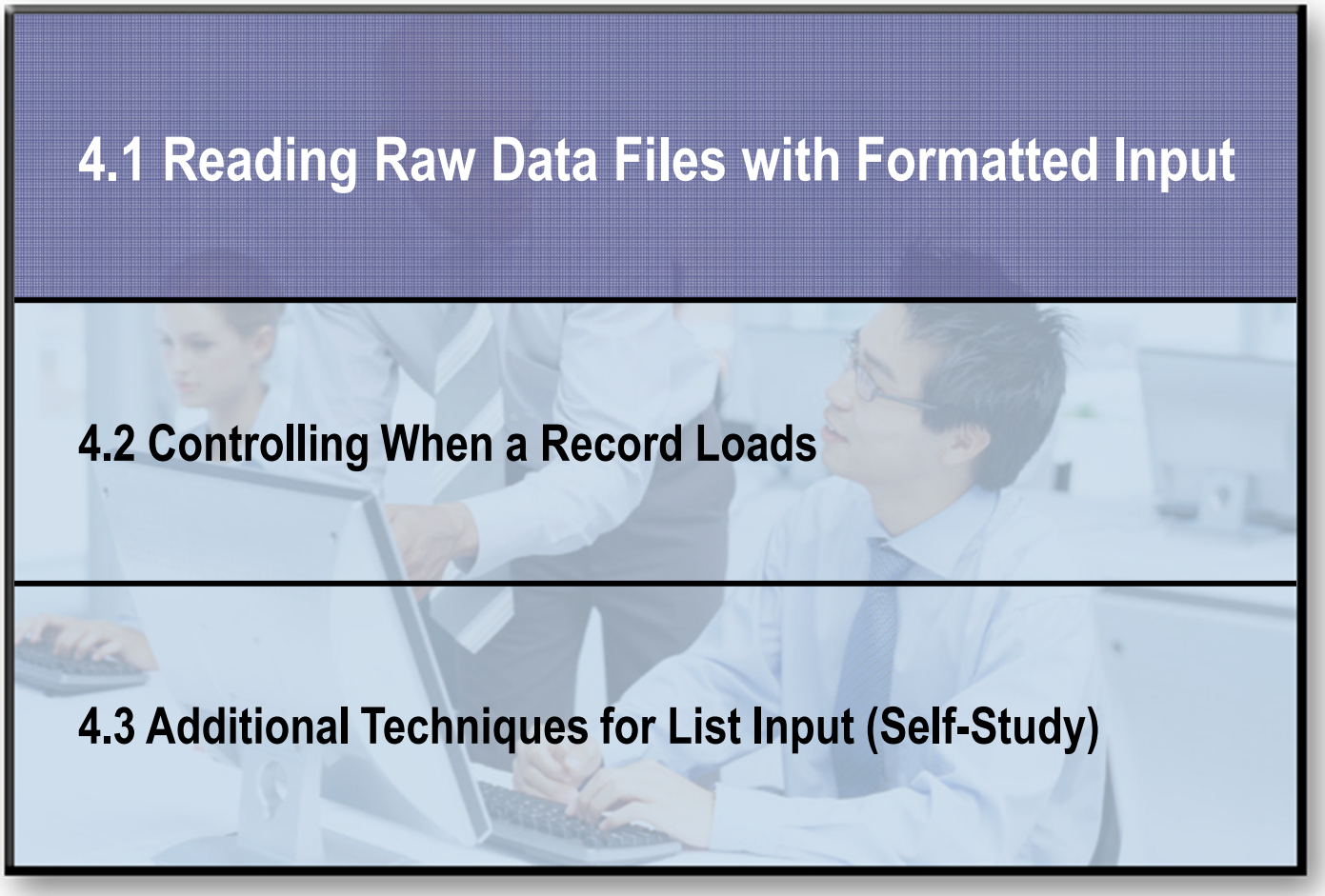
Chapter 4: Reading Raw Data Files



Objectives

- Read raw data in fixed columns using formatted input.

Business Scenario – Read the Offers File

The **offers.dat** raw data file contains information about discount offers. Create a SAS data set named **discounts** from the raw data.

Layout: offers.dat

Description	Column
Customer Type	1- 4
Offer Date	5-12
Item Group	14-21
Discount	22-24

Partial offers.dat

```
                1    1    2    2
1---5----0----5----0----5
104012/02/07 Outdoors15%
202010/07/07 Golf     7%
103009/22/07 Shoes   10%
103009/22/07 Clothes 10%
202007/08/07 Clothes 15%
203007/08/07 Clothes 25%
```

Business Scenario – Desired Output

The **discounts** data set should have one observation per input record.

Partial Listing of **discounts**

Cust_ type	Offer_dt	Item_gp	Discount
1040	02DEC2007	Outdoors	0.15
2020	07OCT2007	Golf	0.07
1030	22SEP2007	Shoes	0.10
1030	22SEP2007	Clothes	0.10
	.		
	.		
3010	17MAY2007	Clothes	0.15

4.01 Multiple Choice Poll

For your work, how often do you need to read raw data files?

a. All the time
b. Occasionally
c. Very rarely
d. Never

The DATA Step to Read Raw Data (Review)

To read raw data, the DATA step includes DATA, INFILE, and INPUT statements.

```
DATA output-SAS-data-set;
     INFILE 'raw-data-file-name';
     INPUT specifications;
     <additional SAS statements>
RUN;
```

4.02 Quiz

Use SAS Help to navigate to Starting with Raw Data: The Basics.

Click the **Contents** tab and select:

⇨ **SAS Products**

⇨ **Base SAS**

⇨ **Step-by-Step Programming with Base SAS Software**

⇨ **Getting Your Data into Shape**

⇨ **Starting with Raw Data: The Basics**

Page to Introduction to Raw Data and review this section. What are the three styles of input?

Which Input Style to Choose?

Column input, formatted input, and list input
are all styles of writing INPUT statement specifications.

Style	Used for Reading
Column Input	Standard data in fixed columns
Formatted Input	Standard and nonstandard data in fixed columns
List Input	Standard and nonstandard data separated by blanks or some other delimiter

Standard and Nonstandard Data (Review)

- *Standard data* is data that SAS can read without any special instructions.

  Examples of standard numeric data:

  58 -23 67.23 00.99 5.67E5 1.2E-2

- *Nonstandard data* is any data that SAS cannot read without special instructions.

  Examples of nonstandard numeric data:

  5,823 15% $67.23 01/12/1999 12MAY2006

4.03 Quiz

Which style of INPUT statement specification should you choose to read the **`offers.dat`** raw data file?

Partial `offers.dat`

```
          1    1    2    2
1---5----0----5----0----5
104012/02/07 Outdoors15%
202010/07/07 Golf     7%
103009/22/07 Shoes   10%
103009/22/07 Clothes 10%
```

Reading Data Using Formatted Input

General form of the INPUT statement with formatted input:

INPUT *pointer-control variable informat* . . . ;

Formatted input is used to read data values by

- moving the input pointer to the starting position of the field
- naming the variable
- specifying an informat.

Example:

```
input @5 FirstName $10.;
```

Reading Data Using Formatted Input

Column pointer controls:

@*n* moves the pointer to column *n.*

+*n* moves the pointer *n* positions.

An informat specifies the following:

- the width of the input field
- how to read data values stored in the field

SAS Informat Examples

Examples of informats showing the raw data values and the converted SAS numeric values:

Informat	Raw Data Value	SAS Data Value
$8.	Outdoors	Outdoors
5.	12345	12345
COMMA7. DOLLAR7.	$12,345	12345
COMMAX7. DOLLARX7.	$12.345	12345
EUROX7.	€12.345	12345
PERCENT3.	15%	.15

SAS Date Informat Examples

Examples of date informats showing the nonstandard raw data values and the converted SAS numeric values:

Informat	Raw Data Value	SAS Date Value
MMDDYY6.	010160	0
MMDDYY8.	01/01/60	0
MMDDYY10.	01/01/1960	0
DDMMYY6.	311260	365
DDMMYY8.	31/12/60	365
DDMMYY10.	31/12/1960	365
DATE7.	31DEC59	-1
DATE9.	31DEC1959	-1

Business Scenario – Continued

Use formatted input to create a SAS data set named **discounts** from the raw data in **offers.dat**.

Layout: offers.dat

Description	Column
Customer Type	1- 4
Offer Date	5-12
Item Group	14-21
Discount	22-24

Partial offers.dat

```
                1    1    2    2
1---5----0----5----0----5
104012/02/07 Outdoors15%
202010/07/07 Golf     7%
103009/22/07 Shoes   10%
103009/22/07 Clothes 10%
202007/08/07 Clothes 15%
203007/08/07 Clothes 25%
```

Write INPUT Specifications

Identify the starting position, variable name, and informat for each input field.

```
input @1 Cust_type 4.
```

Layout: `offers.dat`

Description	Column
Customer Type	1- 4
Offer Date	5-12
Item Group	14-21
Discount	22-24

Partial `offers.dat`

```
                1    1    2    2
1---5----0----5----0----5
1040 12/02/07 Outdoors 15%
202010/07/07 Golf       7%
103009/22/07 Shoes     10%
103009/22/07 Clothes   10%
202007/08/07 Clothes   15%
203007/08/07 Clothes   25%
```

4.04 Quiz

Continue writing the INPUT statement to read Offer Date. (Hint: Use the MMDDYY8. informat.)

```
input @1 Cust_type 4.
      ?
```

Layout: `offers.dat`

Description	Column
Customer Type	1- 4
Offer Date	5-12
Item Group	14-21
Discount	22-24

Partial `offers.dat`

```
                1    1    2    2
1---5----0----5----0----5
1040 12/02/07 Outdoors 15%
202010/07/07 Golf      7%
103009/22/07 Shoes    10%
103009/22/07 Clothes  10%
202007/08/07 Clothes  15%
203007/08/07 Clothes  25%
```

Reading Data Using Formatted Input

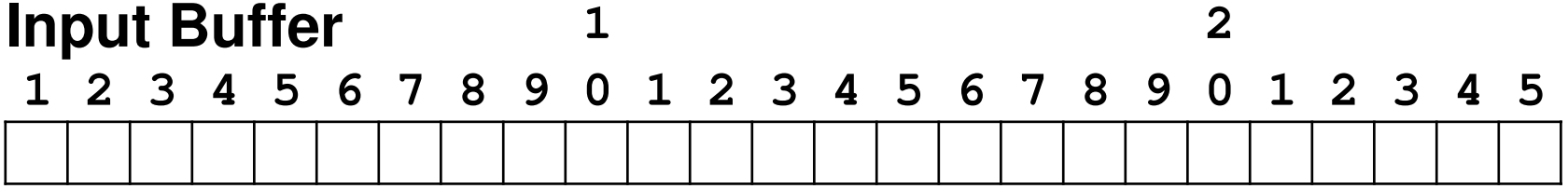
This SAS program uses formatted input to read the raw data file in **offers.dat**.

```
data work.discounts;
   infile 'offers.dat';
   input @1 Cust_type 4.
         @5 Offer_dt mmddyy8.
         @14 Item_gp $8.
         @22 Discount percent3.;
run;
```

Compilation: Formatted Input

```
data work.discounts;
   infile 'offers.dat';
   input @1 Cust_type 4.
         @5 Offer_dt mmddyy8.
         @14 Item_gp $8.
         @22 Discount percent3.;
run;
```

Input Buffer



PDV

Cust_type N 8	Offer_dt N 8	Item_gp $ 8	Discount N 8

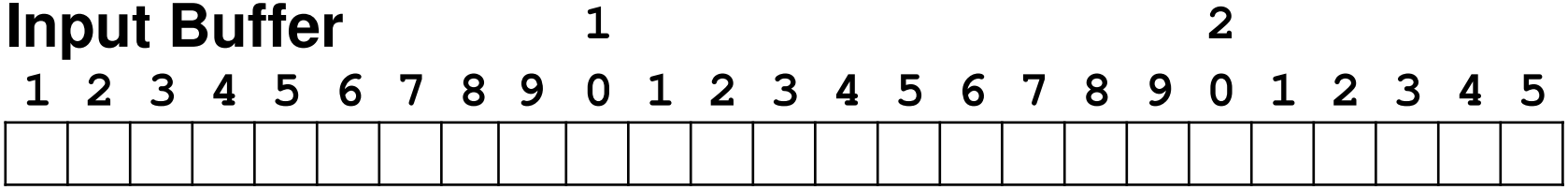
...

Execution: Formatted Input

```
data work.discounts;
   infile 'offers.dat';
   input @1 Cust_type 4.
         @5 Offer_dt mmddyy8.
         @14 Item_gp $8.
         @22 Discount percent3.;
run;
```

Initialize PDV

Input Buffer



PDV

Cust_type	Offer_dt	Item_gp	Discount
N 8	N 8	$ 8	N 8
.	.		.

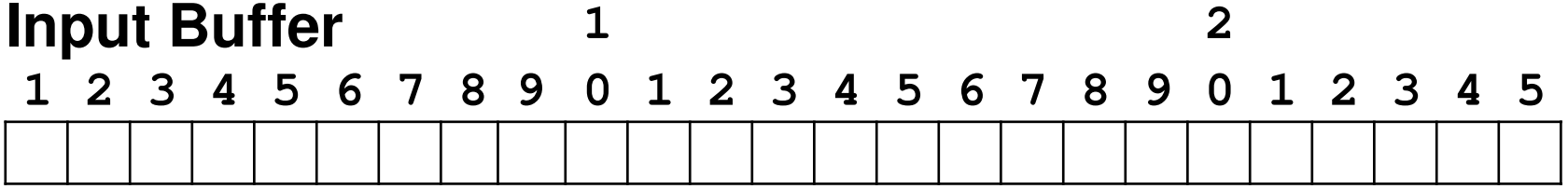
...

Execution: Formatted Input

```
data work.discounts;
    infile 'offers.dat';
    input @1 Cust_type 4.
          @5 Offer_dt mmddyy8.
          @14 Item_gp $8.
          @22 Discount percent3.;
run;
```

Specify input data file

Input Buffer



PDV

Cust_type	Offer_dt	Item_gp	Discount
N 8	N 8	$ 8	N 8
.	.		.

...

Execution: Formatted Input

```
data work.discounts;
   infile 'offers.dat';
   input  @1 Cust_type 4.
          @5 Offer_dt mmddyy8.
          @14 Item_gp $8.
          @22 Discount percent3.;
run;
```

Load input buffer

Input Buffer

```
          1                   2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
1 0 4 0 1 2 / 0 2 / 0 7   O u t d o o r s 1 5 %
```

PDV

Cust_type N 8	Offer_dt N 8	Item_gp $ 8	Discount N 8
.	.		.

...

Execution: Formatted Input

```
data work.discounts;
   infile 'offers.dat';
   input @1 Cust_type 4.
         @5 Offer_dt mmddyy8.
         @14 Item_gp $8.
         @22 Discount percent3.;
run;
```

Load first value into the PDV

Input Buffer

	1										2													
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	0	4	0	1	2	/	0	2	/	0	7		O	u	t	d	o	o	r	s	1	5	%	

PDV

Cust_type N 8	Offer_dt N 8	Item_gp $ 8	Discount N 8
1040	.		.

...

Execution: Formatted Input

```
data work.discounts;
   infile 'offers.dat';
   input @1 Cust_type 4.
         @5 Offer_dt mmddyy8.
         @14 Item_gp $8.
         @22 Discount percent3.;
run;
```

Load second value into the PDV

Input Buffer

	1										2													
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	0	4	0	1	2	/	0	2	/	0	7		O	u	t	d	o	o	r	s	1	5	%	

PDV

Cust_type N 8	Offer_dt N 8	Item_gp $ 8	Discount N 8
1040	17502		.

...

Execution: Formatted Input

```
data work.discounts;
   infile 'offers.dat';
   input @1 Cust_type 4.
         @5 Offer_dt mmddyy8.
         @14 Item_gp $8.
         @22 Discount percent3.;
run;
```

Load third value into the PDV

Input Buffer

	1									2													
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	0	4	0	1	2	/	0	2	/	0	7		O	u	t	d	o	o	r	s	1	5	%	

PDV

Cust_type N 8	Offer_dt N 8	Item_gp $ 8	Discount N 8
1040	17502	Outdoors	.

...

Execution: Formatted Input

```
data work.discounts;
   infile 'offers.dat';
   input @1 Cust_type 4.
         @5 Offer_dt mmddyy8.
         @14 Item_gp $8.
         @22 Discount percent3.;
run;
```

Load fourth value into the PDV

Input Buffer

	1									2														
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	0	4	0	1	2	/	0	2	/	0	7		O	u	t	d	o	o	r	s	1	5	%	

PDV

Cust_type N 8	Offer_dt N 8	Item_gp $ 8	Discount N 8
1040	17502	Outdoors	.15

...

Execution: Formatted Input

```
data work.discounts;
   infile 'offers.dat';
   input @1 Cust_type 4.
         @5 Offer_dt mmddyy8.
         @14 Item_gp $8.
         @22 Discount percent3.;
run;
```

Implicit OUTPUT;
Implicit RETURN;

Input Buffer

	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
	1	0	4	0	1	2	/	0	2	/	0	7		O	u	t	d	o	o	r	s	1	5	%	

PDV

Cust_type N 8	Offer_dt N 8	Item_gp $ 8	Discount N 8
1040	17502	Outdoors	.15

...

Execution: Formatted Input

```
data work.discounts;
   infile 'offers.dat';
   input @1 Cust_type 4.
         @5 Offer_dt mmddyy8.
         @14 Item_gp $8.
         @22 Discount percent3.;
run;
```

Continue until EOF

Input Buffer

	1										2													
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
3	0	1	0	0	5	/	1	7	/	0	7		C	l	o	t	h	e	s		1	5	%	

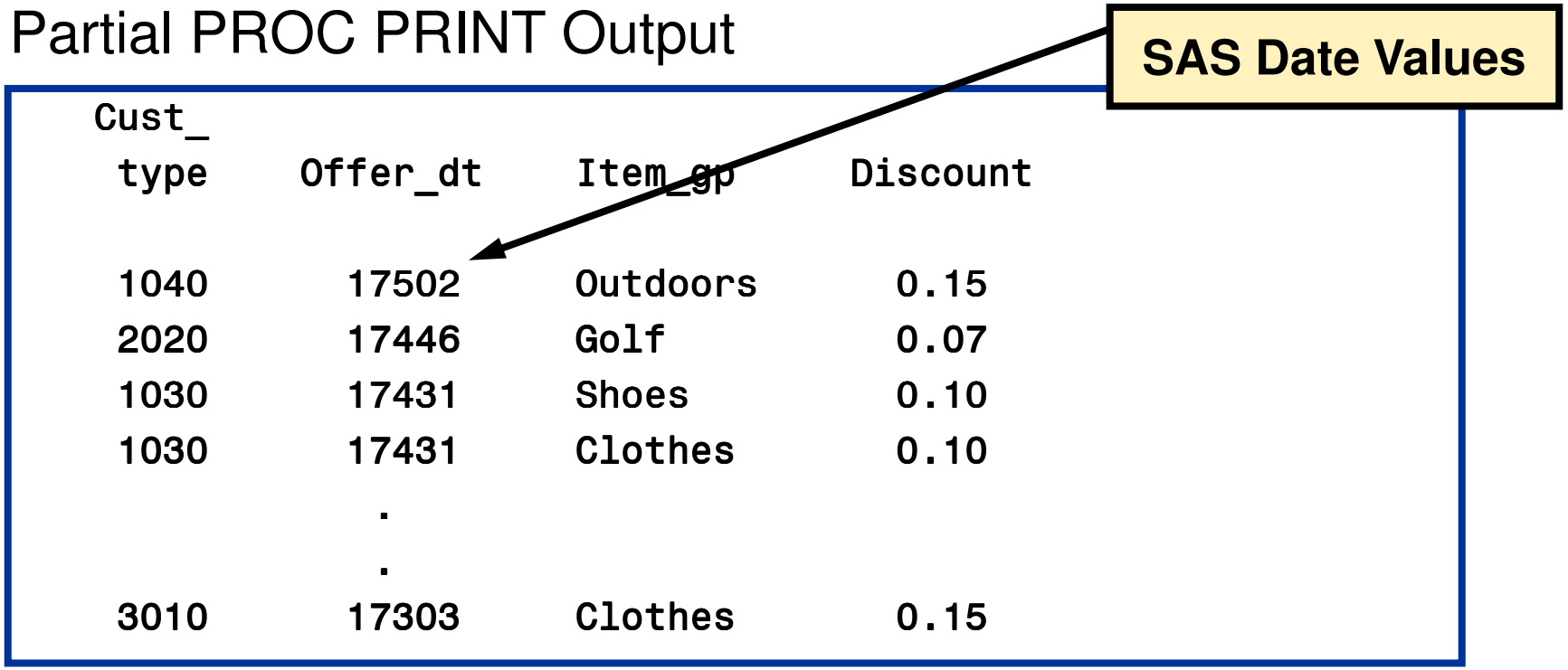
PDV

Cust_type N 8	Offer_dt N 8	Item_gp $ 8	Discount N 8
3010	17303	Clothes	.15

Read Discount Offers File – Output

```
proc print data=work.discounts noobs;
run;
```

Partial PROC PRINT Output



Cust_ type	Offer_dt	Item_gp	Discount
1040	17502	Outdoors	0.15
2020	17446	Golf	0.07
1030	17431	Shoes	0.10
1030	17431	Clothes	0.10
	.		
	.		
3010	17303	Clothes	0.15

Read Discount Offers File – Output

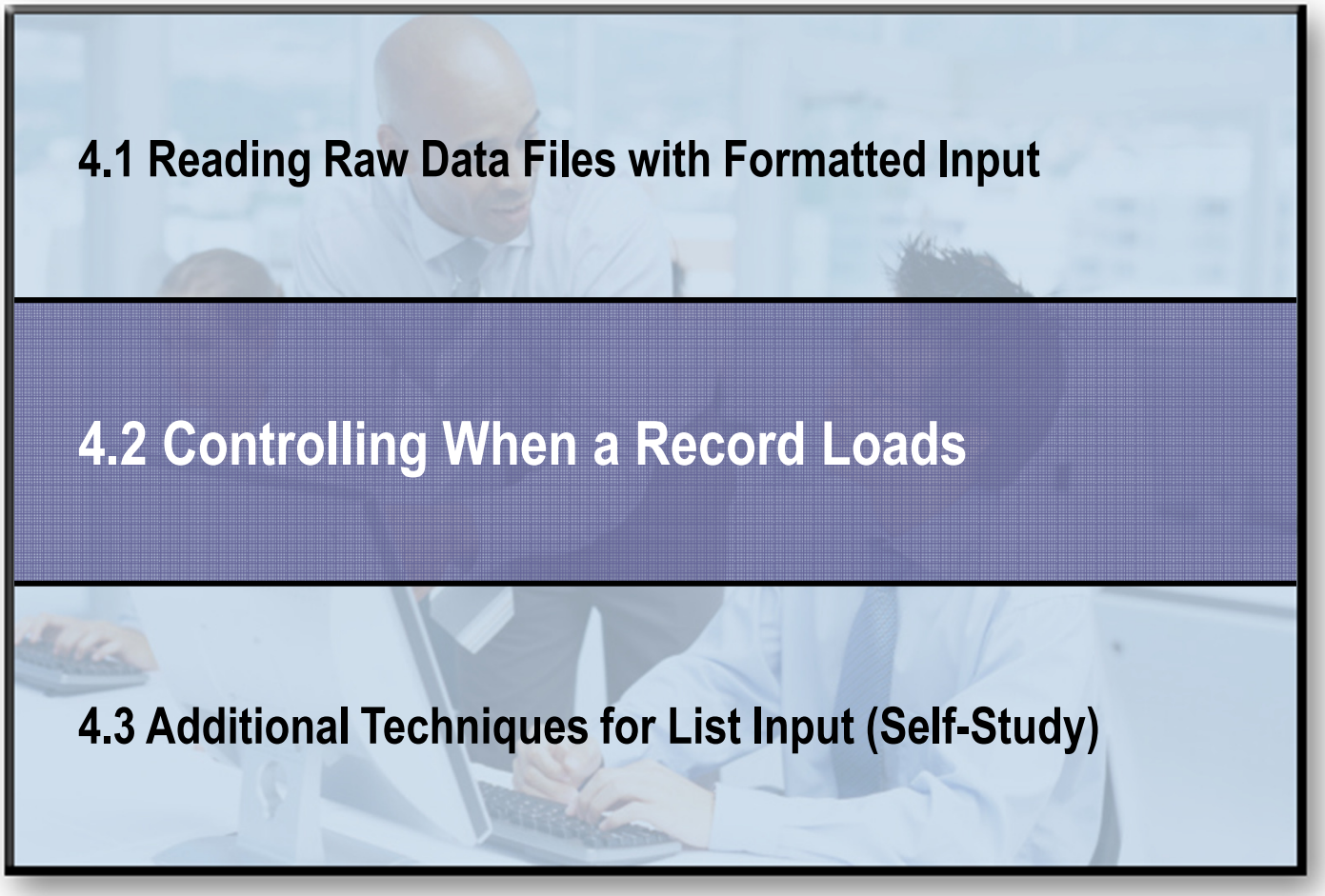
```
proc print data=work.discounts noobs;
   format Offer_dt date9.;
run;
```

Partial PROC PRINT Output

```
  Cust_
   type      Offer_dt     Item_gp      Discount

   1040     02DEC2007     Outdoors        0.15
   2020     07OCT2007     Golf            0.07
   1030     22SEP2007     Shoes           0.10
   1030     22SEP2007     Clothes         0.10
                .
                .
   3010     17MAY2007     Clothes         0.15
```

Chapter 4: Reading Raw Data Files



Objectives

- Read a raw data file with multiple records per observation.
- Read a raw data file with mixed record types.
- Subset from a raw data file with mixed record types.

Business Scenario – Read Contacts Data

The raw data file **`Address.dat`** contains name, mailing address, and phone information.

Partial **`Address.dat`**

```
          1    1    2    2
1---5----0----5----0----5
Ms. Sue Farr
15 Harvey Rd.
Macon, GA  31298
869-7008
Dr. Kay B. Cox
163 McNeil Pl.
Kern, CA  93280
483-3321
```

Create a SAS data set, **`contacts`**, that contains the name, phone, and second line of the mailing address.

The information for each person is on four lines in the raw data file.

Business Scenario – Desired Output

The **contacts** data set should have one observation per person.

Partial Listing of **contacts**

FullName	Address2	Phone
Ms. Sue Farr	Macon, GA 31298	869-7008
Dr. Kay B. Cox	Kern, CA 93280	483-3321
Mr. Ron Mason	Miami, FL 33054	589-9030
Ms. G. H. Ruth	Munger, MI 48747	754-3582

Multiple INPUT Statements

By default, SAS loads a new record into the input buffer when it encounters an INPUT statement.

You can have multiple INPUT statements in one DATA step.

```
DATA SAS-data-set;
     INFILE 'raw-data-file-name';
     INPUT specifications;
     INPUT specifications;
     <additional SAS statements>
RUN;
```

Each INPUT statement ends with a semicolon.

Multiple INPUT Statements

```
data contacts;
   infile 'address.dat';
   input FullName $30.;
   input;
   input Address2 $25.;
   input Phone $8.;
run;
```

Load first line of raw data

Partial Input Buffer

```
          1                   2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
M s .   S u e   F a r r
```

Multiple INPUT Statements

```
data contacts;
   infile 'address.dat';
   input FullName $30.;
   input;
   input Address2 $25.;
   input Phone $8.;
run;
```

Load second line of raw data

Partial Input Buffer

```
          1                   2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
1 5   H a r v e y   R d .
```

- Even though no variables are listed, the INPUT statement will still load the raw data line into the input buffer.

...

Multiple INPUT Statements

```
data contacts;
   infile 'address.dat';
   input FullName $30.;
   input;
   input Address2 $25.;
   input Phone $8.;
run;
```

Load third line of raw data

Partial Input Buffer

```
          1                   2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
M a c o n ,   G A   3 1 2 9 8
```


...

Multiple INPUT Statements

```
data contacts;
   infile 'address.dat';
   input FullName $30.;
   input;
   input Address2 $25.;
   input Phone $8.;
run;
```

Load fourth line of raw data

Partial Input Buffer

```
          1                   2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
8 6 9 - 7 0 0 8
```

Multiple INPUT Statements

Partial SAS Log

```
NOTE: 48 records were read from the infile 'address.dat'.
      The minimum record length was 18.
      The maximum record length was 30.
NOTE: The data set WORK.CONTACTS has 12 observations
      and 3 variables.
```

Multiple INPUT Statements

```
proc print data=contacts noobs;
run;
```

Partial PROC PRINT Output

FullName	Address2	Phone
Ms. Sue Farr	Macon, GA 31298	869-7008
Dr. Kay B. Cox	Kern, CA 93280	483-3321
Mr. Ron Mason	Miami, FL 33054	589-9030
Ms. G. H. Ruth	Munger, MI 48747	754-3582

Line Pointer Controls

You can also use line pointer controls to control when SAS loads a new record.

```
DATA SAS-data-set;
      INFILE 'raw-data-file-name' ;
      INPUT specifications /
        specifications;
      <additional SAS statements>
RUN;
```

SAS loads the next record when it encounters a forward slash.

Line Pointer Controls

```
data contacts;
   infile 'address.dat';
   input FullName $30. / /
         Address2 $25. /
         Phone $8. ;
run;
```

Load first line of raw data

Partial Input Buffer

```
          1                   2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
M s .   S u e   F a r r
```

Line Pointer Controls

```
data contacts;
   infile 'address.dat';
   input FullName $30. / /
         Address2 $25. /
         Phone $8. ;
run;
```

Load second line of raw data

Partial Input Buffer

```
          1                   2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
| 1 | 5 |   | H | a | r | v | e | y |   | R | d | . |   |   |   |   |   |   |   |
```


...

Line Pointer Controls

```
data contacts;
   infile 'address.dat';
   input FullName $30. / /
         Address2 $25. /
         Phone $8. ;
run;
```

Load third line of raw data

Partial Input Buffer

```
          1                   2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
M a c o n ,   G A   3 1 2 9 8
```


...

Line Pointer Controls

```
data contacts;
   infile 'address.dat';
   input FullName $30. / /
         Address2 $25. /
         Phone $8. ;
run;
```

Load fourth line of raw data

Partial Input Buffer

	1									2									
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
8	6	9	–	7	0	0	8												

Line Pointer Controls

Partial SAS Log

```
NOTE: 48 records were read from the infile 'address.dat'.
      The minimum record length was 18.
      The maximum record length was 30.
NOTE: The data set WORK.CONTACTS has 12 observations
      and 3 variables.
```

Line Pointer Controls

```
proc print data=contacts noobs;
run;
```

Partial PROC PRINT Output

FullName	Address2	Phone
Ms. Sue Farr	Macon, GA 31298	869-7008
Dr. Kay B. Cox	Kern, CA 93280	483-3321
Mr. Ron Mason	Miami, FL 33054	589-9030
Ms. G. H. Ruth	Munger, MI 48747	754-3582

4.05 Quiz

Using pen and paper, write an INPUT statement to read the data from the raw data file.

Raw Data

```
          1    1    2    2
1---5----0----5----0----5
10458Pine Mt. Sports
02/22/07 $2,405.50
00103RFG Textile Inc.
09/01/07 $1,095.30
24221Fifth Wheel Ltd.
06/04/07   $956.70
```

Line 1 Layout

Description	Column
Supplier Code	1- 5
Supplier Name	6-25

Line 2 Layout

Description	Column
Shipment Date	1- 8
Amount	10-18

- Supplier Code and Supplier Name contain character values.

Business Scenario – Read Top Sales Data

The raw data file, **sales.dat**, contains data about the largest sales made in the first quarter of 2007.

sales.dat

```
                1    1    2    2    3
1---5----0----5----0----5----0
101   USA 1-20-2007 3295.50
3034  EUR 30JAN2007 1876,30
101   USA 1-30-2007 2938.00
128   USA 2-5-2007  2908.74
1345  EUR 6FEB2007  3145,60
109   USA 3-17-2007 2789.10
```

Create a SAS data set, **salesQ1**, from the raw data in **sales.dat**.

Mixed Record Types

Not all records have the same format.

sales.dat

```
          1    1    2    2    3
1---5----0----5----0----5----0
101  USA 1-20-2007 3295.50
3034 EUR 30JAN2007 1876,30
101  USA 1-30-2007 2938.00
128  USA 2-5-2007  2908.74
1345 EUR 6FEB2007  3145,60
109  USA 3-17-2007 2789.10
```

The decimal places and commas are reversed for the U.S. and European sales figures, and the dates are represented differently.

Desired Output

Listing of **`salesQ1`**

```
Sale                    Sale
 ID       Location      Date      Amount

101         USA        17186     3295.50
3034        EUR        17196     1876.30
101         USA        17196     2938.00
128         USA        17202     2908.74
1345        EUR        17203     3145.60
109         USA        17242     2789.10
```

Mixed Record Types – First Attempt

This code is a good start to reading the mixed record types, but it gives unexpected results.

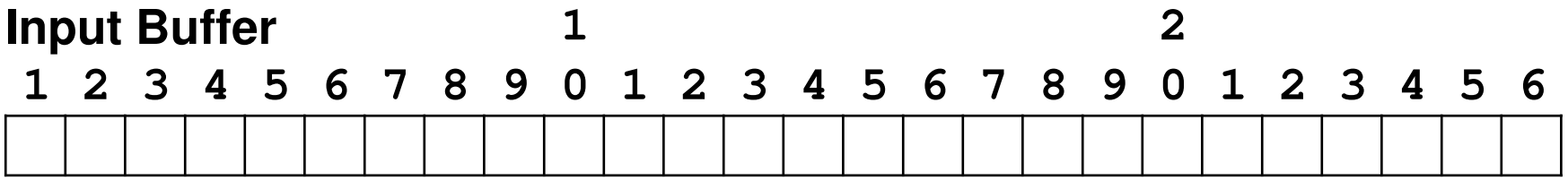
```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3.;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Execution: First Attempt

```
data salesQ1;
   infile 'sal
   input SaleI                cation $3.;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Initialize PDV

Input Buffer



PDV

SaleID	Location	SaleDate	Amount
$ 4	$ 3	N 8	N 8
		.	.

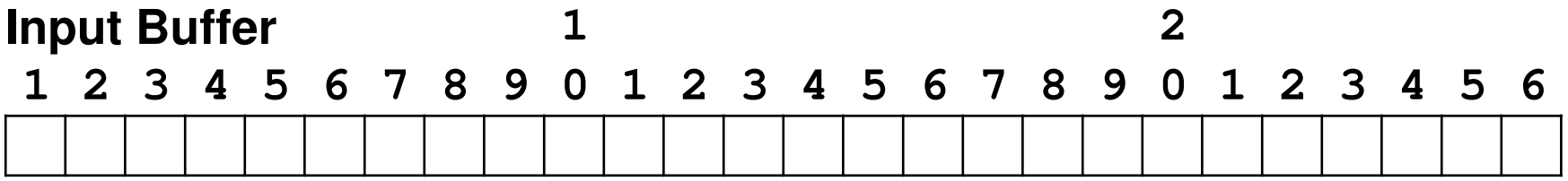
...

Execution: First Attempt

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3.;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Specify input data file

Input Buffer



PDV

SaleID	Location	SaleDate	Amount
$ 4	$ 3	N 8	N 8
		.	.

...

Execution: First Attempt

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3.;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Load the input buffer

Input Buffer

									1										2						
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
1	0	1			U	S	A		1	-	2	0	-	2	0	0	7		3	2	9	5	.	5	0

PDV

SaleID	Location	SaleDate	Amount
$ 4	$ 3	N 8	N 8
		.	.

...

Execution: First Attempt

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3.;
   if Location='USA' then
      input @10 SaleD
            @20 Amoun
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Load values into the PDV

Input Buffer

	1									2										3						
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	
1	0	1			U	S	A		1	-	2	0	-	2	0	0	7		3	2	9	5	.	5	0	

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount N 8
101	USA	.	.

...

Execution: First Attempt

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3.;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

True

Input Buffer

	1									2										3						
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
1	0	1			U	S	A		1	-	2	0	-	2	0	0	7		3	2	9	5	.	5	0

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount N 8
101	USA	.	.

...

Execution: First Attempt

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3.;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate
            @20 Amount commax7.;
run;
```

Load the input buffer

Input Buffer

	1									2															
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
3	0	3	4		E	U	R		3	0	J	A	N	2	0	0	7		1	8	7	6	,	3	0

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount N 8
101	USA	.	.

...

Execution: First Attempt

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3.;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Invalid data message written to SAS log

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
3	0	3	4		E	U	R		3	0	J	A	N	2	0	0	7		1	8	7	6	,	3	0

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount N 8
101	USA	.	.

...

Execution: First Attempt

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3.;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDat
            @20 Amount
run;
```

Implicit OUTPUT;
Implicit RETURN;

Input Buffer

	1									2															
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
3	0	3	4		E	U	R		3	0	J	A	N	2	0	0	7		1	8	7	6	,	3	0

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount N 8
101	USA	.	.

...

Execution: First Attempt

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3.;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Continue until EOF

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
3	0	3	4		E	U	R		3	0	J	A	N	2	0	0	7		1	8	7	6	,	3	0

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount N 8
101	USA	.	.

First Attempt – Unexpected Output

Partial SAS Log

```
NOTE: Invalid data for SaleDate in line 2 10-19.
NOTE: Invalid data for Amount in line 2 20-26.
RULE:     ----+----1----+----2----+----3----+----4----+----5----+
          3034 EUR 30JAN2007 1876,30
SaleID=101 Location=USA SaleDate=. Amount=. _ERROR_=1 _N_=1
          .
          .
NOTE: 6 records were read from the infile 'sales.dat'.
      The minimum record length was 26.
      The maximum record length was 27.
NOTE: The data set WORK.SALESQ1 has 3 observations and 4 variables.
```

First Attempt – Unexpected Output

```
proc print data=salesQ1 noobs;
run;
```

PROC PRINT Output

Sale ID	Location	Sale Date	Amount
101	USA	.	.
101	USA	17202	2908.74
1345	EUR	.	278910.00

To get the correct results, SAS needs some way to keep the second INPUT statement from moving to the next line of raw data.

The Single Trailing @

The single trailing @ holds a raw data record in the input buffer until SAS does one of the following:

- executes an INPUT statement with no trailing @
- begins the next iteration of the DATA step

General form of an INPUT statement with the single trailing @:

INPUT *specifications …* **@;**

Mixed Record Types – Correct Program

Adding the single trailing @ gives the correct output.

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3. @;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Partially stepping through the execution of the DATA step illustrates the effect of the trailing @.

Execution: Correct Program

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3. @;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Load the input buffer

Input Buffer

									1										2						
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
1	0	1			U	S	A		1	-	2	0	-	2	0	0	7		3	2	9	5	.	5	0

PDV

SaleID	Location	SaleDate	Amount
$ 4	$ 3	N 8	N 8
		.	.

...

Execution: Correct Program

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3. @;
   if Location='USA' then
      input @10 SaleD
            @20 Amoun
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Load values into the PDV

Input Buffer

```
          1                   2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
1 0 1     U S A   1 - 2 0 - 2 0 0 7   3 2 9 5 . 5 0
```

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount N 8
101	USA	.	.

...

Execution: Correct Program

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3. @;
   if Location='USA' then
      input @10 SaleDa
            @20 Amount
   else if Location='E
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Do not read new record at next INPUT statement

Hold

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
1	0	1			U	S	A		1	-	2	0	-	2	0	0	7		3	2	9	5	.	5	0

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount N 8
101	USA	.	.

...

Execution: Correct Program

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3. @;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

True

Hold

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
1	0	1			U	S	A		1	-	2	0	-	2	0	0	7		3	2	9	5	.	5	0

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount N 8
101	USA	.	.

...

Execution: Correct Program

```
data salesQ1;
   infile 'sales.dat';
   input SaleI                                    ;
   if Location=
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Do **not** load the input buffer.

Hold

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
1	0	1			U	S	A		1	-	2	0	-	2	0	0	7		3	2	9	5	.	5	0

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount N 8
101	USA	.	.

...

Execution: Correct Program

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3. @;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Input Buffer

	1									2														
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
1	0	1			U	S	A		1	-	2	0	-	2	0	0	7		3	2	9	5	.	5	0

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount N 8
101	USA	17186	3295.50

...

Execution: Correct Program

```
data salesQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3. @;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
run;
```

Continue until EOF

Input Buffer

	1									2										3						
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
1	0	1			U	S	A		1	-	2	0	-	2	0	0	7		3	2	9	5	.	5	0

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount N 8
101	USA	17186	3295.50

Correct Program – Output

Partial SAS Log

```
NOTE: 6 records were read from the infile 'sales.dat'.
      The minimum record length was 26.
      The maximum record length was 27.
NOTE: The data set WORK.SALESQ1 has 6 observations and 4 variables.
```

PROC PRINT Output

Sale ID	Location	Sale Date	Amount
101	USA	17186	3295.50
3034	EUR	17196	1876.30
101	USA	17196	2938.00
128	USA	17202	2908.74
1345	EUR	17203	3145.60
109	USA	17242	2789.10

Subsetting Mixed Record Types

Create a SAS data set, **EuropeQ1**, that contains only the European observations.

sales.dat

```
                1    1    2    2    3
1---5----0----5----0----5----0
101   USA 1-20-2007 3295.50
3034  EUR 30JAN2007 1876,30
101   USA 1-30-2007 2938.00
128   USA 2-5-2007  2908.74
1345  EUR 6FEB2007  3145,60
109   USA 3-17-2007 2789.10
```

Desired Output

Listing of **EuropeQ1**

Sale ID	Location	Sale Date	Amount
3034	EUR	17196	1876.3
1345	EUR	17203	3145.6

Adding a subsetting IF statement to the SAS program from the previous example produces this output.

4.06 Quiz

Is this the best placement for the subsetting IF statement?

```
data EuropeQ1;
   infile 'sales.dat';
   input SaleID $4. @6 Location $3. @;
   if Location='USA' then
      input @10 SaleDate mmddyy10.
            @20 Amount 7.;
   else if Location='EUR' then
      input @10 SaleDate date9.
            @20 Amount commax7.;
   if Location = 'EUR';
run;
```

Placement of the Subsetting IF Statement

Generally, the most efficient place to put the subsetting IF statement is as soon as all the variables that are needed to evaluate the condition are assigned values.

```
data EuropeQ1;
  infile 'sales.dat';
  input @6 Location $3. @;
  if Location = 'EUR';
  input  @1 SaleID $4.
         @10 SaleDate date9.
         @20 Amount commax7.;
run;
```

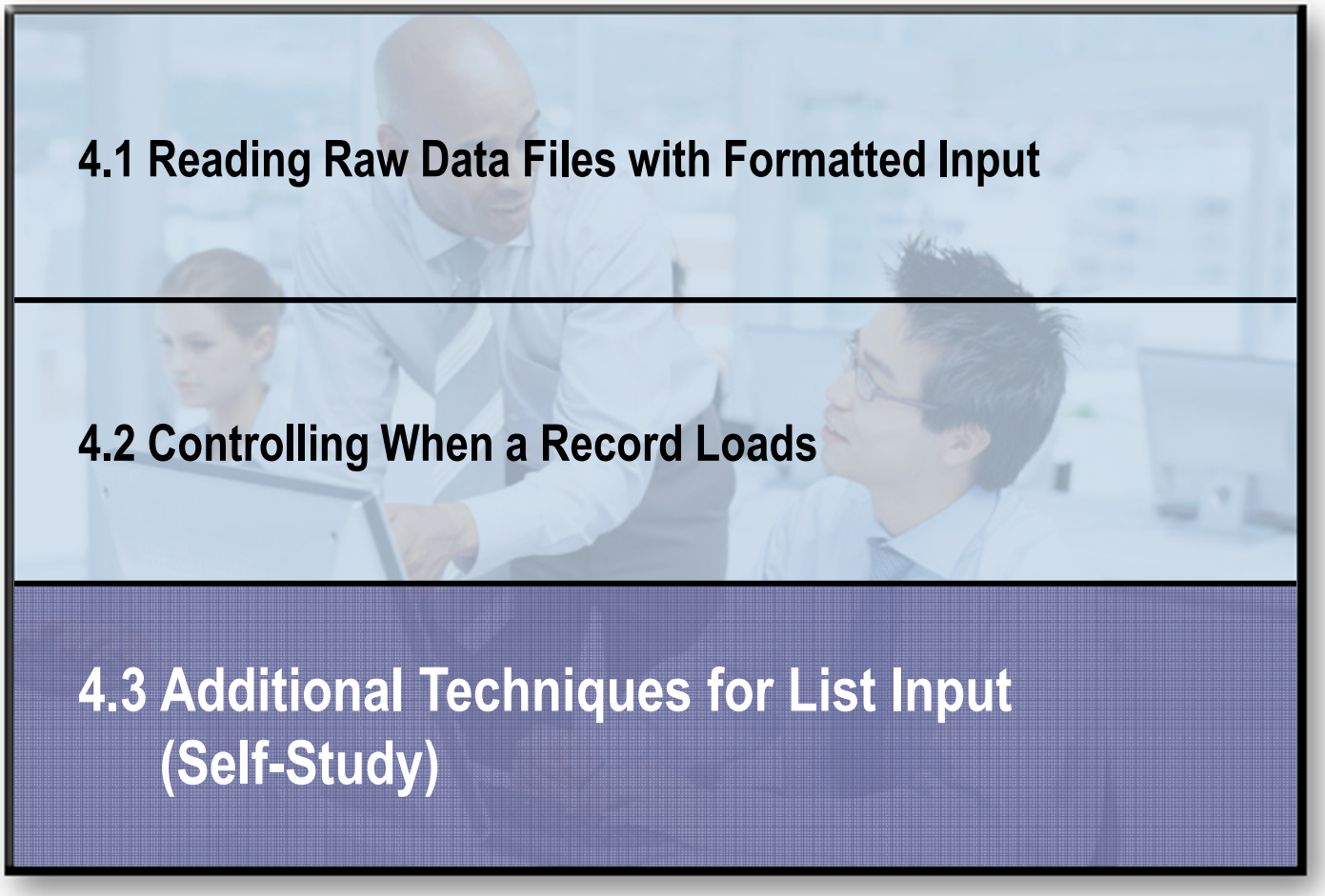
Subsetting Mixed Record Types – Output

```
proc print data=EuropeQ1 noobs;
   var SaleID Location SaleDate Amount;
run;
```

PROC PRINT Output

Sale ID	Location	Sale Date	Amount
3034	EUR	17196	1876.3
1345	EUR	17203	3145.6

Chapter 4: Reading Raw Data Files



Objectives

Read delimited raw data files with any of these special characteristics:

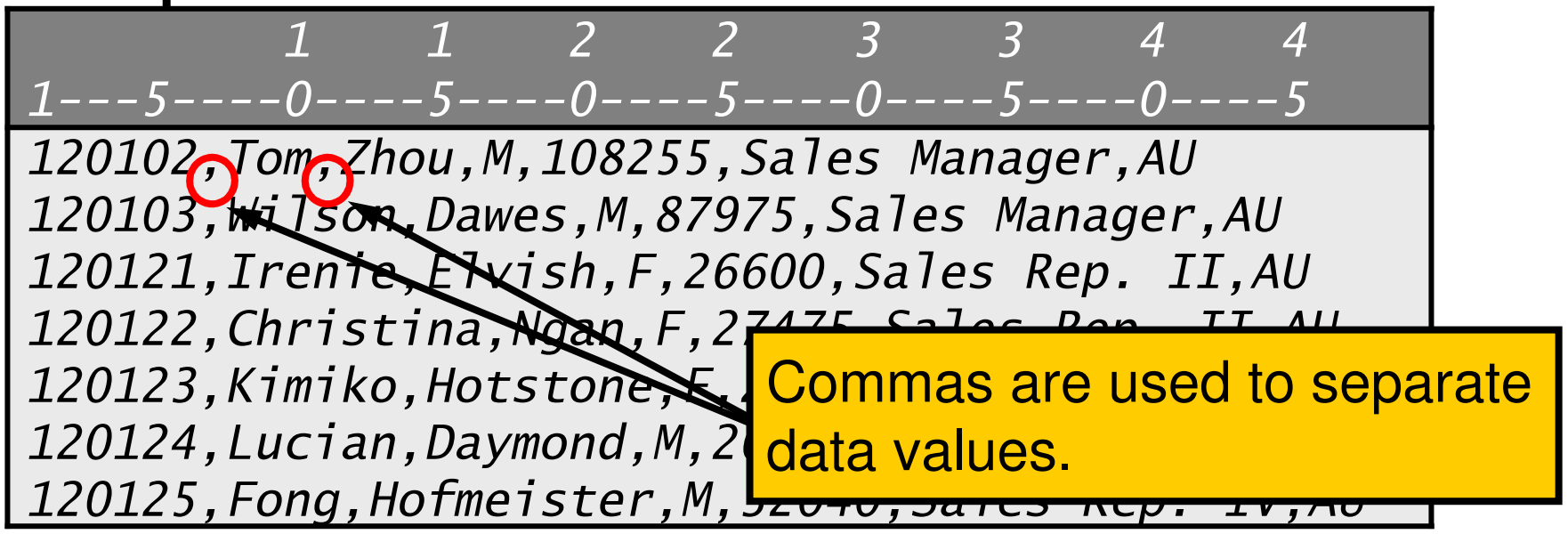
- missing data at the end of a record
- missing data represented by consecutive delimiters
- multiple observations per record

Reading Delimited Data Files (Review)

Characteristics of delimited data files:

- Data values are separated by a delimiting character.
- The delimiting character can be a blank, tab, comma, or any other character.

Example: Delimited Raw Data File



Specifying the Delimiter (Review)

A blank is the default delimiter.

The DLM= options can be added to the INFILE statement to specify an alternative delimiter.

General DATA step syntax to use list input for reading comma-delimited files:

```
DATA output-SAS-data-set;
      LENGTH variable(s) $ length;
      INFILE 'raw-data-file-name' DLM=',';
      INPUT variable-1 <$>
            <variable-2 <$>…variable-n>;
      <additional SAS statements>
RUN;
```

Additional Techniques for List Input

Additional techniques are needed if the raw data file has any of the following special characteristics:

- missing data at the end of a record
- missing data represented by consecutive delimiters
- multiple observations per record

Each of these special cases is explored through separate business scenarios.

Business Scenario – Read Contact Data

The raw data file **phone.csv** contains contact names and phone numbers for Orion customers.

Create a new SAS data set, **contacts**, by reading the raw data file.

phone.csv

```
          1    1    2    2    3    3    4    4
1---5----0----5----0----5----0----5----0----5
James Kvarniq,(704) 293-8126,(701) 281-8923
Sandrina Stephano,(919) 871-7830
Cornelia Krahl,(212) 891-3241,(212) 233-5413
Karen Ballinger,(714) 344-4321
Elke Wallstab,(910) 763-5561,(910) 545-3421
```

Missing Values at the End of a Record

The data values in **`phone.csv`** are separated by commas. Each record has a contact name, then a phone number, and finally a mobile number.

`phone.csv`

The mobile number is missing from some of the lines of data.

```
          1         2         3         4
1---5----0----5----0----5----0----5----0----5
James Kvarniq,(704) 293-8126,(701) 281-8923
Sandrina Stephano,(919) 871-7830
Cornelia Krahl,(212) 891-3241,(212) 233-5413
Karen Ballinger,(714) 344-4321
Elke Wallstab,(910) 763-5561,(910) 545-3421
```

4.07 Quiz

Open and submit **p204a01**. Examine the SAS log. How many input records were read and how many observations were created?

```
data contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile 'phone.csv' dlm=',';
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts noobs;
run;
```

p204a01

Unexpected Results

The missing mobile phone numbers have caused unexpected results in the output.

PROC PRINT output

Name	Phone	Mobile
James Kvarniq	(704) 293-8126	(701) 281-8923
Sandrina Stephano	(919) 871-7830	Cornelia Krahl
Karen Ballinger	(714) 344-4321	Elke Wallstab

Partial SAS Log

```
NOTE: 5 records were read from the infile 'phone.csv'.
      The minimum record length was 31.
      The maximum record length was 44.
NOTE: SAS went to a new line when INPUT statement
reached past the end of a line.
NOTE: The data set WORK.CONTACTS has 3 observations and
3 variables.
```

Missing Values at the End of a Record

By default, when there is missing data at the end of a row, SAS does the following:

- loads the next record to finish the observation
- writes a note to the log

The MISSOVER Option

The MISSOVER option prevents SAS from loading a new record when the end of the current record is reached.

General form of an INFILE statement with a MISSOVER option:

INFILE *'raw-data-file'* MISSOVER**;**

If SAS reaches the end of the row without finding values for all fields, variables without values are set to missing.

4.08 Quiz

Open **p204a02** and add the MISSOVER option to the INFILE statement. Submit the program and examine the SAS log. How many input records were read and how many observations were created?

```
data contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile 'phone.csv' dlm=',';
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts noobs;
run;
```

p204a02

Results

Adding the MISSOVER option gives the expected results.

PROC PRINT Output

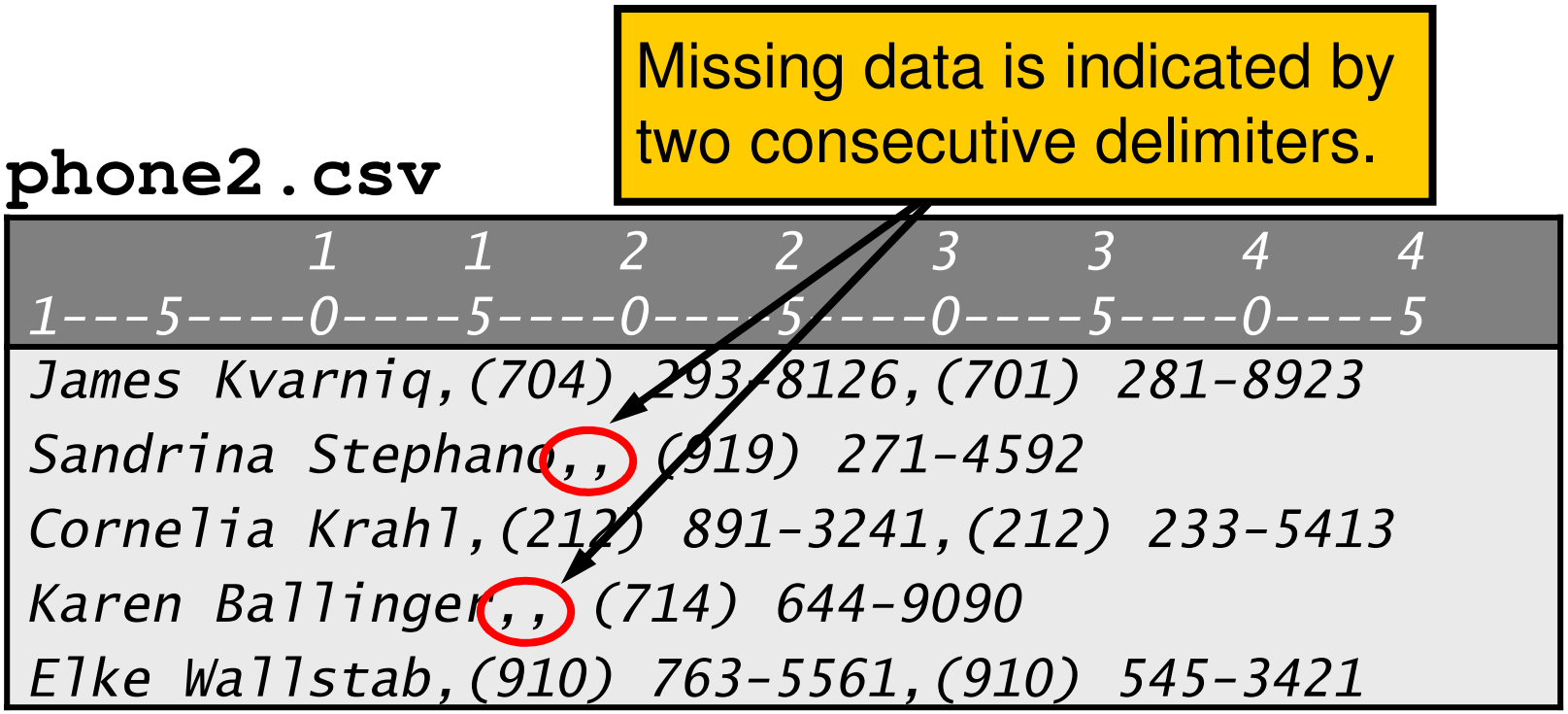
Name	Phone	Mobile
James Kvarniq	(704) 293-8126	(701) 281-8923
Sandrina Stephano	(919) 871-7830	
Cornelia Krahl	(212) 891-3241	(212) 233-5413
Karen Ballinger	(714) 344-4321	
Elke Wallstab	(910) 763-5561	(910) 545-3421

Partial SAS Log

```
NOTE: 5 records were read from the infile 'phone.csv'.
      The minimum record length was 31.
      The maximum record length was 44.
NOTE: The data set WORK.CONTACTS has 5 observations and
3 variables.
```

Missing Values before the End of the Record

Each record in **`phone2.csv`** has a contact name, phone number, and a mobile number. The phone number is missing from some of the records.



4.09 Quiz

Open and submit **p204a03**. Examine the SAS log. How many input records were read and how many observations were created?

```
data contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile 'phone2.csv' dlm=',';
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts noobs;
run;
```

p204a03

Unexpected Results

The missing phone numbers have caused unexpected results in the output.

PROC PRINT Output

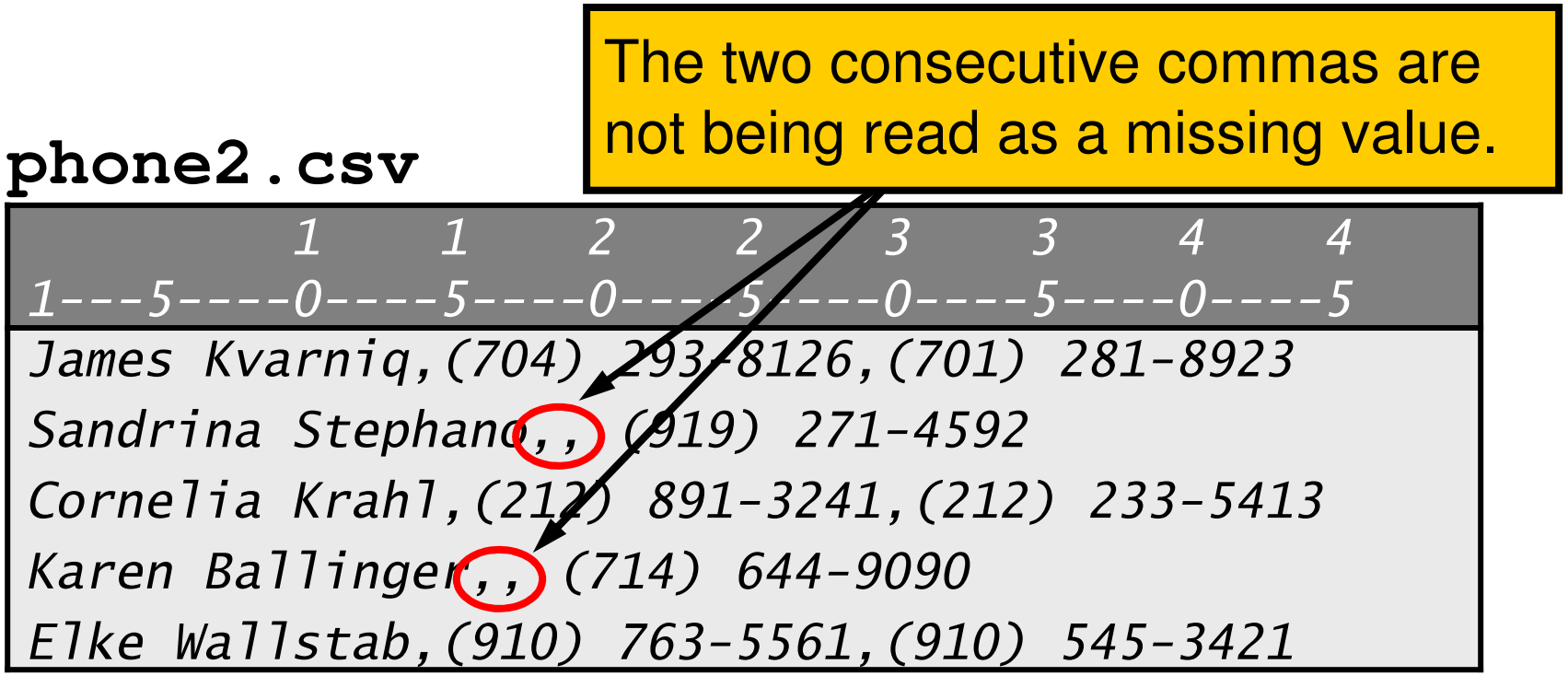
Name	Phone	Mobile
James Kvarniq	(704) 293-8126	(701) 281-8923
Sandrina Stephano	(919) 271-4592	Cornelia Krahl
Karen Ballinger	(714) 644-9090	Elke Wallstab

Partial SAS Log

```
NOTE: 5 records were read from the infile 'phone2.csv'.
      The minimum record length was 31.
      The maximum record length was 44.
NOTE: SAS went to a new line when INPUT statement reached
past the end of a line.
NOTE: The data set WORK.CONTACTS has 3 observations and 3
variables.
```

Consecutive Delimiters in List Input

By default, list input treats two or more consecutive delimiters as a single delimiter and they are not treated as a missing value.



The DSD Option

The DSD option for the INFILE statement

- sets the default delimiter to a comma
- treats consecutive delimiters as missing values
- enables SAS to read values with embedded delimiters if the value is surrounded by quotation marks.

General form of a DSD option in an INFILE statement:

INFILE '*file-name*' DSD;

Using the DSD Option

Adding the DSD option will correctly read the **`phone2.csv`** data file.

```
data contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile 'phone2.csv' dsd;
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts noobs;
run;
```

- The DLM=',' option is no longer needed in the INFILE statement because the DSD option sets the default delimiter to a comma.

Results

Adding the DSD option gives the expected results.

PROC PRINT Output

Name	Phone	Mobile
James Kvarniq	(704) 293-8126	(701) 281-8923
Sandrina Stephano		(919) 271-4592
Cornelia Krahl	(212) 891-3241	(212) 233-5413
Karen Ballinger		(714) 644-9090
Elke Wallstab	(910) 763-5561	(910) 545-3421

Partial SAS Log

```
NOTE: 5 records were read from the infile 'phone2.csv'.
      The minimum record length was 31.
      The maximum record length was 44.
NOTE: The data set WORK.CONTACTS has 5 observations and
3 variables.
```

Business Scenario – Read Charity Donations

The raw data file **charity.dat** contains data about donations made in 2007. The information for each donation consists of a charity ID and an amount.

Create a SAS data set, **donate07**, from the raw data in **charity.dat**.

charity.dat

```
          1    1    2    2    3
1---5----0----5----0----5----0
 AQI  495  CCI  200  CNI 249
CS  279  CU  780   DAI
875 ES  290  FFC 0  MI 745
SBA 900 V2 550 YYCR 0
```

Business Scenario – Desired Output

The output SAS data set should have one observation per donation.

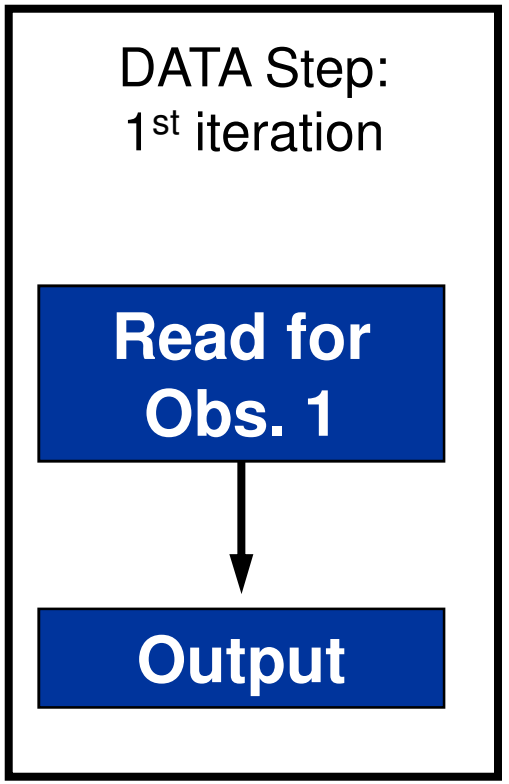
Partial Listing of **donate07**

ID	Amount
AQI	495
CCI	200
CNI	249
CS	279
CU	780
DAI	875

Processing: What Is Required?

charity.dat

```
                1    1    2    2    3
1---5----0----5----0----5----0----5----0
 AQI  495  CCI  200  CNI 249
CS  279  CU  780   DAI
875 ES  290  FFC 0  MI 745
SBA 900 V2 550 YYCR 0
```



Each raw data line contains information for multiple donations.

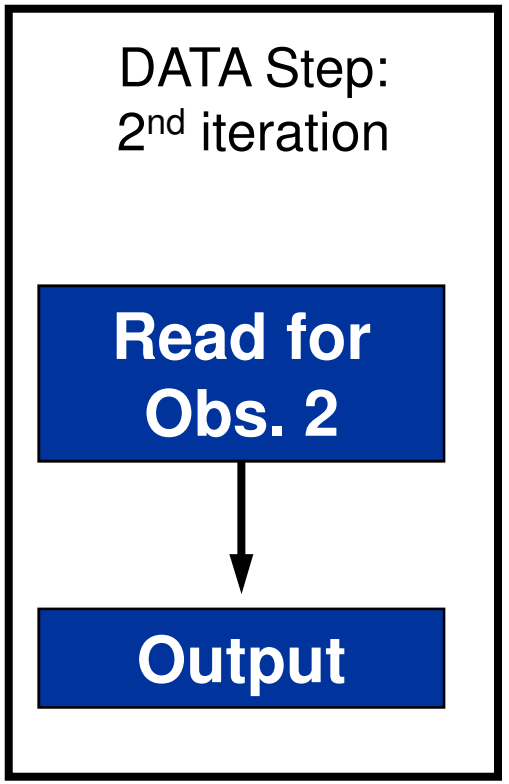
Each iteration of the DATA step must read data for one donation.

...

Processing: What Is Required?

charity.dat

```
          1    1    2    2    3
1---5----0----5----0----5----0
 AQI  495  CCI  200  CNI 249
CS  279  CU  780  DAI
875 ES  290  FFC 0  MI 745
SBA 900 V2 550 YYCR 0
```



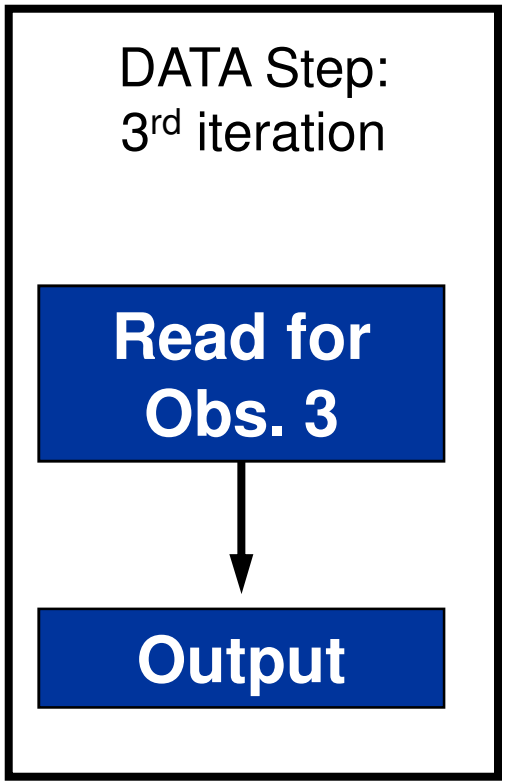
To do this kind of processing, SAS needs to use the same raw data line in several iterations of the DATA step.

...

Processing: What Is Required?

charity.dat

```
                1    1    2    2    3
1---5----0----5----0----5----0----5----0
 AQI  495  CCI  200  CNI 249
CS  279  CU  780   DAI
875 ES  290  FFC 0  MI 745
SBA 900 V2 550 YYCR 0
```



...

The Double Trailing @

The *double trailing @* holds the raw data record across iterations of the DATA step until the line pointer moves past the end of the line.

INPUT *var1 var2 var3 … @@*;

The double trailing @ should only be used with list input.

The Double Trailing @

```
data donate07;
   length ID $ 4;
   infile 'charity.dat';
   input ID $ Amount @@;
run;
```

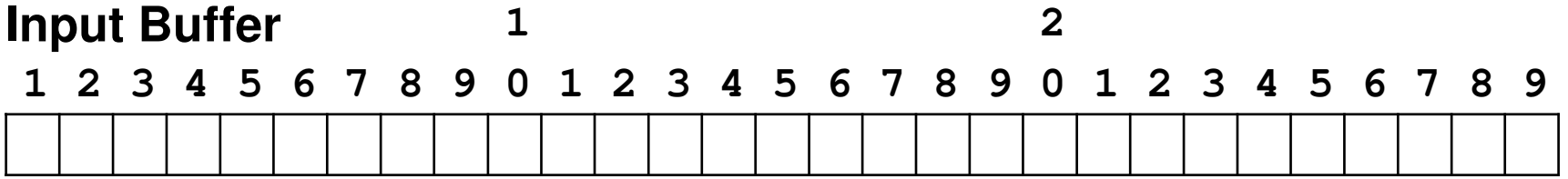
Stepping through the execution of the program will illustrate how the double trailing @ holds the raw data record across iterations of the DATA step.

Execution: The Double Trailing @

```
data donate07;
   length ID $ 4;
   infile 'charity.dat';
   input ID $ Amount @@;
run;
```

Initialize PDV

Input Buffer



PDV

ID	Amount
$ 4	N 8
	.

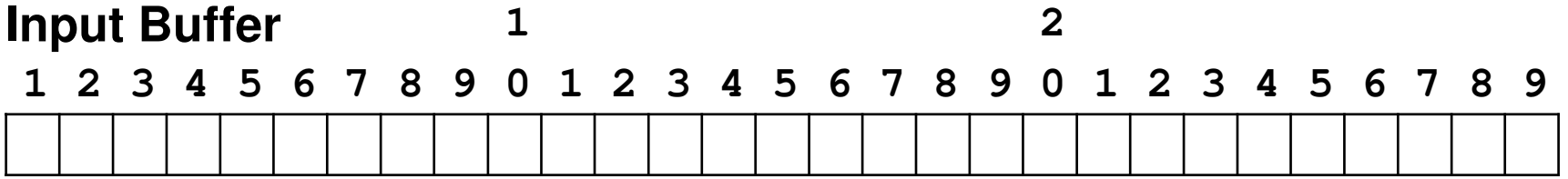
...

Execution: The Double Trailing @

```
data donate07;
    length ID $ 4;
    infile 'charity.dat';
    input ID $ Amount @@;
run;
```

Specify input data file

Input Buffer



PDV

ID	Amount
$ 4	N 8

...

Execution: The Double Trailing @

```
data donate07;
   length ID $ 4;
   infile 'charity.dat';
   input  ID $ Amount @@;
run;
```

Load input buffer

Input Buffer

	1									2																
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
	A	Q	I			4	9	5			C	C	I			2	0	0			C	N	I		2	4	9	

PDV

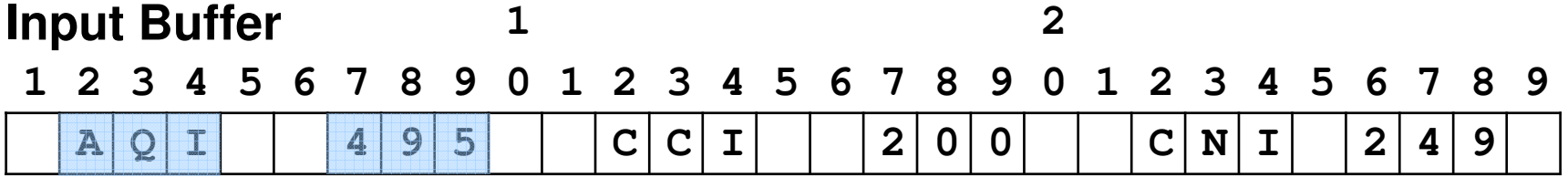
ID	Amount
$ 4	N 8

...

Execution: The Double Trailing @

```
data donate07;
   length ID $ 4;
   infile 'charity.dat';
   input ID $ Amount @@;
run;
```

Input Buffer



PDV

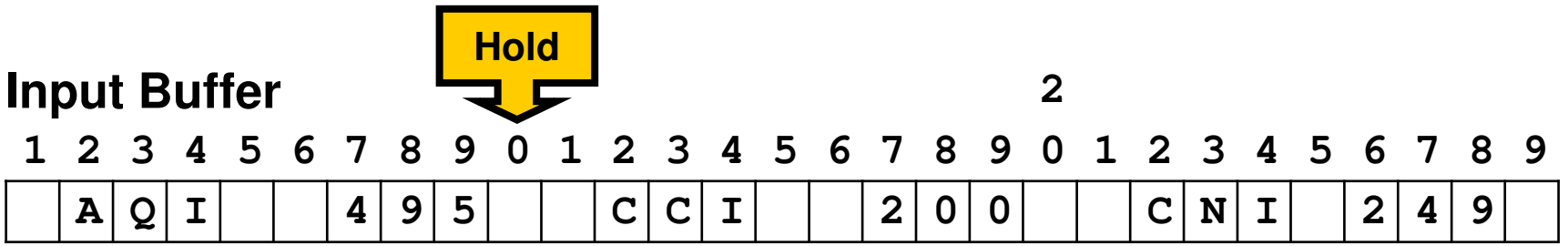
ID $ 4	Amount N 8
AQI	495

...

Execution: The Double Trailing @

```
data donate07;
   length ID $ 4;
   infile 'charity.dat';
   input ID $ Amount @@;
run;
```

Hold record in the input buffer



PDV

ID $ 4	Amount N 8
AQI	495

...

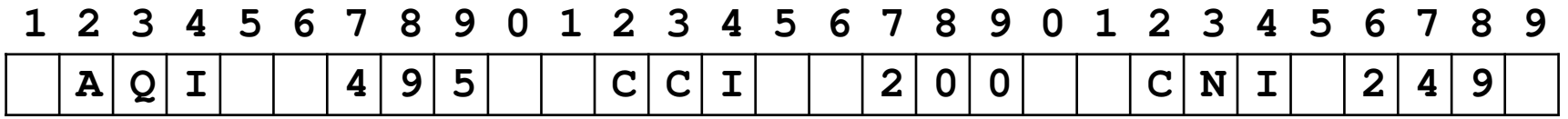
Execution: The Double Trailing @

```
data donate07;
   length ID $ 4;
   infile 'charity.dat';
   input ID $ Amount @@;
run;
```

Implicit OUTPUT;
Implicit RETURN;

Hold

Input Buffer



PDV

ID $ 4	Amount N 8
AQI	495

...

Execution: The Double Trailing @

```
data donate07;
    length ID $ 4;
    infile 'charity.dat';
    input ID $ Amount @@;
run;
```

Reinitialize PDV

Hold

Input Buffer

	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	
		A	Q	I			4	9	5			C	C	I			2	0	0			C	N	I		2	4	9		

PDV

ID	Amount
$ 4	N 8
	.

...

Execution: The Double Trailing @

```
data donate07;
   length ID $ 4;
   infile 'charity.dat';
   input ID $ Amount @@;
run;
```

Specify input data file

Hold

Input Buffer

```
                                                                  2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
  A Q I     4 9 5     C C I     2 0 0     C N I     2 4 9
```

PDV

ID $ 4	Amount N 8

...

Execution: The Double Trailing @

```
data donate07;
   length ID $ 4;
   infile 'charity.dat';
   input ID $ Amount @@;
run;
```

Do *not* load input buffer

Hold

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
	A	Q	I			4	9	5			C	C	I			2	0	0			C	N	I		2	4	9	

PDV

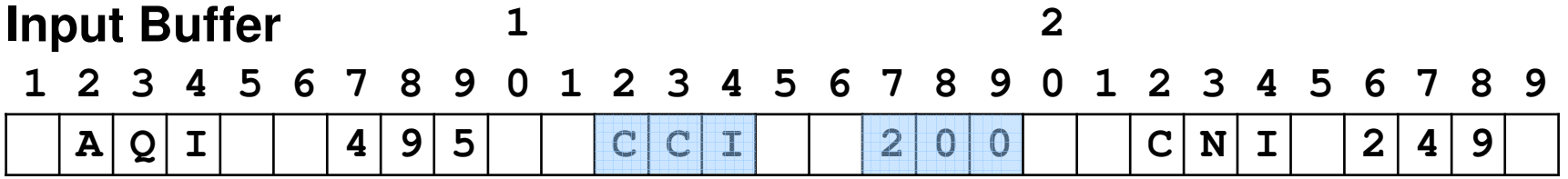
ID	Amount
$ 4	N 8

...

Execution: The Double Trailing @

```
data donate07;
   length ID $ 4;
   infile 'charity.dat';
   input ID $ Amount @@;
run;
```

Input Buffer



PDV

ID $ 4	Amount N 8
CCI	200

...

Execution: The Double Trailing @

```
data donate07;
   length ID $ 4;
   infile 'charity.dat';
   input ID $ Amount @@;
run;
```

Hold record in the input buffer

Hold

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
	A	Q	I			4	9	5			C	C	I			2	0	0			C	N	I		2	4	9	

PDV

ID $ 4	Amount N 8
CCI	200

...

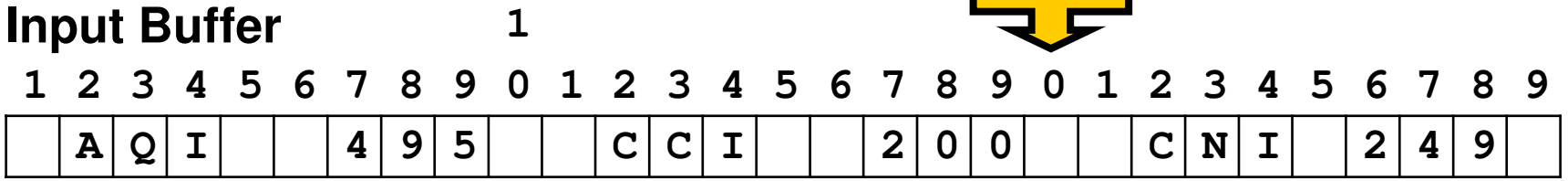
Execution: The Double Trailing @

```
data donate07;
   length ID $ 4;
   infile 'charity.dat';
   input ID $ Amount @@;
run;
```

Continue until EOF

Hold

Input Buffer



PDV

ID $ 4	Amount N 8
CCI	200

Creating Multiple Observations per Record

Partial SAS Log

```
NOTE: 4 records were read from the infile 'charity.dat'.
      The minimum record length was 23.
      The maximum record length was 28.
NOTE: SAS went to a new line when INPUT statement reached
past the end of a line.
NOTE: The data set WORK.DONATE07 has 12 observations and 2
variables.
```

The **SAS went to a new line** message is expected when a DATA step uses a double trailing @.

Creating Multiple Observations per Record

```
proc print data=donate07 noobs;
run;
```

Partial PROC PRINT Output

ID	Amount
AQI	495
CCI	200
CNI	249
CS	279
CU	780
DAI	875

Single Trailing @ versus Double Trailing @

Option	Effect
@	Holds raw data record until ■ an INPUT statement with no trailing @ or ■ the next iteration of the DATA step.
@@	Holds raw data record in the input buffer until SAS reads past the end of the line.

Chapter Review

1. What style of INPUT statement specification is used to read data in fixed columns?

2. In the INPUT statement, what is the symbol used to move the input pointer to a specified column?

3. What two things does the informat specify when used with formatted input?

continued...

Chapter Review

4. What does a forward slash (/) indicate when used in an INPUT statement?

5. In the INPUT statement, what does a trailing @ specify?

6. Generally, where is the most efficient place to put a subsetting IF statement?