMITH	TOYS	6.97	0.52275
7	JONES	CLOTHING	11.58	0.57900
8	SMITH	CLOTHING	12.76	0.63800
9	SMITH	CLOTHING	15.49	0.77450
10	ASHLY	TOYS	5.45	0.40875
11	ASHLY	CLOTHING	15.90	0.79500
12	ASHLY	TOYS	17.93	1.34475
13	JONES	CLOTHING	9.04	0.45200
14	SMITH	TOYS	19.29	1.44675
15	SMITH	CLOTHING	12.67	0.63350
16	ASHLY	CLOTHING	16.65	0.83250

7.1 c.

CLERK	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
ASHLY	5	31.3	5	31.3
JONES	4	25.0	9	56.3
SMITH	7	43.8	16	100.0

Chapter 7 Output

7.1 d.

TABLE OF CLERK BY DEPT

CLERK / DEPT	CLOTHING	TOYS	TOTAL
ASHLY	2	3	5
FREQUENCY			
PERCENT	12.50	18.75	31.25
ROW PCT	40.00	60.00	
COL PCT	22.22	42.86	
JONES	3	1	4
FREQUENCY			
PERCENT	18.75	6.25	25.00
ROW PCT	75.00	25.00	
COL PCT	33.33	14.29	
SMITH	4	3	7
FREQUENCY			
PERCENT	25.00	18.75	43.75
ROW PCT	57.14	42.86	
COL PCT	44.44	42.86	
TOTAL	9	7	16
	56.25	43.75	100.00

7.1 e.

VARIABLE	MEAN	MINIMUM VALUE	MAXIMUM VALUE
COST	13.20125000	3.72000000	21.92000000

Chapter 7 Output

7.1 f.

VARIABLE	N	MEAN	SUM
---------- CLERK=ASHLY ----------			
COST	5	14.35200000	71.76000000
---------- CLERK=JONES ----------			
COST	4	11.56500000	46.26000000
---------- CLERK=SMITH ----------			
COST	7	13.31428571	93.20000000

Chapter 9 Output

9.1 b.

OBS	DEPT	NAME	NUMBER	SEX	NETPAY	GROSSPAY
1	914	LARSON	11357	F	265.47	383.92
2	914	RYAN	10961	M	291.56	399.20
3	914	JOHNSON	10546	M	435.23	618.72
4	914	THOMPSON	11584	M	221.09	297.56
5	915	WOOD	8113	F	238.17	372.24
6	915	TAYLOR	2943	F	287.22	401.72
7	915	MOORE	12649	M	557.18	728.12
8	915	HELMS	3658	M	479.26	701.26
9	916	SHIRES	10052	F	487.12	729.22
10	916	RICHARDS	5781	M	319.27	472.84
11	916	SMITH	6237	F	207.09	345.88
12	916	MURPHY	6927	M	272.66	385.11
13	916	FAIRLEY	8846	M	521.66	834.80
14	917	HERZOG	6433	M	692.57	1045.26
15	917	TALL	11931	M	355.19	492.26
16	917	HIGGINS	11658	M	777.50	1235.46
17	917	DOOB	10554	F	669.06	972.29
18	918	EPERT	7716	M	224.36	310.40
19	918	CLANCY	6648	M	245.37	347.12
20	918	POWELL	8123	F	789.39	1271.54

Chapter 9 Output

9.1 c.

OBS	DEPT	NAME	NUMBER	SEX	NETPAY	GROSSPAY
1	914	LARSON	11357	F	$265.47	$383.92
2	914	RYAN	10961	M	$291.56	$399.20
3	914	JOHNSON	10546	M	$435.23	$618.72
4	914	THOMPSON	11584	M	$221.09	$297.56
5	915	WOOD	8113	F	$238.17	$372.24
6	915	TAYLOR	2943	F	$287.22	$401.72
7	915	MOORE	12649	M	$557.18	$728.12
8	915	HELMS	3658	M	$479.26	$701.26
9	916	SHIRES	10052	F	$487.12	$729.22
10	916	RICHARDS	5781	M	$319.27	$472.84
11	916	SMITH	6237	F	$207.09	$345.88
12	916	MURPHY	6927	M	$272.66	$385.11
13	916	FAIRLEY	8846	M	$521.66	$834.80
14	917	HERZOG	6433	M	$692.57	$1,045.26
15	917	TALL	11931	M	$355.19	$492.26
16	917	HIGGINS	11658	M	$777.50	$1,235.46
17	917	DOOB	10554	F	$669.06	$972.29
18	918	EPERT	7716	M	$224.36	$310.40
19	918	CLANCY	6648	M	$245.37	$347.12
20	918	POWELL	8123	F	$789.39	$1,271.54

Chapter 9 Output

9.1 d.

OBS	DEPT	NAME	NUMBER	SEX	NETPAY	GROSSPAY
1	MARKETING	LARSON	11357	FEMALE	$265.47	$383.92
2	MARKETING	RYAN	10961	MALE	$291.56	$399.20
3	MARKETING	JOHNSON	10546	MALE	$435.23	$618.72
4	MARKETING	THOMPSON	11584	MALE	$221.09	$297.56
5	MAINTENANCE	WOOD	8113	FEMALE	$238.17	$372.24
6	MAINTENANCE	TAYLOR	2943	FEMALE	$287.22	$401.72
7	MAINTENANCE	MOORE	12649	MALE	$557.18	$728.12
8	MAINTENANCE	HELMS	3658	MALE	$479.26	$701.26
9	RESEARCH & DEVELOPMENT	SHIRES	10052	FEMALE	$487.12	$729.22
10	RESEARCH & DEVELOPMENT	RICHARDS	5781	MALE	$319.27	$472.84
11	RESEARCH & DEVELOPMENT	SMITH	6237	FEMALE	$207.09	$345.88
12	RESEARCH & DEVELOPMENT	MURPHY	6927	MALE	$272.66	$385.11
13	RESEARCH & DEVELOPMENT	FAIRLEY	8846	MALE	$521.66	$834.80
14	ADMINISTRATION	HERZOG	6433	MALE	$692.57	$1,045.26
15	ADMINISTRATION	TALL	11931	MALE	$355.19	$492.26
16	ADMINISTRATION	HIGGINS	11658	MALE	$777.50	$1,235.46
17	ADMINISTRATION	DOOB	10554	FEMALE	$669.06	$972.29
18	EDUCATION	EPERT	7716	MALE	$224.36	$310.40
19	EDUCATION	CLANCY	6648	MALE	$245.37	$347.12
20	EDUCATION	POWELL	8123	FEMALE	$789.39	$1,271.54

9.1 e.

VARIABLE	LABEL	MEAN
NETPAY	NET PAY OF EMPLOYEE	416.82100000

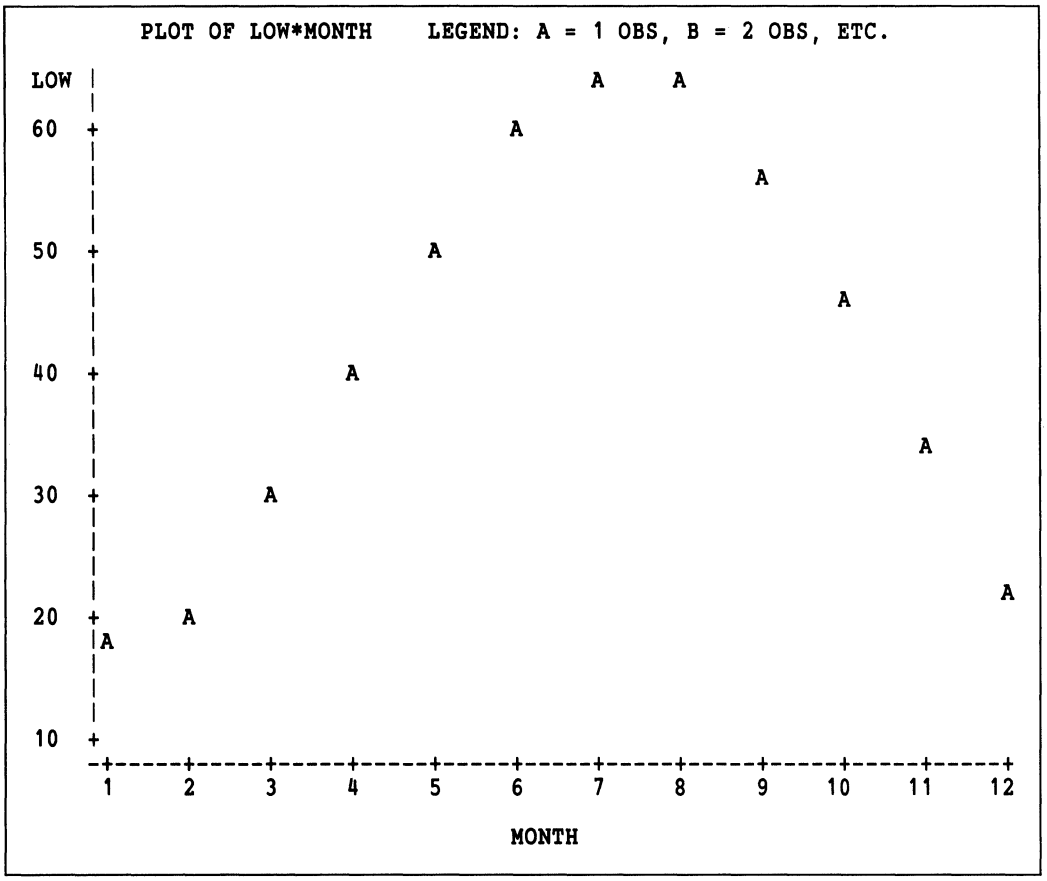
Chapter 9 Output

9.1 f.

```
VARIABLE   LABEL                          MEAN

---------- DEPARTMENT =MARKETING ----------

NETPAY     NET PAY OF EMPLOYEE   303.33750000

---------- DEPARTMENT =MAINTENANCE ----------

NETPAY     NET PAY OF EMPLOYEE   390.45750000

---- DEPARTMENT =RESEARCH & DEVELOPMENT -----

NETPAY     NET PAY OF EMPLOYEE   361.56000000

-------- DEPARTMENT =ADMINISTRATION ---------

NETPAY     NET PAY OF EMPLOYEE   623.58000000

---------- DEPARTMENT =EDUCATION ----------

NETPAY     NET PAY OF EMPLOYEE   419.70666667
```

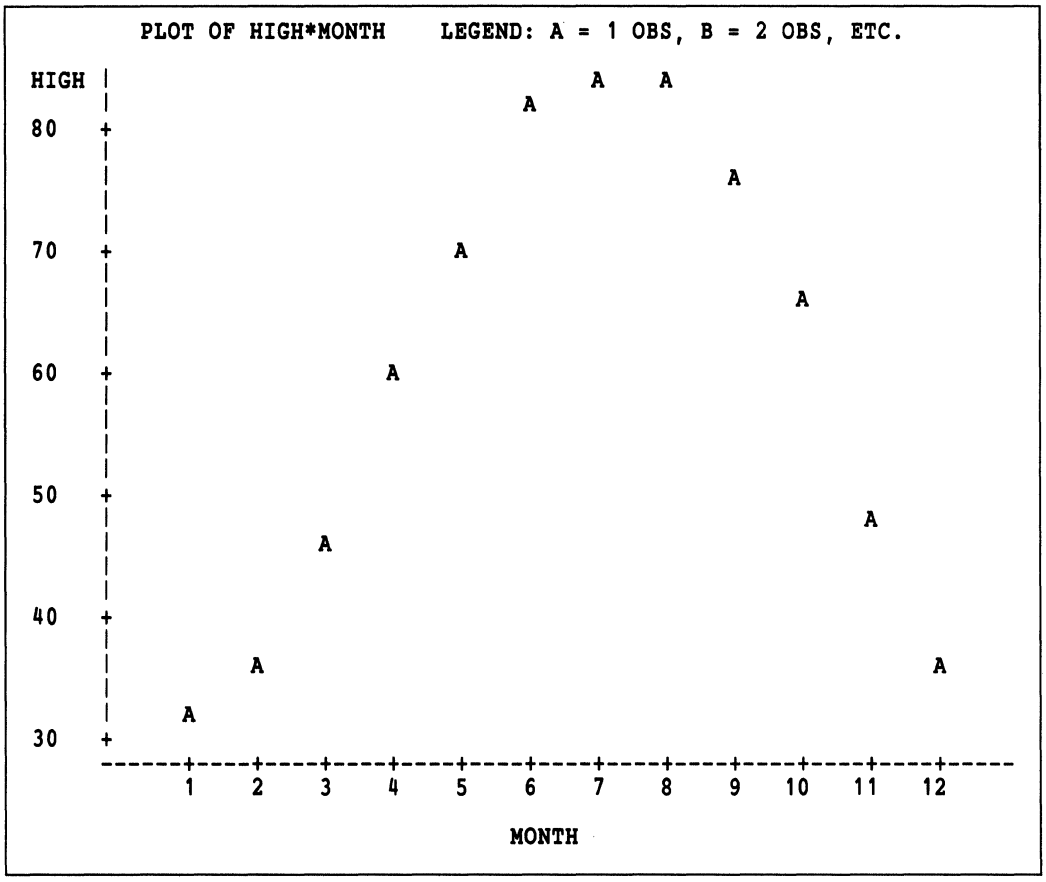
Chapter 11 Output

11.1 b.



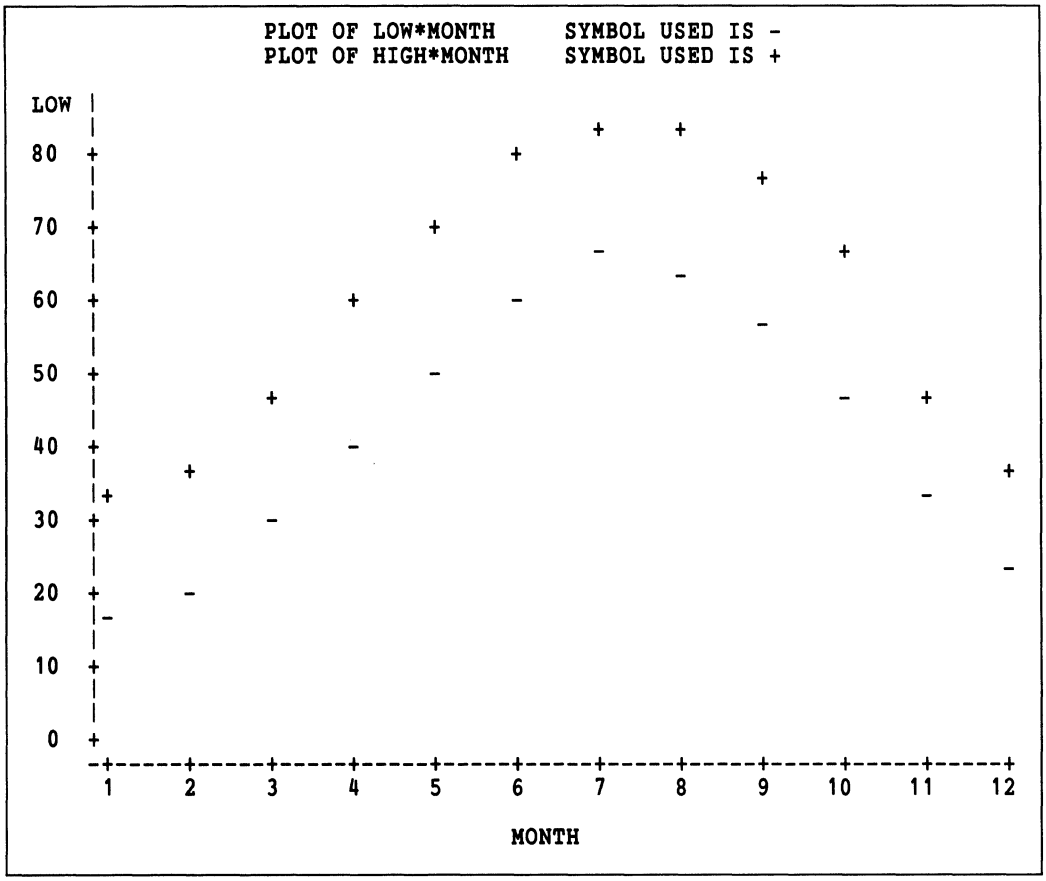
Chapter 11 Output

11.1 c.



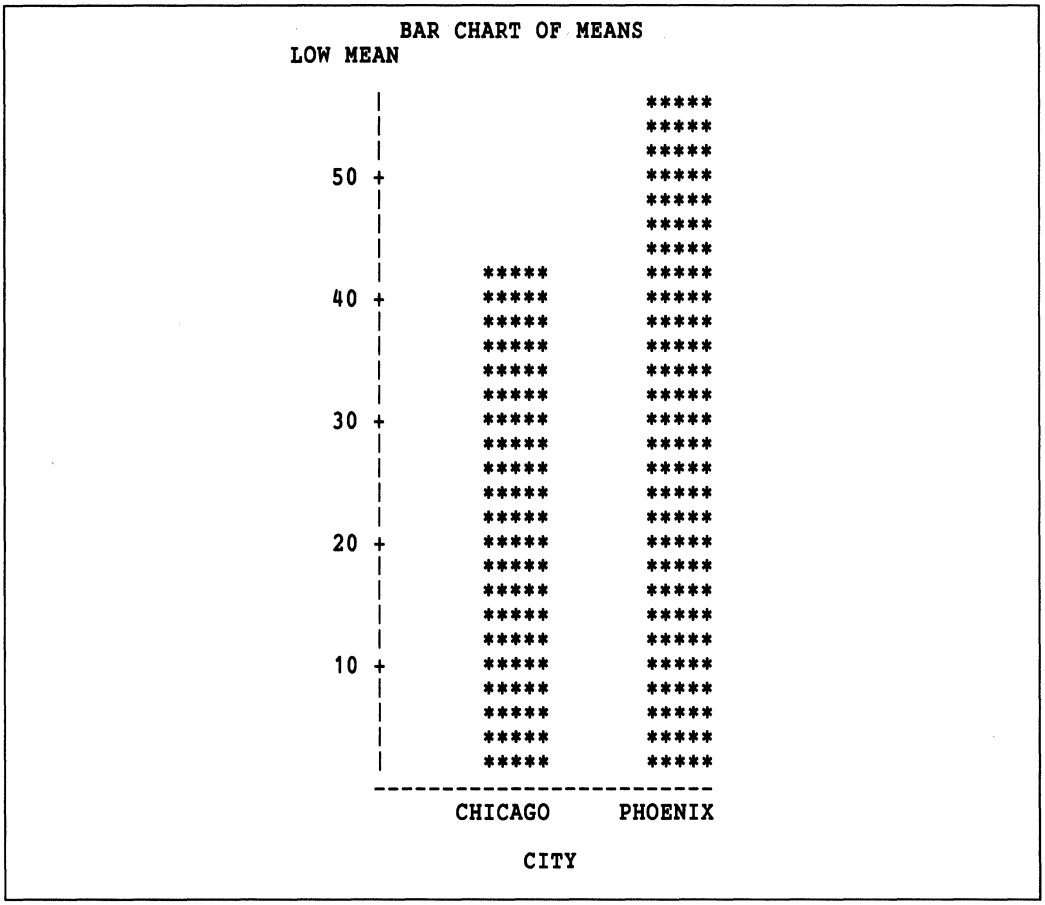
Chapter 11 Output

11.1 d.



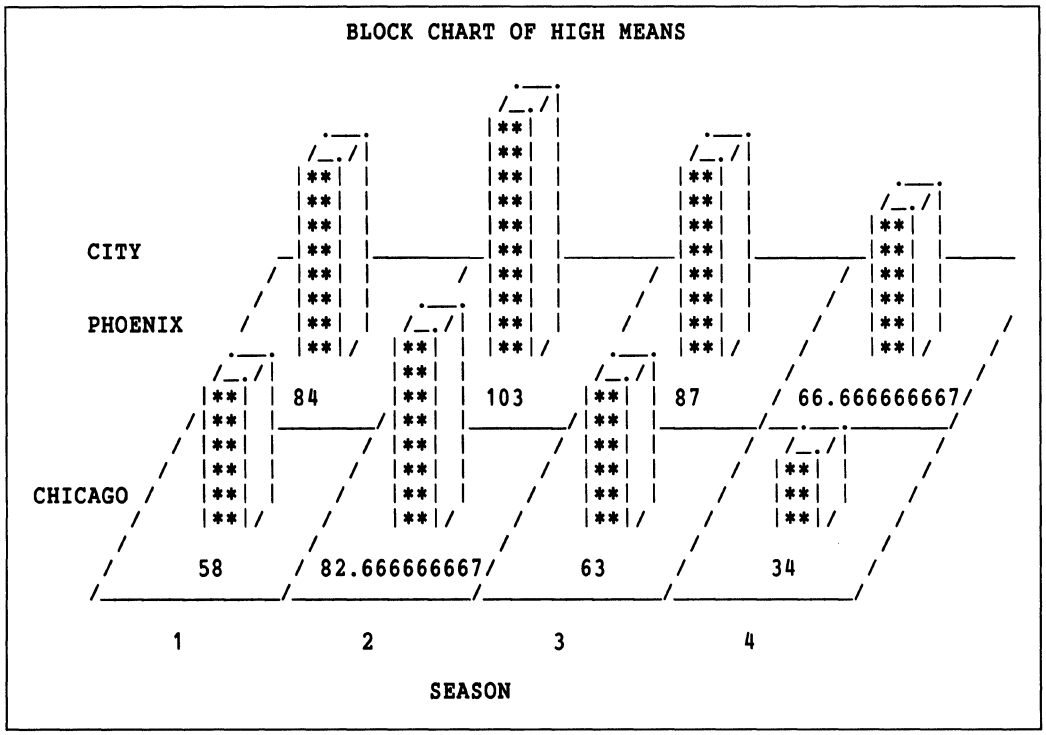
Chapter 11 Output

11.2 b.



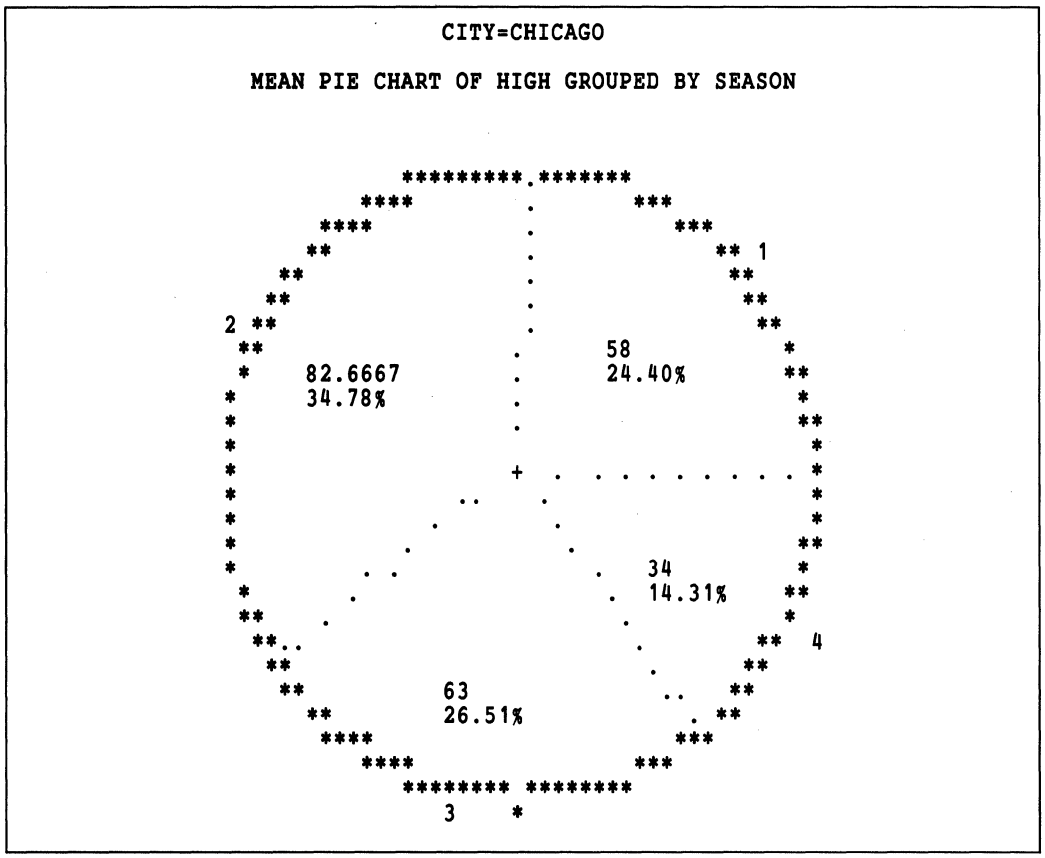
Chapter 11 Output

11.2 c.



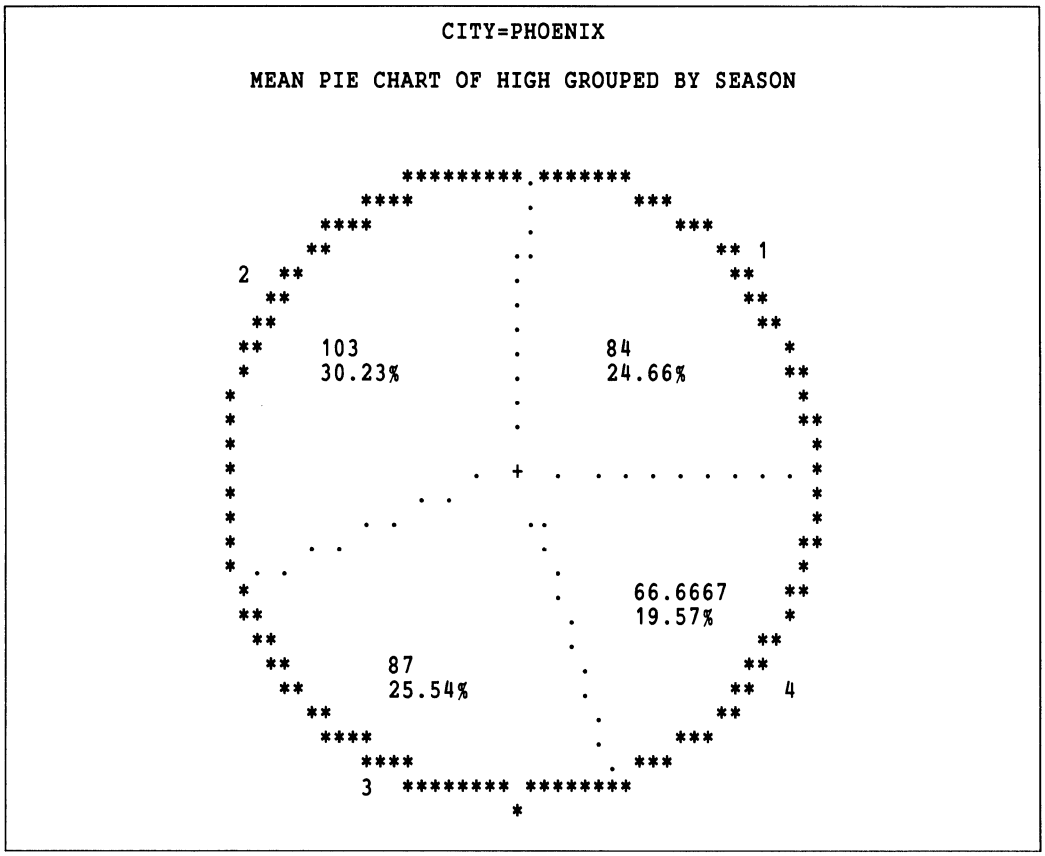
Chapter 11 Output

11.2 d.



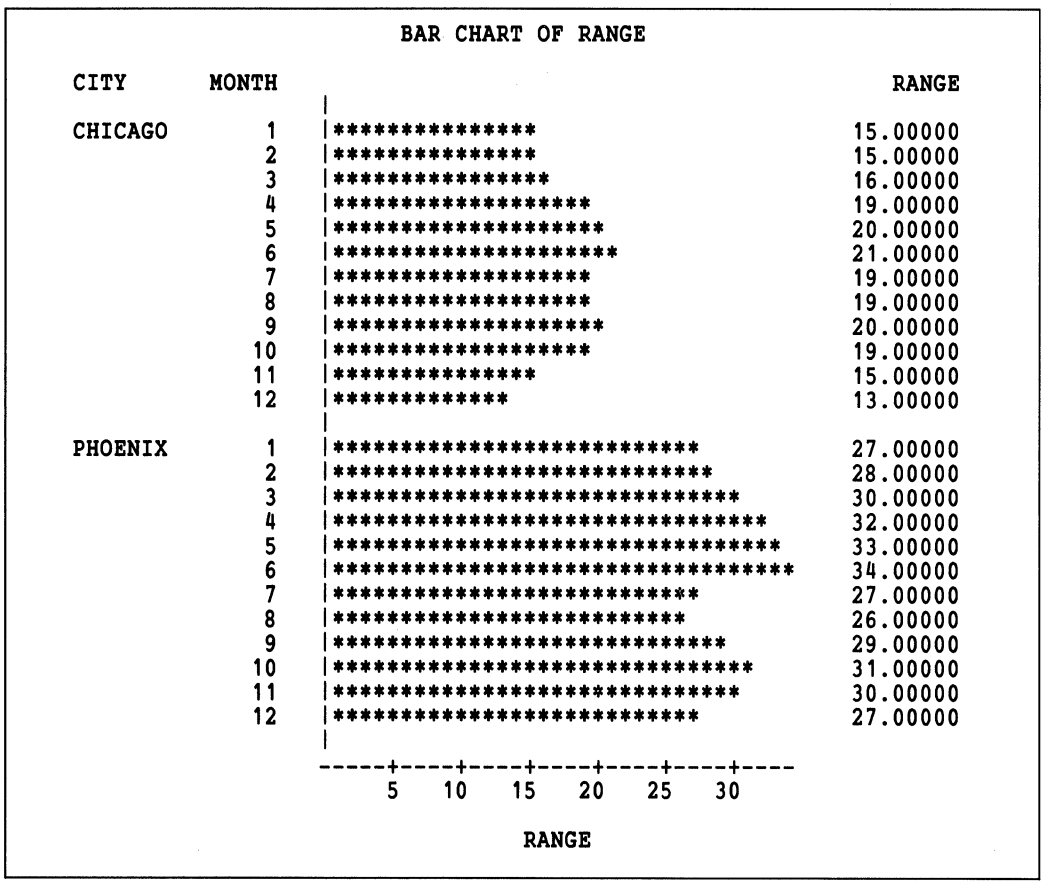
Chapter 11 Output

11.2 d. (continued)



Chapter 11 Output

11.2 e.



BAR CHART OF RANGE

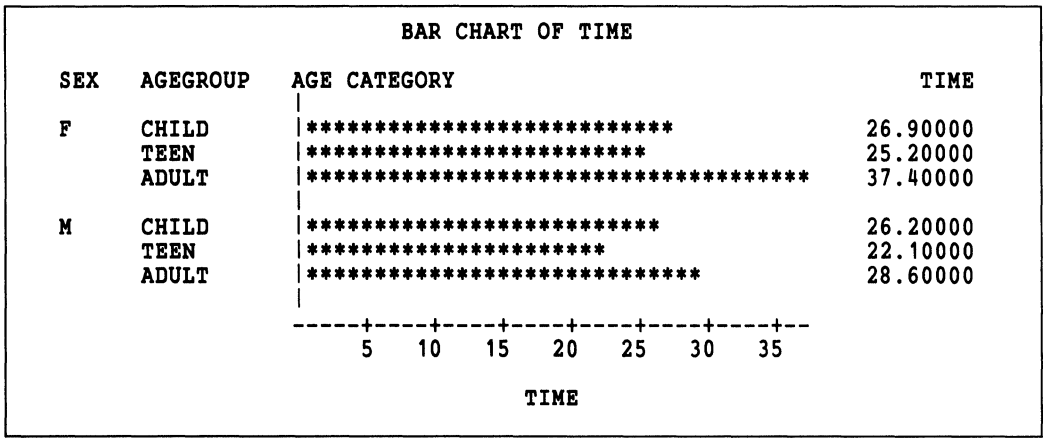
CITY	MONTH	RANGE
CHICAGO	1	15.00000
	2	15.00000
	3	16.00000
	4	19.00000
	5	20.00000
	6	21.00000
	7	19.00000
	8	19.00000
	9	20.00000
	10	19.00000
	11	15.00000
	12	13.00000
PHOENIX	1	27.00000
	2	28.00000
	3	30.00000
	4	32.00000
	5	33.00000
	6	34.00000
	7	27.00000
	8	26.00000
	9	29.00000
	10	31.00000
	11	30.00000
	12	27.00000

5 10 15 20 25 30

RANGE

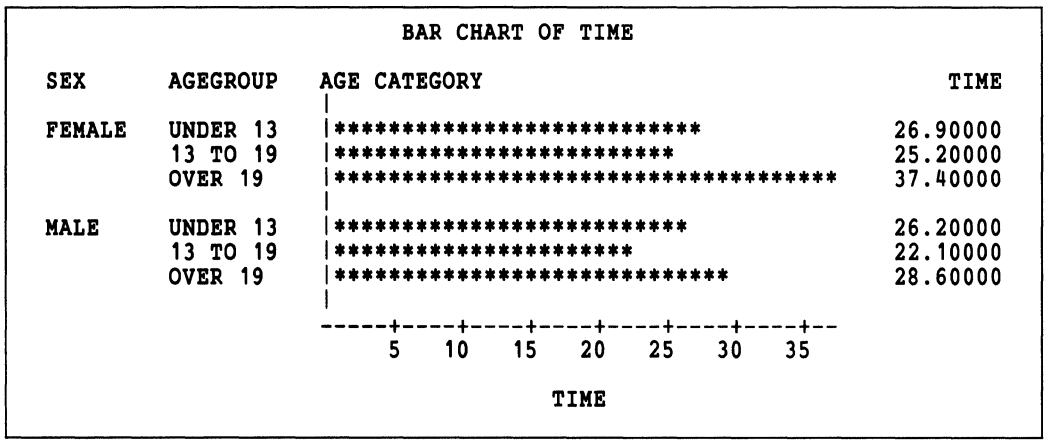
Chapter 11 Output

11.3 a.



```
                              BAR CHART OF TIME

 SEX    AGEGROUP    AGE CATEGORY                                      TIME
                    |
 F      CHILD       |*************************                      26.90000
        TEEN        |************************                       25.20000
        ADULT       |************************************           37.40000
                    |
 M      CHILD       |*************************                      26.20000
        TEEN        |**********************                         22.10000
        ADULT       |****************************                   28.60000
                    |
                    -----+----+----+----+----+----+----+--
                         5    10   15   20   25   30   35

                                    TIME
```

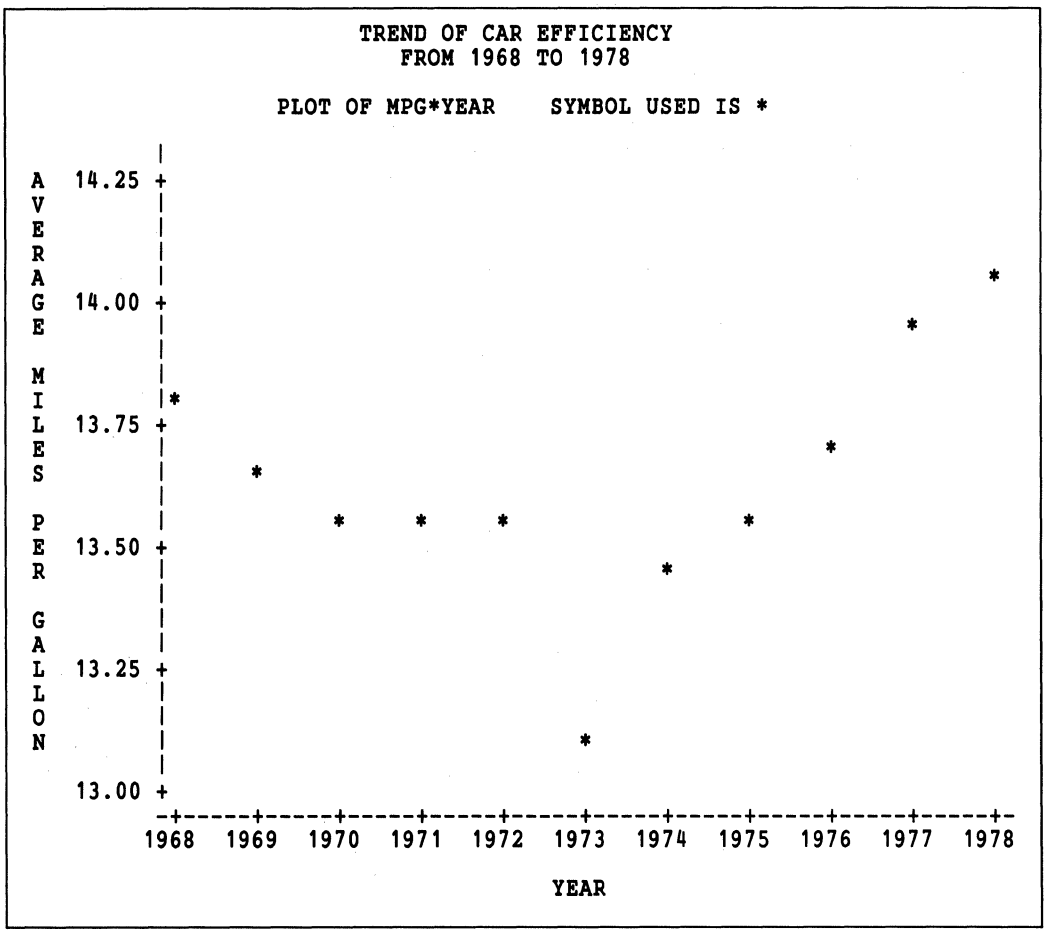
11.3 b.



```
                              BAR CHART OF TIME

 SEX       AGEGROUP    AGE CATEGORY                                   TIME
                       |
 FEMALE    UNDER 13    |*************************                   26.90000
           13 TO 19    |************************                    25.20000
           OVER 19     |************************************        37.40000
                       |
 MALE      UNDER 13    |*************************                   26.20000
           13 TO 19    |**********************                      22.10000
           OVER 19     |****************************                28.60000
                       |
                       -----+----+----+----+----+----+----+--
                            5    10   15   20   25   30   35

                                    TIME
```

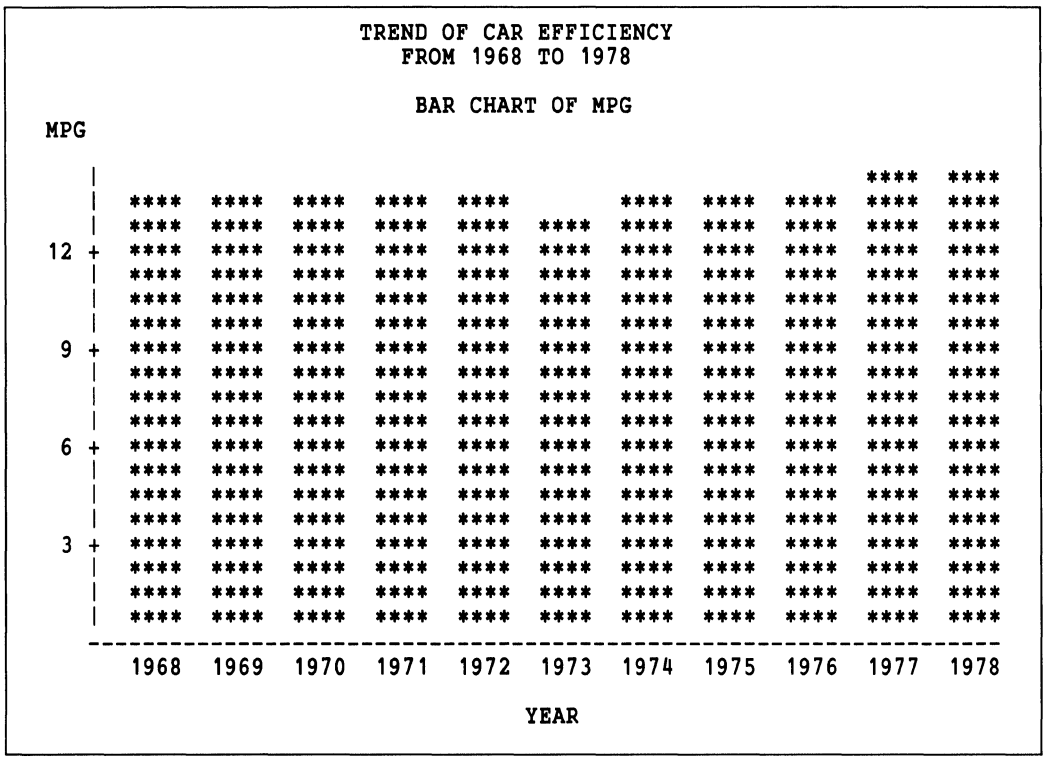
Chapter 11 Output

11.4 a.



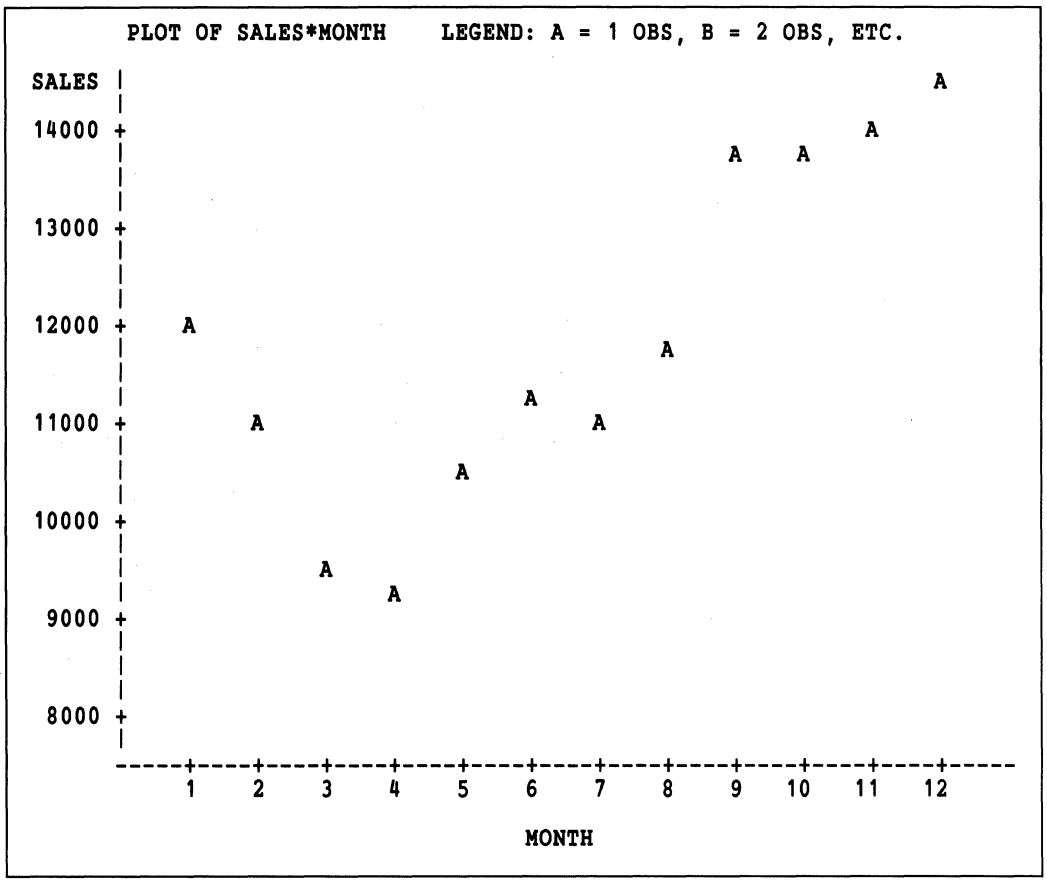
Chapter 11 Output

11.4 b.



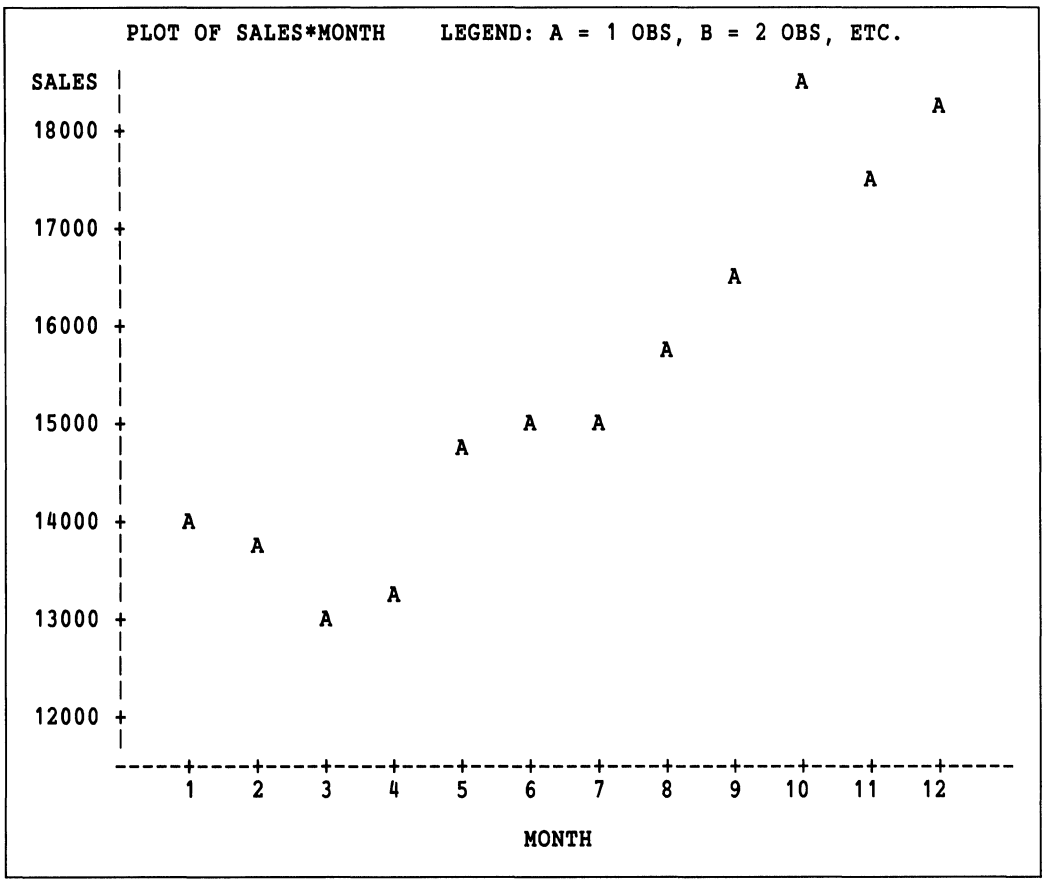
Chapter 12 Output

12.3 a.



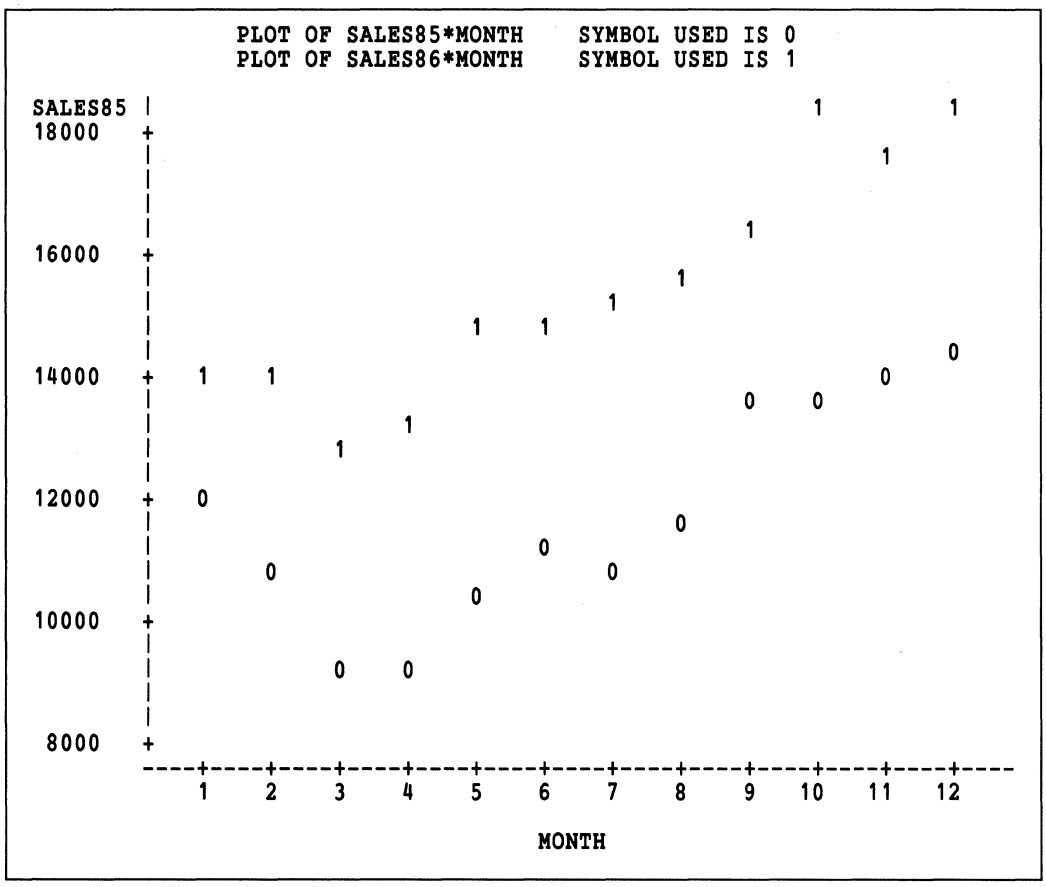
Chapter 12 Output

12.3 b.



Chapter 12 Output

12.3 c.



Chapter 12 Output

12.3 d.

```
VARIABLE          MEAN

------- MONTH=1 -------

SALES      12985.0000000

------- MONTH=2 -------

SALES      12364.5000000

------- MONTH=3 -------

SALES      11150.5000000

------- MONTH=4 -------

SALES      11277.0000000

------- MONTH=5 -------

SALES      12686.5000000

------- MONTH=6 -------

SALES      13160.5000000

------- MONTH=7 -------

SALES      12998.0000000

------- MONTH=8 -------

SALES      13671.5000000
```

Chapter 12 Output

12.3 d. (continued)

```
VARIABLE          MEAN

------- MONTH=9 -------

SALES      15183.0000000

------ MONTH=10 -------

SALES      16083.5000000

------ MONTH=11 -------

SALES      15739.5000000

------ MONTH=12 -------

SALES      16473.0000000
```

Chapter 15 Output

15.1

```
--------------------------------DEPT=914--------------------------------

              EMPLOYEE'S        SOCIAL SECURITY     ANNUAL
                 NAME              NUMBER           SALARY

              LARSON,J.L.         152-72-7162       $21,500
              RYAN,K.E.           276-38-2167       $23,600
              JOHNSON,M.C.        267-66-2722       $32,250
              THOMPSON,P.L.       278-26-3933       $16,500
              ------------                          -------
              DEPT                                  $93,850


--------------------------------DEPT=915--------------------------------

               EMPLOYEE'S       SOCIAL SECURITY      ANNUAL
                  NAME             NUMBER            SALARY

               WOOD,A.K.          926-31-6235       $15,800
               TAYLOR,W.R.        921-37-7463       $27,900
               MOORE,S.E.         763-94-8493       $58,000
               HELMS,T.W.         328-74-6832       $34,500
               -----------                          --------
               DEPT                                 $136,200


--------------------------------DEPT=916--------------------------------

              EMPLOYEE'S        SOCIAL SECURITY      ANNUAL
                 NAME              NUMBER            SALARY

              SHIRES,A.M.         324-08-9747       $10,800
              RICHARDS,T.R.       983-27-4783       $11,500
              SMITH,D.P.          438-76-4673       $46,900
              MURPHY,D.H.         849-73-3343       $24,000
              FAIRLEY,R.R.        324-64-2639       $20,800
              ------------                          --------
              DEPT                                  $114,000
```

Chapter 15 Output

15.1 (continued)

```
--------------------------------DEPT=917--------------------------------

              EMPLOYEE'S       SOCIAL SECURITY       ANNUAL
                 NAME             NUMBER             SALARY

              HERZOG,T.          783-86-3732        $16,600
              TALL,F.B.          937-43-3533        $28,700
              HIGGINS,H.L.       845-58-2644        $31,000
              DOOB,P.E.          743-68-2622        $42,000
              ------------                          --------
              DEPT                                  $118,300


--------------------------------DEPT=918--------------------------------

               EMPLOYEE'S      SOCIAL SECURITY       ANNUAL
                  NAME            NUMBER             SALARY

               EPERT,C.D.         637-25-4733        $12,000
               CLANCY,L.G.        783-26-4872        $13,200
               POWELL,E.          763-63-3327        $36,000
               -----------                           --------
               DEPT                                  $61,200
                                                     ========
                                                     $523,550
```

Chapter 15 Output

15.2

SOCIAL SECURITY NUMBER	DEPARTMENT NUMBER	ANNUAL SALARY
152-72-7162	914	$21,500
276-38-2167	914	$23,600
267-66-2722	914	$32,250
278-26-3933	914	$16,500
926-31-6235	915	$15,800
921-37-7463	915	$27,900
763-94-8493	915	$58,000
328-74-6832	915	$34,500
324-08-9747	916	$10,800
983-27-4783	916	$11,500
438-76-4673	916	$46,900
849-73-3343	916	$24,000
324-64-2639	916	$20,800
783-86-3732	917	$16,600
937-43-3533	917	$28,700
845-58-2644	917	$31,000
743-68-2622	917	$42,000
637-25-4733	918	$12,000
783-26-4872	918	$13,200
763-63-3327	918	$36,000

Index

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Z

Proofreading was performed in the **Technical Writing Department** by **Frances A. Kienzle**, **E. Ellen Fussell**, **Mariam C. Chilman**, **Lisa K. Hunt**, and **Deborah J. Vause** under the supervision of **David D. Baggett**. **Gigi Hassan** is index editor.

Graphic Arts provided coding, typesetting and production under the direction of **Carol M. Thompson**. Preliminary production was provided by **Gail C. Freeman**, **Blanche W. Phillips**, and **James K. Hart**. Final production was provided by **Arlene B. Drezek** and **Joseph H. Moore, Jr.** Text composition programming was provided by **Craig R. Sampson**, **Pamela A. Troutman**, and **Cynthia M. Hopkins**.

Creative Services artist **Michael J. Pezzoni** provided illustrations under the direction of **Jennifer K. Davis**.

Your Turn

If you have comments about SAS Institute's instructor-based training or the written materials used for this course, please let us know by writing to us with your ideas.

Write the Course Development Department, Education Division, SAS Institute Inc., SAS Circle, Box 8000, Cary, NC 27512-8000.

1-55544-004-5