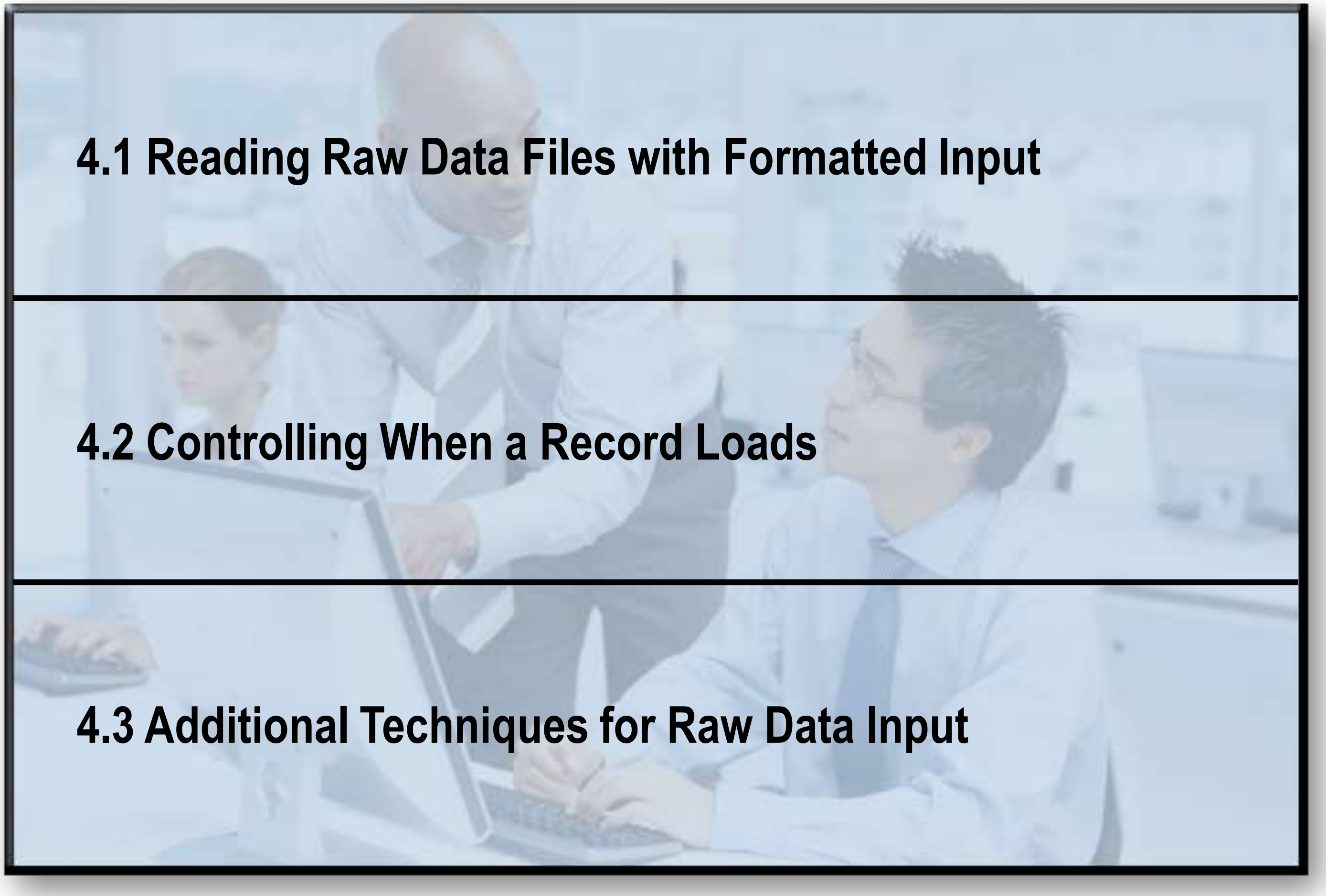
STAT604

Lesson SAS 15

Chapter 4: Reading Raw Data Files

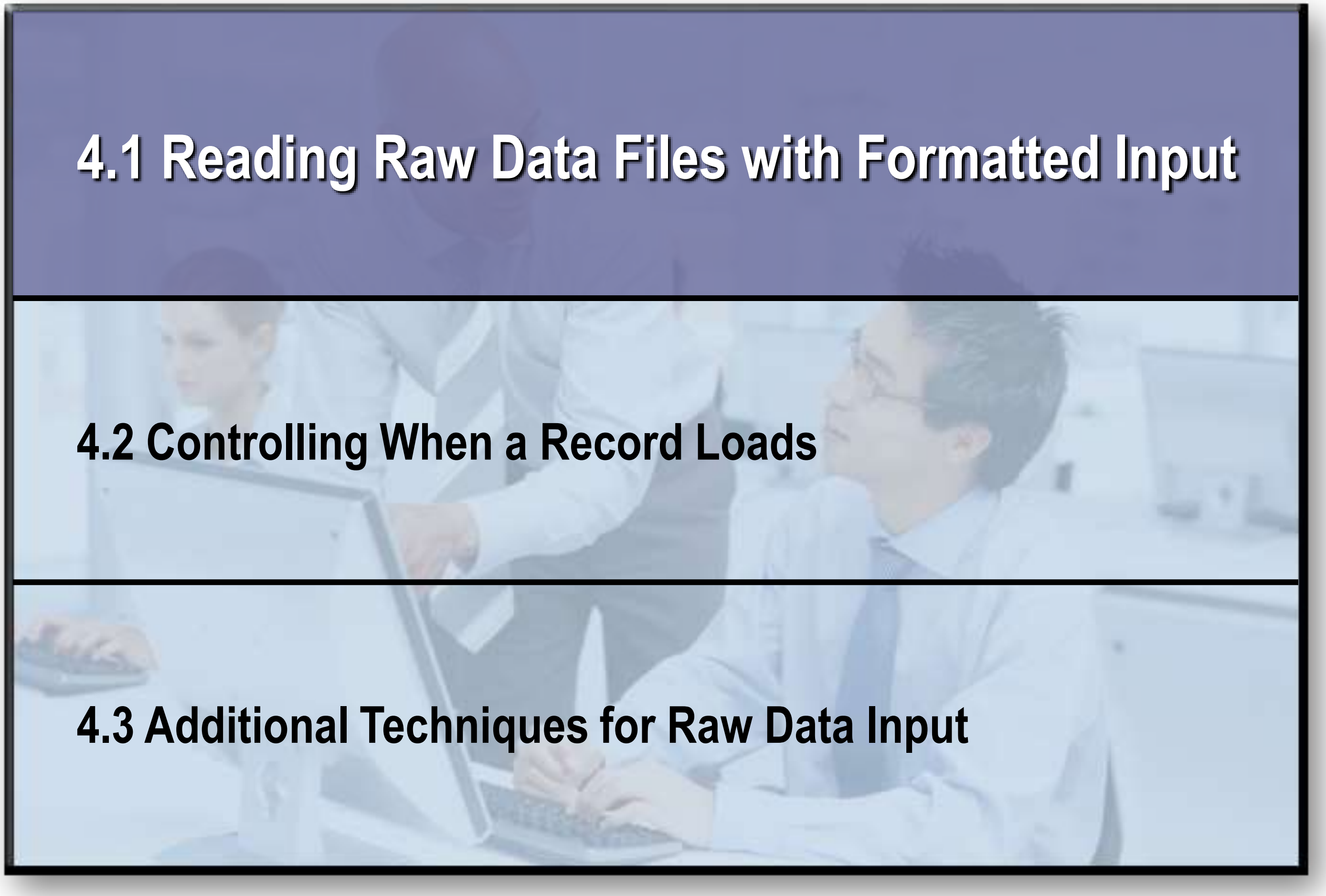


4.1 Reading Raw Data Files with Formatted Input

4.2 Controlling When a Record Loads

4.3 Additional Techniques for Raw Data Input

Chapter 4: Reading Raw Data Files



4.1 Reading Raw Data Files with Formatted Input

4.2 Controlling When a Record Loads

4.3 Additional Techniques for Raw Data Input

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
```

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;
```

Poll
Quiz

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 …**
      - ⇨ **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?

4.02 Quiz – Correct Answer

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 …**
      - ⇨ **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?

List, column, and formatted 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

Poll
Quiz

4.03 Quiz

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

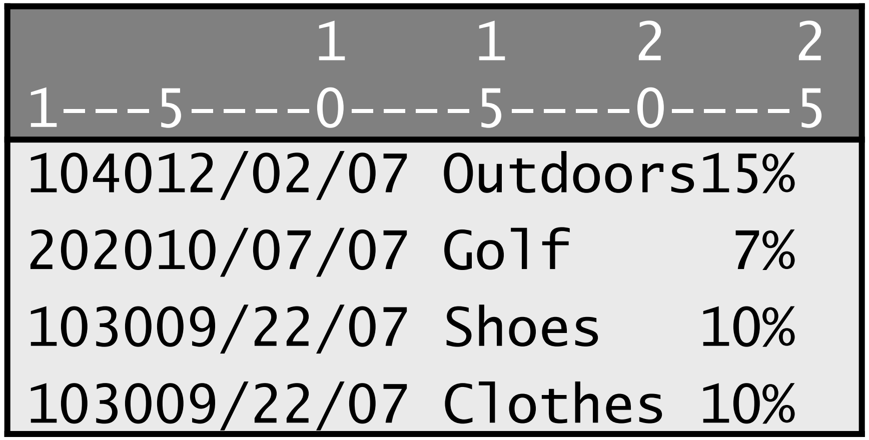
Partial offers.dat

```
         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%
```

4.03 Quiz – Correct Answer

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%
```

***Formatted input* is the best style of INPUT statement specification to read this data.**

The `offers.dat` file is in fixed columns and has nonstandard data.

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
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%
```

Poll
Quiz

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
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%
```

4.04 Quiz – Correct Answer

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

```
input @1 Cust_type 4.
      @5 Offer_dt mmddyy8.
```

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%
```

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;
```

p204d01

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

									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

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;
```

Initialize 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

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;
```

Specify input data file

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

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 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	Offer_dt	Item_gp	Discount
N 8	N 8	$ 8	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

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

Read Discount Offers File – Output

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

Partial PROC PRINT Output

SAS Date Values

```
   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

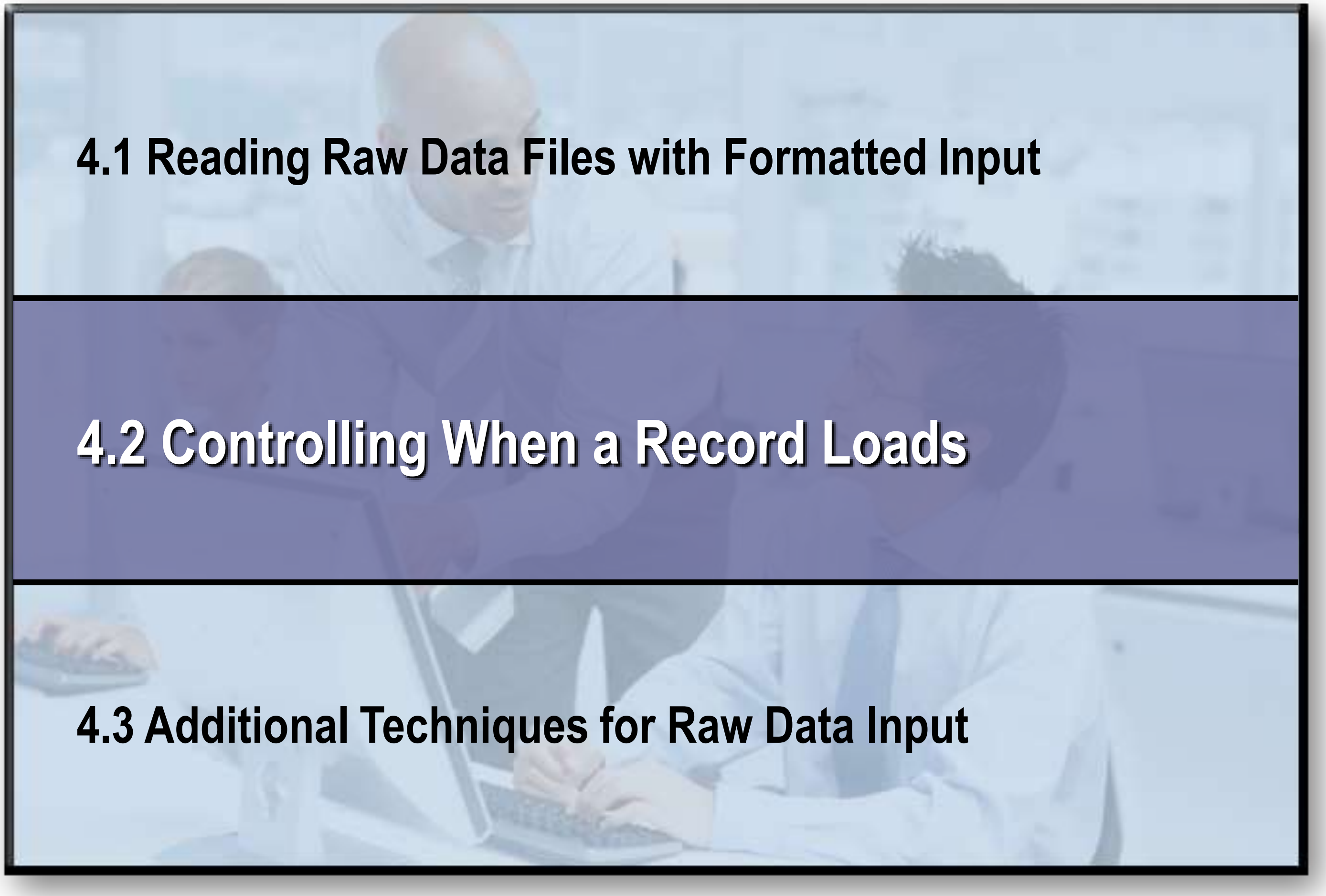
```
proc print data=work.discounts noobs;
   format Offer_dt date9.;
run;
```

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



4.1 Reading Raw Data Files with Formatted Input

4.2 Controlling When a Record Loads

4.3 Additional Techniques for Raw Data Input

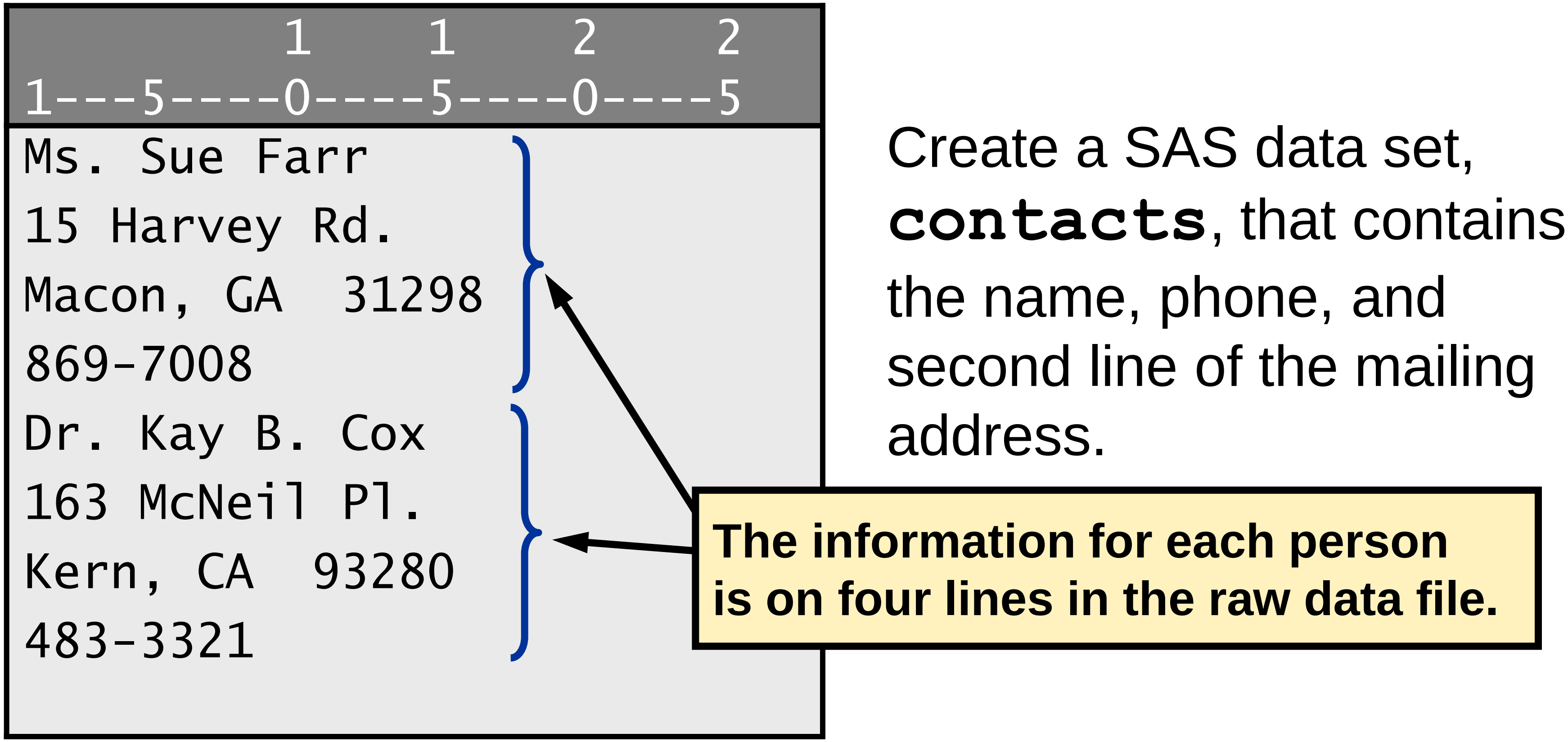
Objectives

- 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
```

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

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

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								

p204d02

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								

@1 is default and not required on INPUT statement

p204d03

...

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
```

Absolute Line Pointer Controls

```
data contacts;
   infile 'address.dat';
   input #1 FullName $30.
         #3 Address2 $25.
         #4 Phone $8. ;
run;
```

An *absolute* line pointer control moves the pointer to a specific line in a group of lines.

p204d03

Poll

Quiz

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.

4.05 Quiz – Correct Answer

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

One answer is shown here:

```
input @1 Supplier_Code $5.
      @6 Supplier_Name $20. /
      @1 Ship_Date mmddyy8.
      @10 Amount dollar9.;
```

There are other ways to correctly write the INPUT statement.

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;
```

p204d04

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

									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

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount 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

									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

PDV

SaleID $ 4	Location $ 3	SaleDate N 8	Amount 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 $ 4	Location $ 3	SaleDate N 8	Amount 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						
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       tion $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						
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 SaleDate
            @20 Amount c
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						
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	Location	SaleDate	Amount
$ 4	$ 3	N 8	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                  Sale
       ID     Location      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 @.

p204d05

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 $ 4	Location $ 3	SaleDate N 8	Amount 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	Location	SaleDate	Amount
$ 4	$ 3	N 8	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						
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        ation $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						
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						
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	Location	SaleDate	Amount
$ 4	$ 3	N 8	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						
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                    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
```

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                   Sale
    ID     Location       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.

Poll
Quiz

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;
```

p204d06

4.06 Quiz – Correct Answer

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;
```

No, the subsetting IF statement should appear as early in the DATA step as possible.

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;
```

p204d07

Subsetting Mixed Record Types – Output

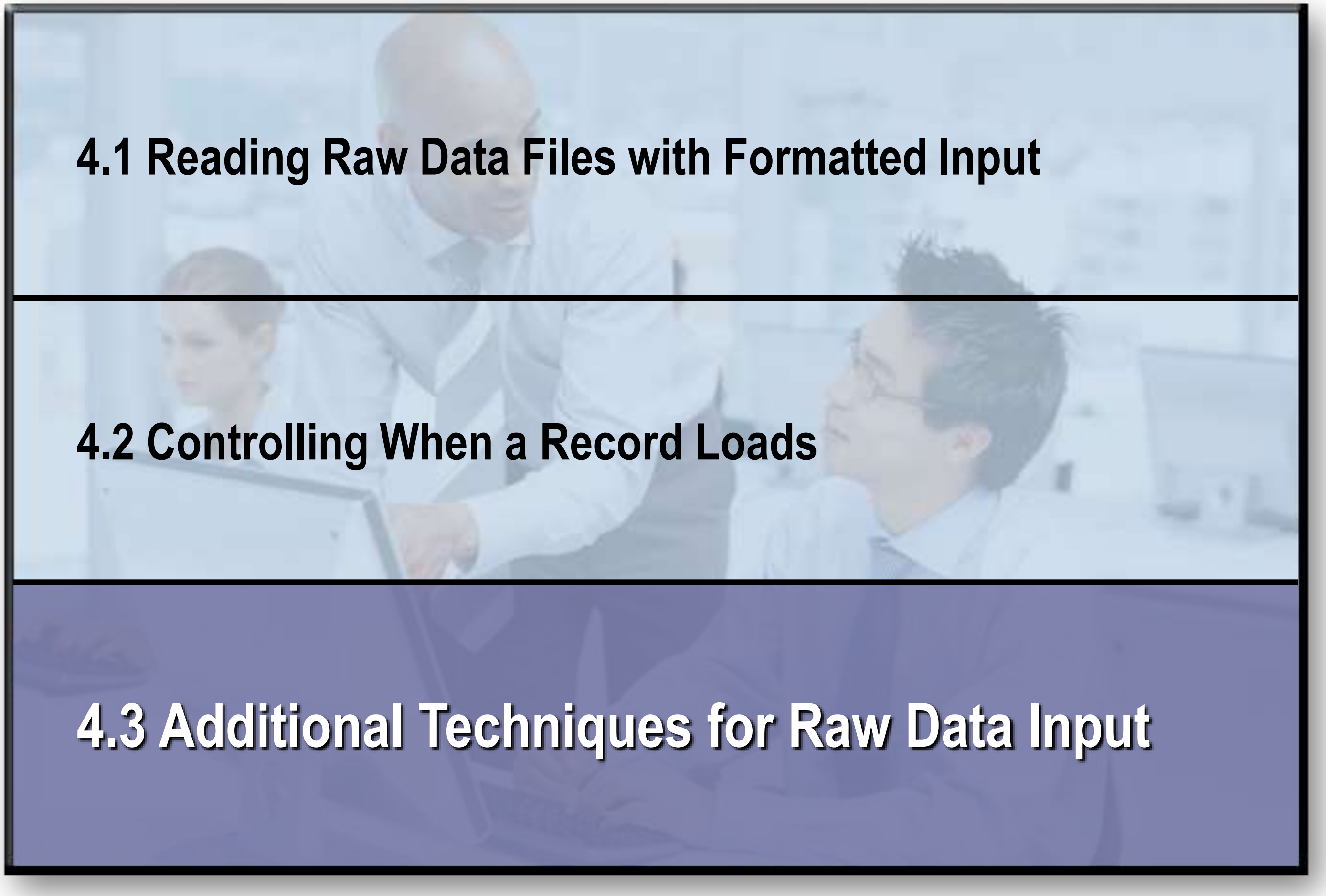
```
proc print data=EuropeQ1 noobs;
   var SaleID Location SaleDate Amount;
run;
```

PROC PRINT Output

```
   Sale                    Sale
    ID      Location       Date     Amount

   3034       EUR         17196     1876.3
   1345       EUR         17203     3145.6
```

Chapter 4: Reading Raw Data Files



4.1 Reading Raw Data Files with Formatted Input

4.2 Controlling When a Record Loads

4.3 Additional Techniques for Raw Data Input

Objectives

Read raw data files with any of these special characteristics:

- delimited data with multiple observations per record
- records that are shorter than specified
- long records that exceed the default length

Additional Techniques for Raw Data Input

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

- list input with multiple observations per record
- records that are shorter than the specified length
- records that are longer than the default

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
```

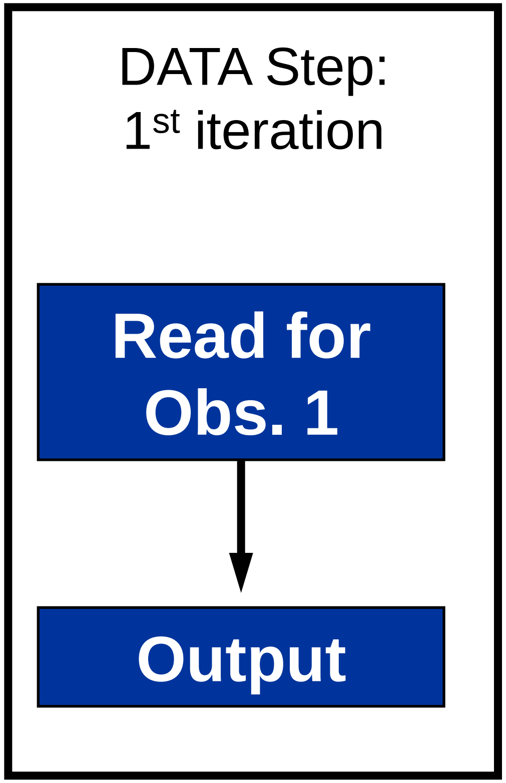
Processing: What Is Required?

charity.dat

```
         1    1    2    2    3
1---5----0----5----0----5----0
 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.



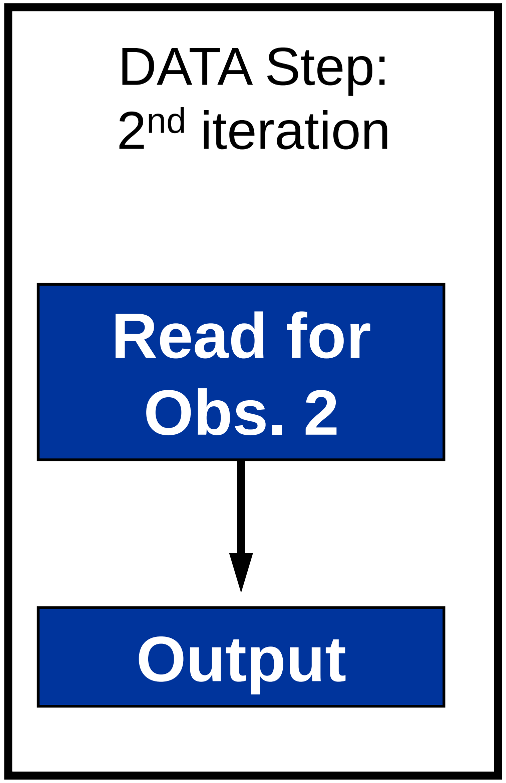
...

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.

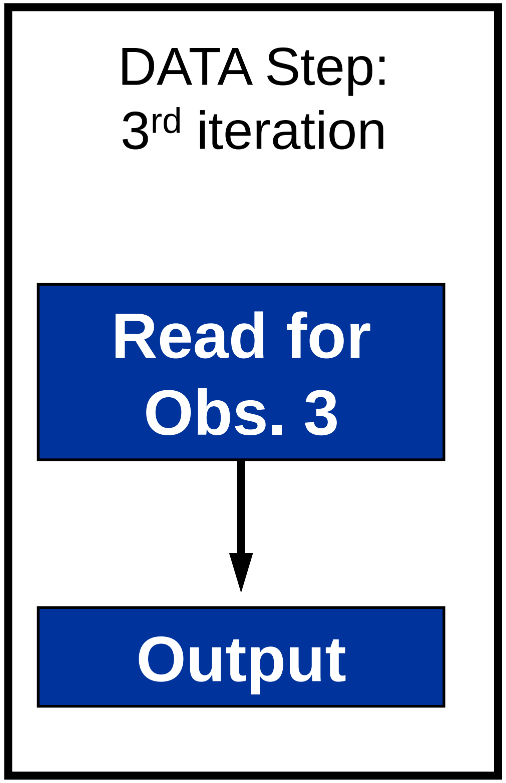


...

Processing: What Is Required?

charity.dat

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



DATA Step:
3rd iteration

Read for Obs. 3

Output

...

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.

p204d10

Execution: The Double Trailing @

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

Initialize 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	7	8	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

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

PDV

ID $ 4	Amount 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 $ 4	Amount N 8

...

Execution: The Double Trailing @

```
data donate07;
   length ID $ 4;
   infile 'charity.dat';
   input ID $ Amount @@;
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	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
AQI	495

...

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

																			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

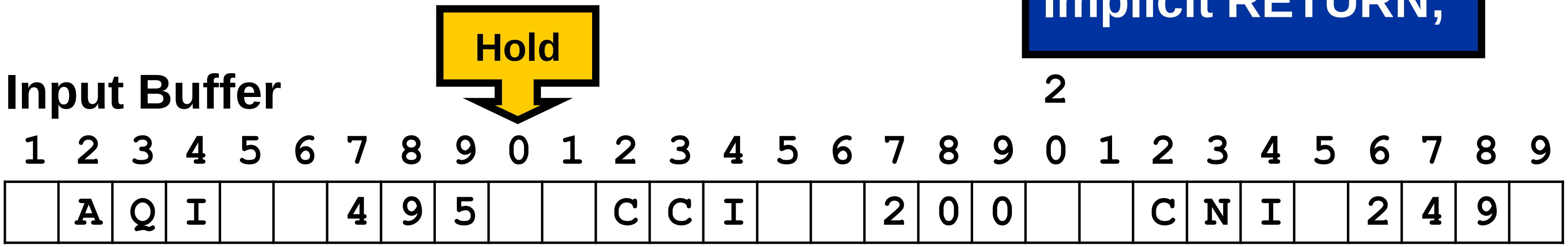
ID $ 4	Amount N 8
AQI	495

...

Execution: The Double Trailing @

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

Implicit OUTPUT;
Implicit RETURN;



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
AQI	495

...

Execution: The Double Trailing @

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

Reinitialize PDV

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

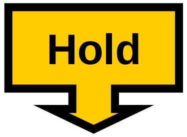
ID $ 4	Amount N 8
	.

...

Execution: The Double Trailing @

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

Specify input data file



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	

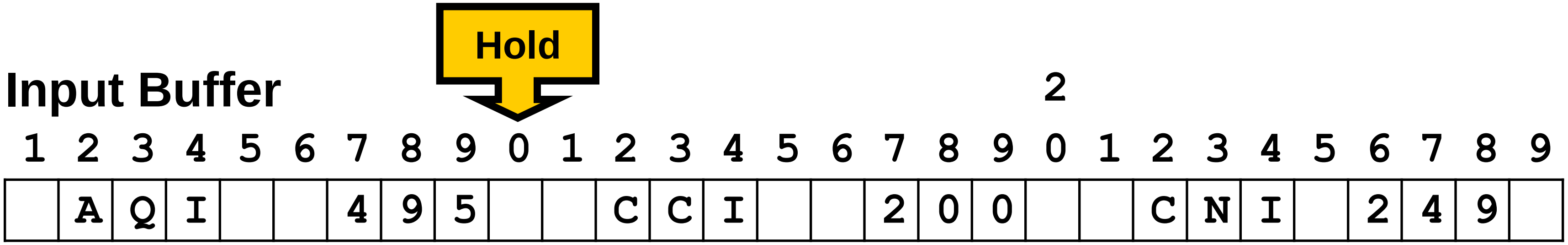
PDV

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

																			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;
```

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

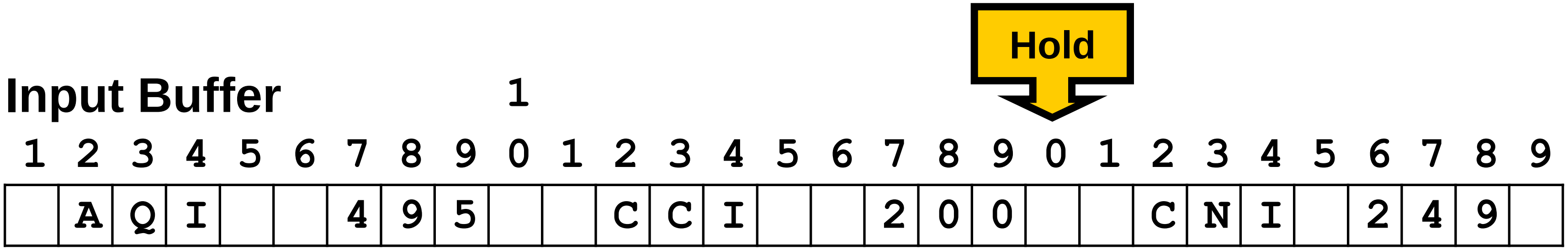
ID $ 4	Amount N 8
CCI	200

Execution: The Double Trailing @

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

Hold record in the input buffer

Input Buffer



									1																			
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	

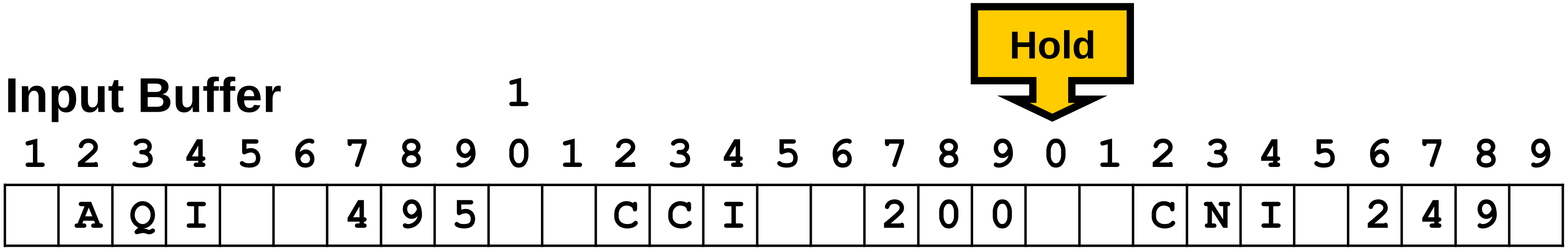
PDV

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



Input Buffer

Hold

									1																			
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

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.

Business Scenario – Read Contact Data

The raw data file **phone2.dat** contains contact names, phone numbers, and ratings for Orion customers.

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

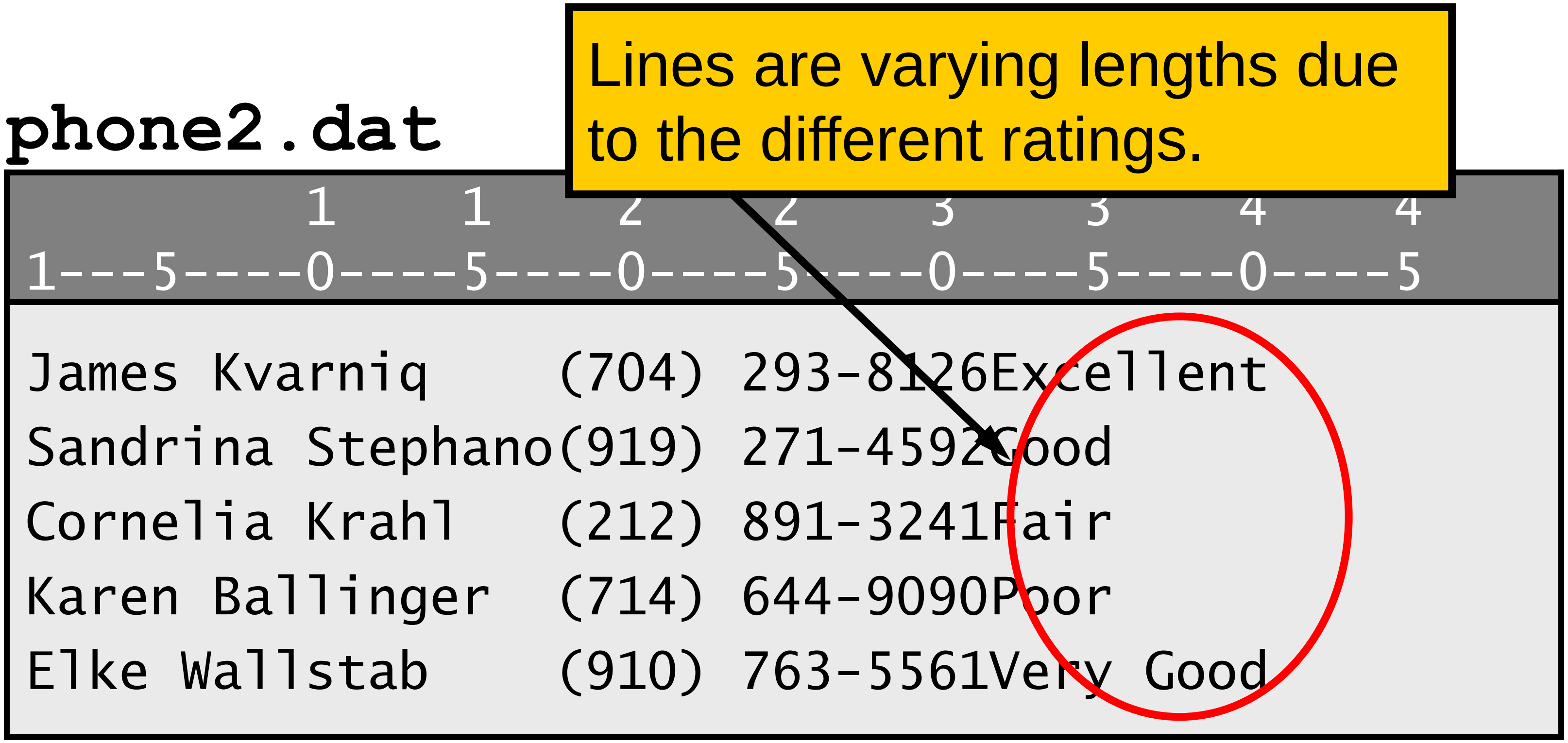
phone2.dat

```
         1    1    2    2    3    3    4    4
1---5----0----5----0----5----0----5----0----5

James Kvarniq    (704) 293-8126Excellent
Sandrina Stephano(919) 271-4592Good
Cornelia Krahl   (212) 891-3241Fair
Karen Ballinger  (714) 644-9090Poor
Elke Wallstab    (910) 763-5561Very Good
```

Short Values at the End of a Record

The data values in **`phone2.dat`** are fixed width. Each record has a contact name in columns 1-17, then a phone number in 18-31, and finally a rating in 32-40.

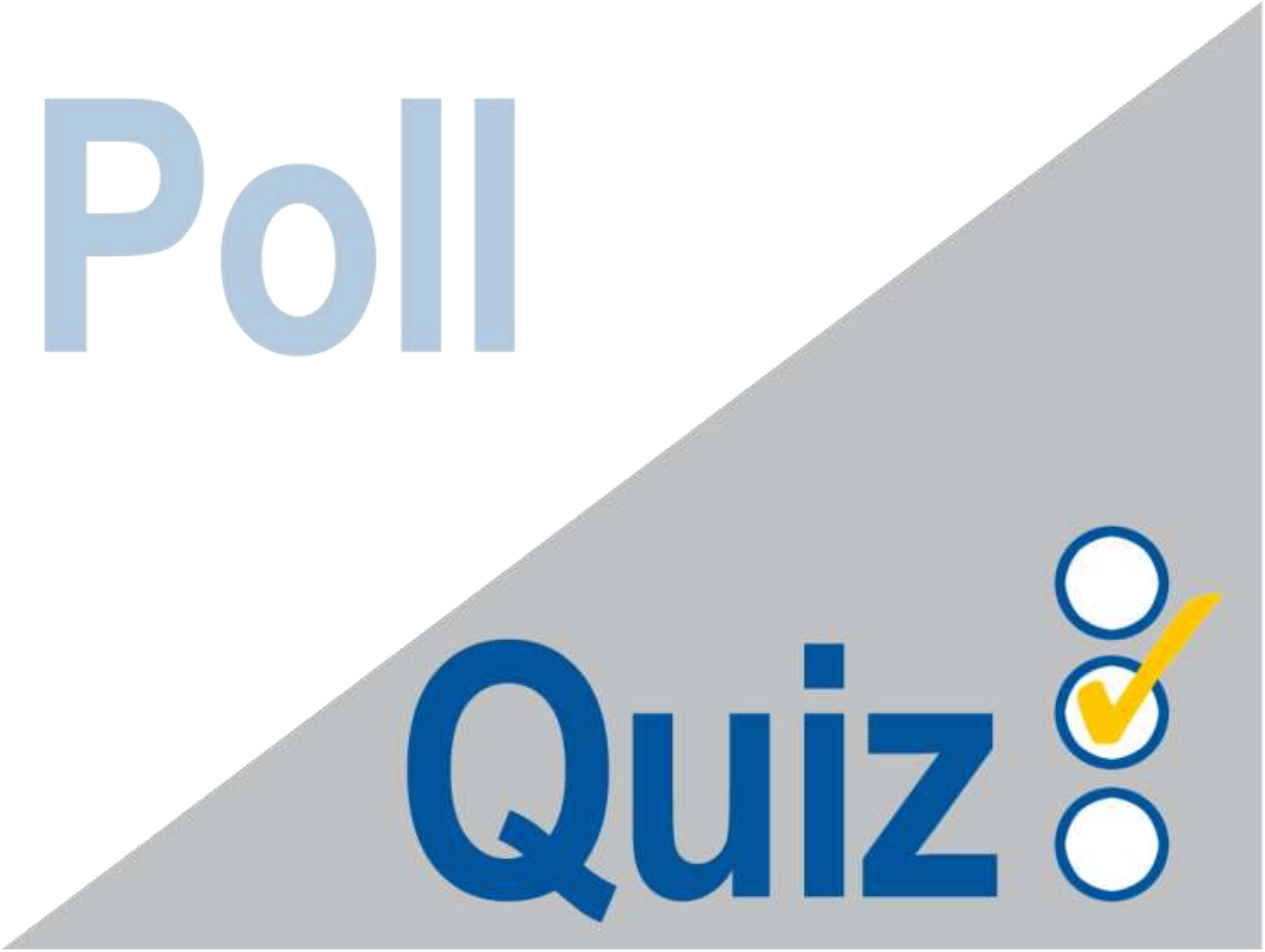


`phone2.dat`

Lines are varying lengths due to the different ratings.

```
         1    1    2    2    3    3    4    4
1---5----0----5----0----5----0----5----0----5

James Kvarniq    (704) 293-8126Excellent
Sandrina Stephano(919) 271-4592Good
Cornelia Krahl   (212) 891-3241Fair
Karen Ballinger  (714) 644-9090Poor
Elke Wallstab    (910) 763-5561Very Good
```


Poll
Quiz

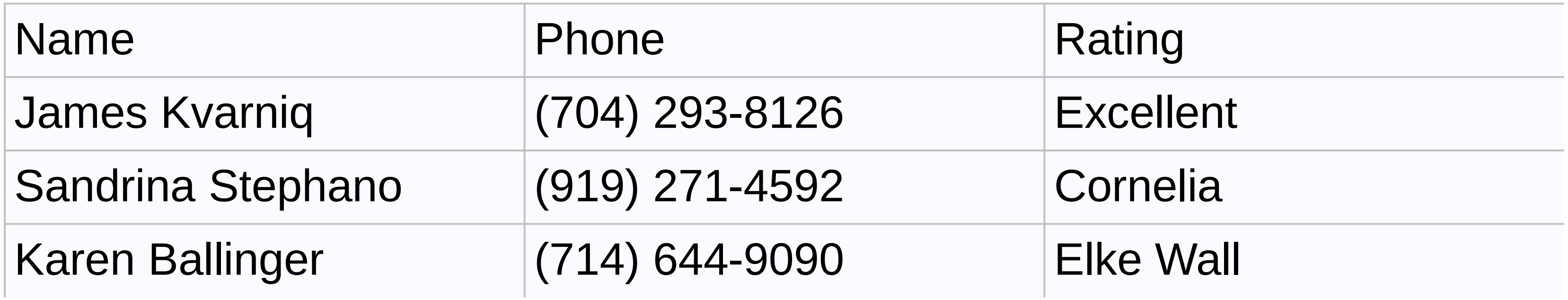
Quiz

How many observations will be created by the code shown below?

```
data contacts;
   infile 'phone2.dat';
   input Name $17.
         @18 Phone $14.
         @32 Rating $9.;
run;
```

Quiz – Correct Answer

Five records were read from the input file, and three observations were created.



Name	Phone	Rating
James Kvarniq	(704) 293-8126	Excellent
Sandrina Stephano	(919) 271-4592	Cornelia
Karen Ballinger	(714) 644-9090	Elke Wall

Unexpected Results

The short rating values have caused unexpected results in the output.

PROC PRINT output

Name	Phone	Rating
James Kvarniq	(704) 293-8126	Excellent
Sandrina Stephano	(919) 271-4592	Cornelia
Karen Ballinger	(714) 644-9090	Elke Wall

Partial SAS Log

```
NOTE: 5 records were read from the infile PHONE2.
      The minimum record length was 35.
      The maximum record length was 40.
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/Short Values at the End of a Record

By default, if the INPUT statement tries to read past the end of the current input data record, then it:

- moves the input pointer to column 1 of the next record to read the remaining values
- writes a note to the log

MISSOVER vs. TRUNCOVER Option

If the statement is unable to read an entire field because the value is shorter than the field length that is specified in the INPUT statement, the MISSOVER and TRUNCOVER options:

- do not allow the input pointer to go to the next record when the current INPUT statement is not satisfied.
- MISSOVER causes the INPUT statement to set a value to missing
- TRUNCOVER writes whatever characters are read to the appropriate variable

MISSOVER vs. TRUNCOVER Option

General form of an INFILE statement with a MISSOVER or TRUNCOVER option:

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

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

Poll
Quiz

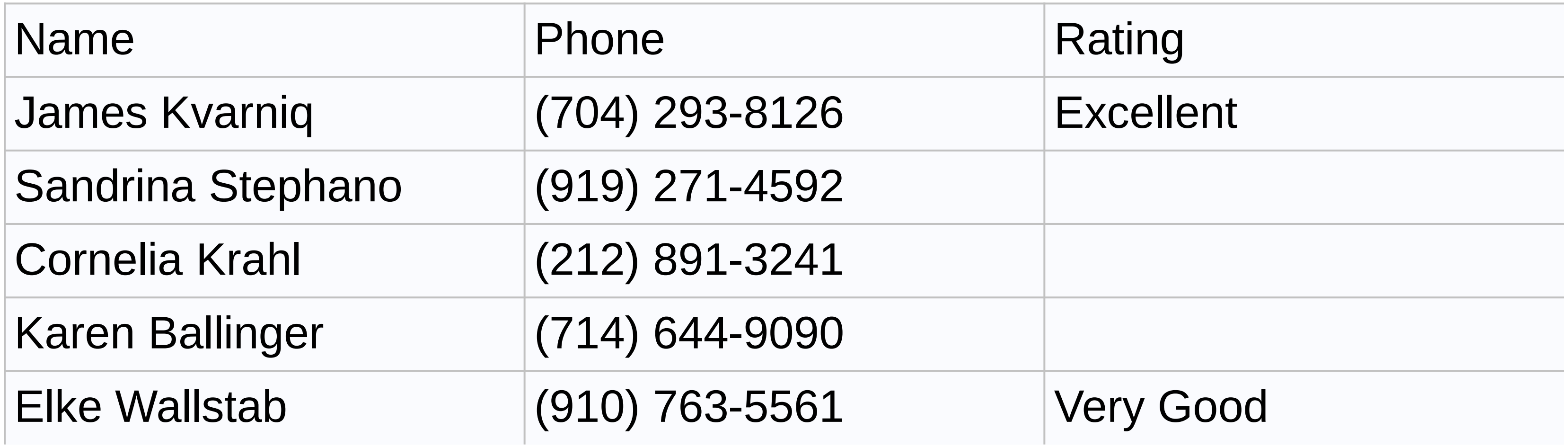
Quiz

How many observations will be created by the code shown below?

```
data contacts;
   infile 'phone2.dat' MISSOVER;
   input Name $17.
         @18 Phone $14.
         @32 Rating $9.;
run;
```

Quiz – Correct Answer

Five records were read from the input file, and five observations were created.



Name	Phone	Rating
James Kvarniq	(704) 293-8126	Excellent
Sandrina Stephano	(919) 271-4592	
Cornelia Krahl	(212) 891-3241	
Karen Ballinger	(714) 644-9090	
Elke Wallstab	(910) 763-5561	Very Good

Undesirable Results

Valuable information is still being lost.

PROC PRINT output

Name	Phone	Rating
James Kvarniq	(704) 293-8126	Excellent
Sandrina Stephano	(919) 271-4592	
Cornelia Krahl	(212) 891-3241	
Karen Ballinger	(714) 644-9090	
Elke Wallstab	(910) 763-5561	Very Good

Partial SAS Log

```
NOTE: 5 records were read from the infile PHONE2.
      The minimum record length was 35.
      The maximum record length was 40.
NOTE: The data set WORK.CONTACTS has 5 observations and
3 variables.
```

TRUNCOVER Option

The TRUNCOVER option will produce the desired results:

```
data contacts;
   infile 'phone2.dat' TRUNCOVER;
   input Name $17.
         @18 Phone $14.
         @32 Rating $9.;
run;
```

Results

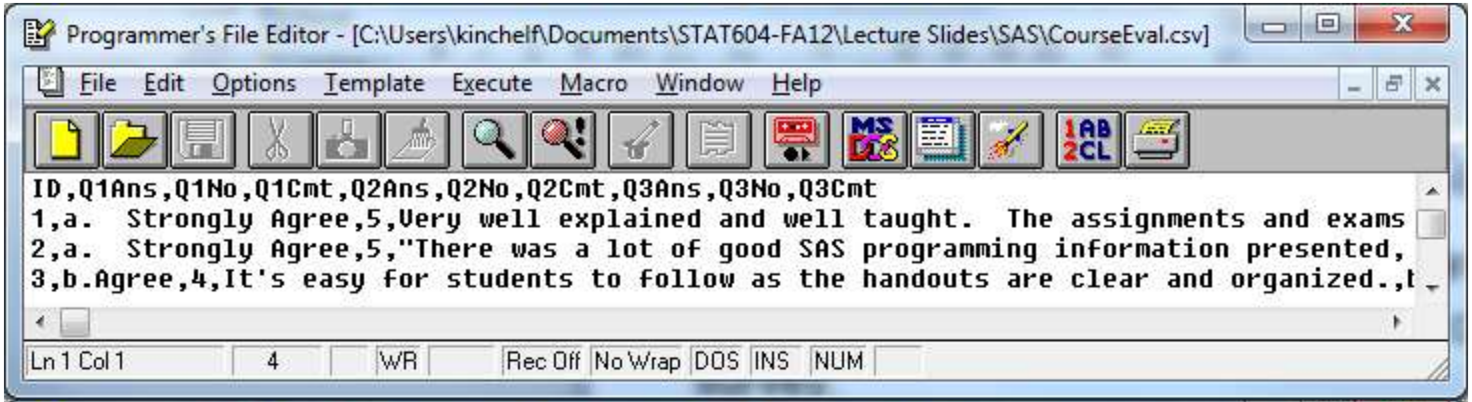
Adding the TRUNCOVER option gives the desired results.

PROC PRINT Output

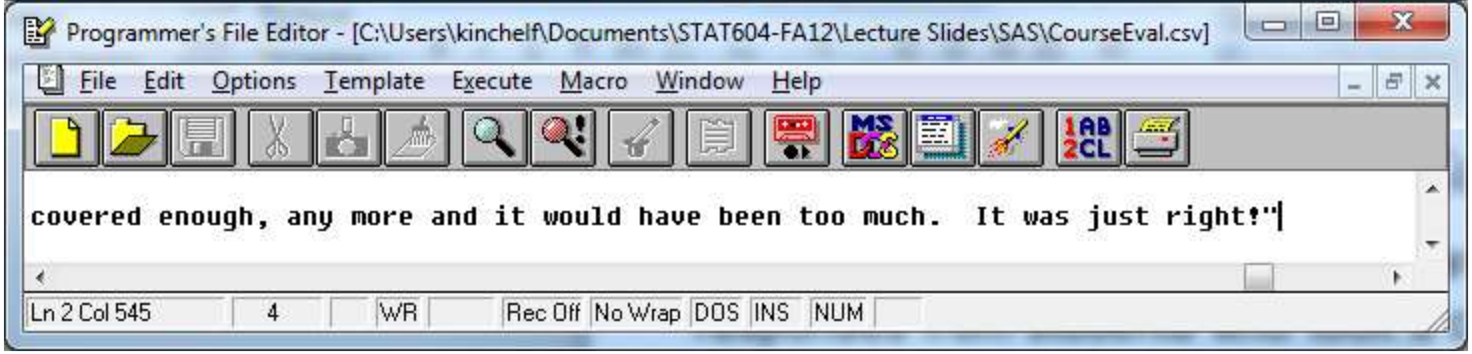
Name	Phone	Rating
James Kvarniq	(704) 293-8126	Excellent
Sandrina Stephano	(919) 271-4592	Good
Cornelia Krahl	(212) 891-3241	Fair
Karen Ballinger	(714) 644-9090	Poor
Elke Wallstab	(910) 763-5561	Very Good

Business Scenario – Read Survey Data

The raw data file **`CourseEval.csv`** contains responses from students who took a course evaluation survey.


Programmer's File Editor - [C:\Users\kinchelf\Documents\STAT604-FA12\Lecture Slides\SAS\CourseEval.csv]
File Edit Options Template Execute Macro Window Help
Ln 1 Col 1 | 4 | WR | Rec Off | No Wrap | DOS | INS | NUM

```
ID,Q1Ans,Q1No,Q1Cmt,Q2Ans,Q2No,Q2Cmt,Q3Ans,Q3No,Q3Cmt
1,a.  Strongly Agree,5,Very well explained and well taught.  The assignments and exams
2,a.  Strongly Agree,5,"There was a lot of good SAS programming information presented,
3,b.Agree,4,It's easy for students to follow as the handouts are clear and organized.,t
```


Programmer's File Editor - [C:\Users\kinchelf\Documents\STAT604-FA12\Lecture Slides\SAS\CourseEval.csv]
File Edit Options Template Execute Macro Window Help
Ln 2 Col 545 | 4 | WR | Rec Off | No Wrap | DOS | INS | NUM

```
covered enough, any more and it would have been too much.  It was just right!"
```

Poll
Quiz

Quiz

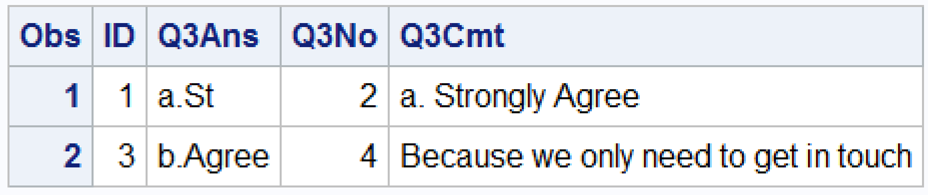
How many observations will be created by the code shown below?

```
data evals;
   infile 'CourseEval.csv' dlm=',' firstobs=2;
   length Q1Ans Q2Ans Q3Ans $ 20
          Q1Cmt Q2Cmt Q3Cmt $ 200;
   input ID Q1Ans $ Q1No Q1Cmt $ Q2Ans $
     Q2No Q2Cmt $ Q3Ans $ Q3No Q3Cmt $;
run;

proc print data=evals;
     var ID Q3Ans $ Q3No Q3Cmt;
run;
```

Quiz – Correct Answer

Three records were read from the input file, and two observations were created.



Obs	ID	Q3Ans	Q3No	Q3Cmt
1	1	a.St	2	a. Strongly Agree
2	3	b.Agree	4	Because we only need to get in touch

Unexpected Results

The long records have caused unexpected results in the output.

Partial SAS Log

```
NOTE: The infile EVAL is:
      Filename=C:\courseeval.csv,
      RECFM=V,LRECL=256,File Size (bytes)=1237,
      Last Modified=26Nov2012:20:35:14,
      Create Time=26Nov2012:20:35:14

NOTE: 3 records were read from the infile EVAL.
      The minimum record length was 256.
      The maximum record length was 256.
      One or more lines were truncated.
NOTE: SAS went to a new line when INPUT statement reached past the
end of a line.
NOTE: The data set WORK.EVALS has 2 observations and 10 variables.
```

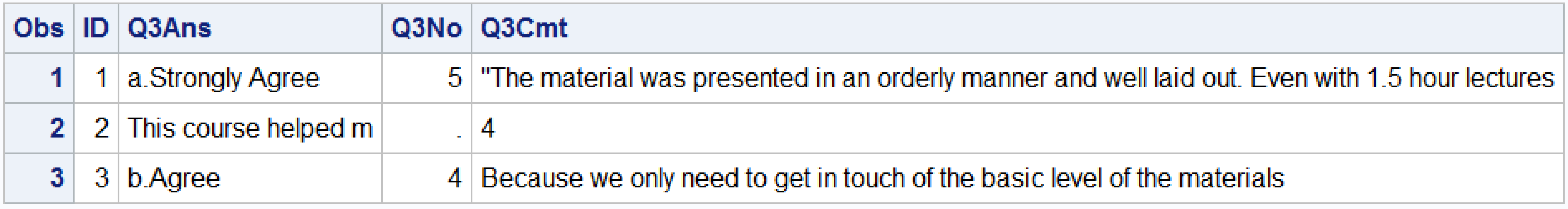
LRECL Infile Option

LRECL=value

specifies the logical record length of the file. If you do not specify an option, SAS chooses a value based on the system options for your installation. (256 pre-9.4, now 32767)

```
data evals;
   infile 'CourseEval.csv' dlm=',' firstobs=2
          LRECL=545 truncover;
   length Q1Ans Q2Ans Q3Ans $ 20
          Q1Cmt Q2Cmt Q3Cmt $ 200;
   input ID Q1Ans $ Q1No Q1Cmt $ Q2Ans $
     Q2No Q2Cmt $ Q3Ans $ Q3No Q3Cmt $;
run;
```

Almost there?



Obs	ID	Q3Ans	Q3No	Q3Cmt
1	1	a.Strongly Agree	5	"The material was presented in an orderly manner and well laid out. Even with 1.5 hour lectures
2	2	This course helped m	.	4
3	3	b.Agree	4	Because we only need to get in touch of the basic level of the materials

Partial SAS Log

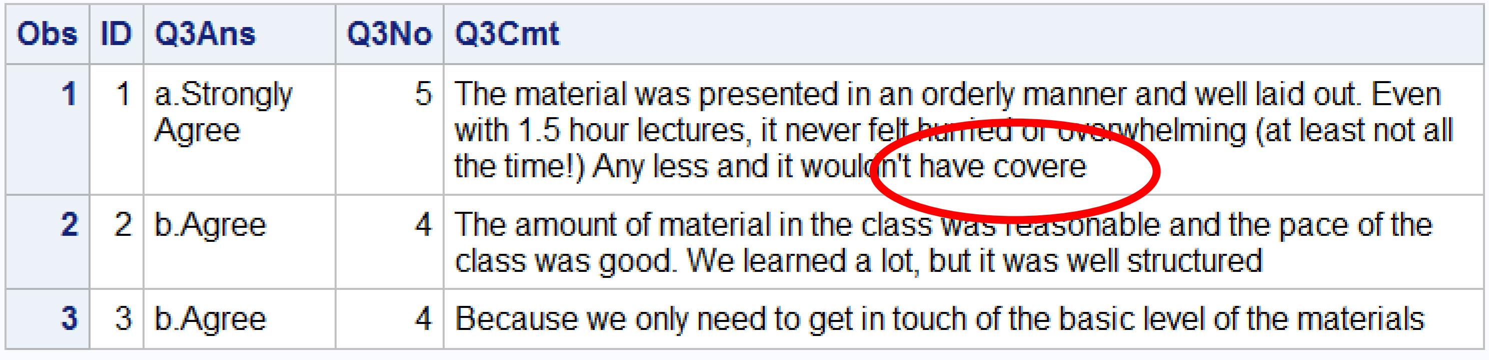
```
NOTE: Invalid data for Q2No in line 3 114-120.
NOTE: Invalid data for Q3No in line 3 198-204.
RULE:     ----+----1----+----2----+----3----+----4----+----5-

301  earned a lot, but it was well structured" 341
Q1Ans=a.  Strongly Agree Q2Ans=and is useful in my Q3Ans=This
course helped m
Q1Cmt="There was a lot of good SAS programming information
presented Q2Cmt=4 Q3Cmt=4 ID=2 Q1No=5 Q2No=. Q3No=. _ERROR_=1
_N_=2
NOTE: 3 records were read from the infile EVAL.
      The minimum record length was 291.
      The maximum record length was 544.
NOTE: The data set WORK.EVALS has 3 observations and 10 variables.
```

Deal with Imbedded Delimiters

```
data evals;
   infile 'CourseEval.csv' dsd firstobs=2
          LRECL=545 truncover;
   length Q1Ans Q2Ans Q3Ans $ 20
          Q1Cmt Q2Cmt Q3Cmt $ 200;
   input ID Q1Ans $ Q1No Q1Cmt $ Q2Ans $
     Q2No Q2Cmt $ Q3Ans $ Q3No Q3Cmt $;
run;
```

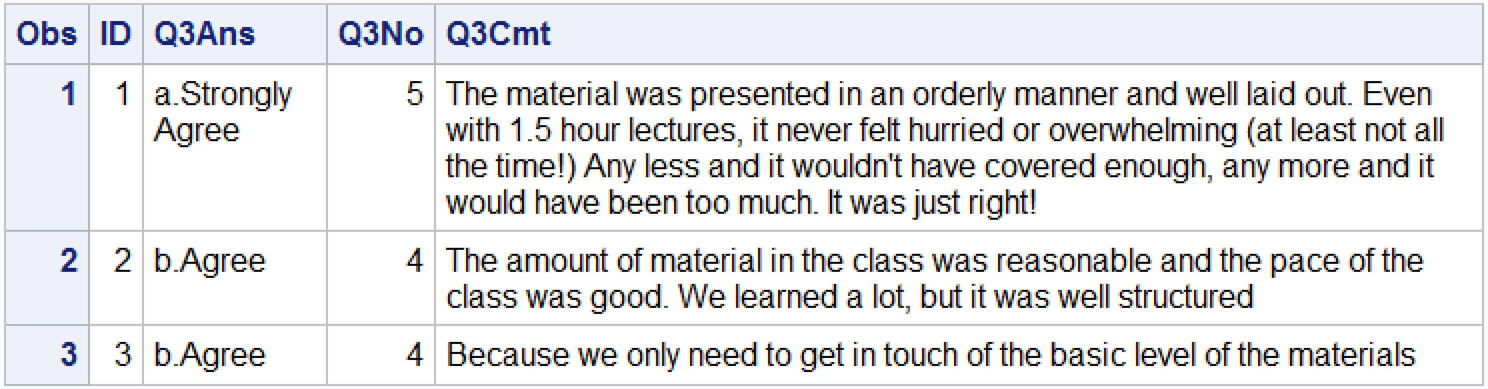
Close but not Quite!



Obs	ID	Q3Ans	Q3No	Q3Cmt
1	1	a.Strongly Agree	5	The material was presented in an orderly manner and well laid out. Even with 1.5 hour lectures, it never felt hurried or overwhelming (at least not all the time!) Any less and it wouldn't have covere
2	2	b.Agree	4	The amount of material in the class was reasonable and the pace of the class was good. We learned a lot, but it was well structured
3	3	b.Agree	4	Because we only need to get in touch of the basic level of the materials

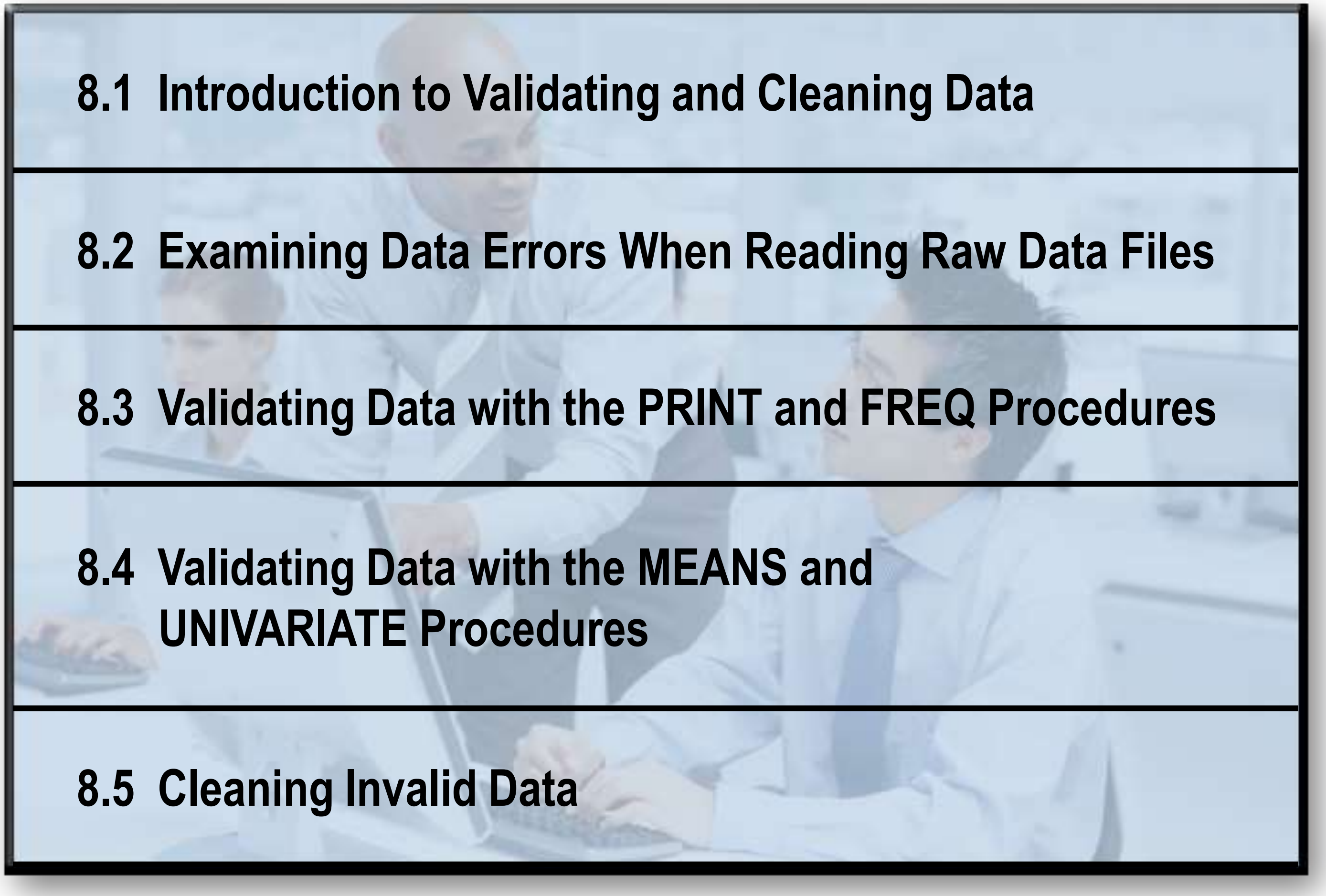
Adjust the Length for Desired Results

```
data evals;
   infile 'CourseEval.csv' dsd firstobs=2
          LRECL=545 truncover;
   length Q1Ans Q2Ans Q3Ans $ 20
          Q1Cmt Q2Cmt $ 200 Q3Cmt $ 275;
   input ID Q1Ans $ Q1No Q1Cmt $ Q2Ans $
     Q2No Q2Cmt $ Q3Ans $ Q3No Q3Cmt $;
run;
```



Obs	ID	Q3Ans	Q3No	Q3Cmt
1	1	a.Strongly Agree	5	The material was presented in an orderly manner and well laid out. Even with 1.5 hour lectures, it never felt hurried or overwhelming (at least not all the time!) Any less and it wouldn't have covered enough, any more and it would have been too much. It was just right!
2	2	b.Agree	4	The amount of material in the class was reasonable and the pace of the class was good. We learned a lot, but it was well structured
3	3	b.Agree	4	Because we only need to get in touch of the basic level of the materials

Chapter 8: Validating and Cleaning Data



8.1 Introduction to Validating and Cleaning Data

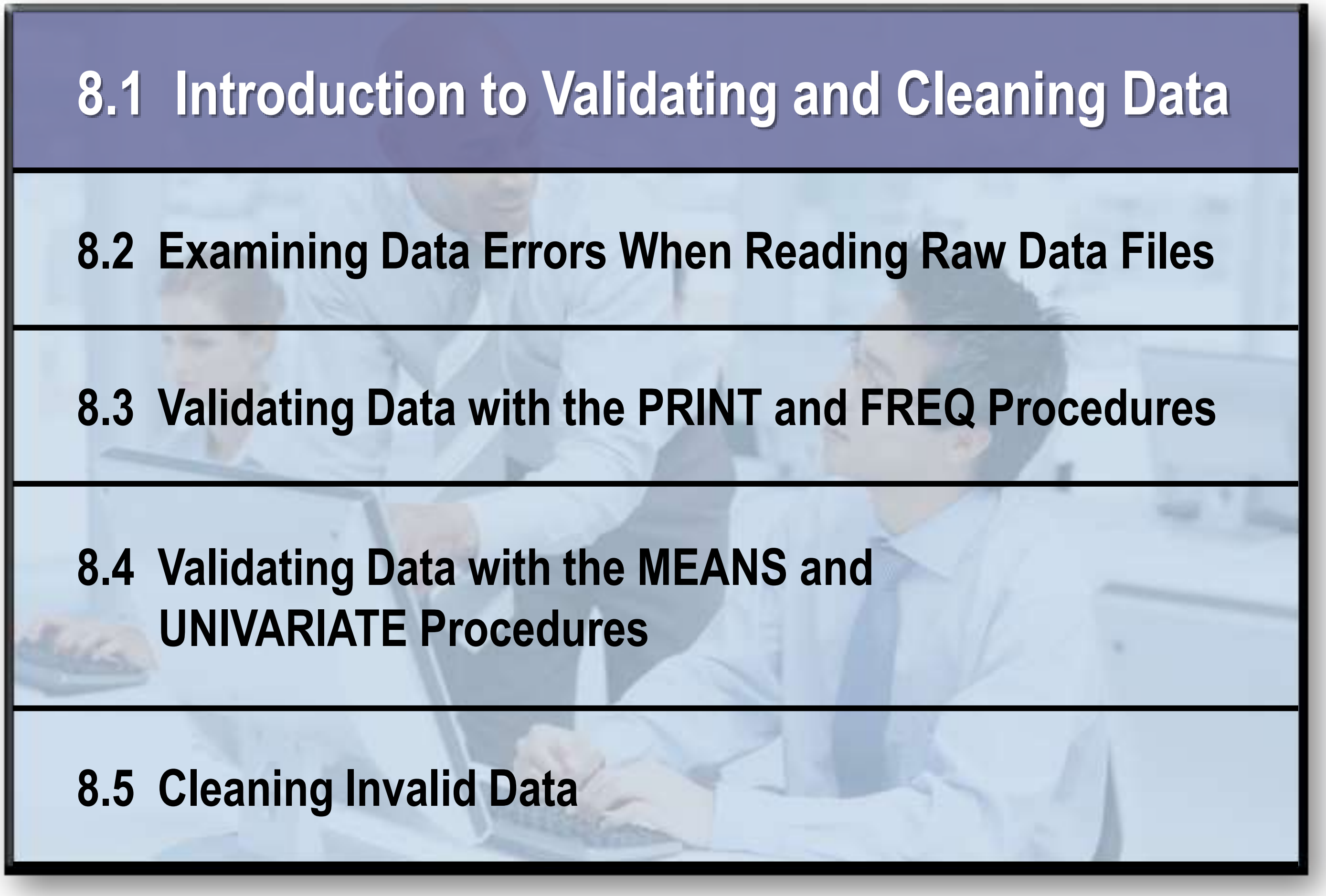
8.2 Examining Data Errors When Reading Raw Data Files

8.3 Validating Data with the PRINT and FREQ Procedures

8.4 Validating Data with the MEANS and UNIVARIATE Procedures

8.5 Cleaning Invalid Data

Chapter 8: Validating and Cleaning Data



8.1 Introduction to Validating and Cleaning Data

8.2 Examining Data Errors When Reading Raw Data Files

8.3 Validating Data with the PRINT and FREQ Procedures

8.4 Validating Data with the MEANS and UNIVARIATE Procedures

8.5 Cleaning Invalid Data

Objectives

- Define data errors in a raw data file.
- Identify procedures for validating data.
- Identify techniques for cleaning data.
- Define the business scenario that will be used with validating and cleaning data.

Business Scenario

A delimited raw data file containing information on Orion Star non-sales employees from Australia and the United States needs to be read to create a data set.

Requirements of non-sales employee data:

- **`Employee_ID`**, **`Salary`**, **`Birth_Date`**, and **`Hire_Date`** must be numeric variables.
- **`First`**, **`Last`**, **`Gender`**, **`Job_Title`**, and **`Country`** must be character variables.

Poll
Quiz

8.01 Quiz

What problems will SAS have reading the numeric data **Salary** and **Hire_Date**?

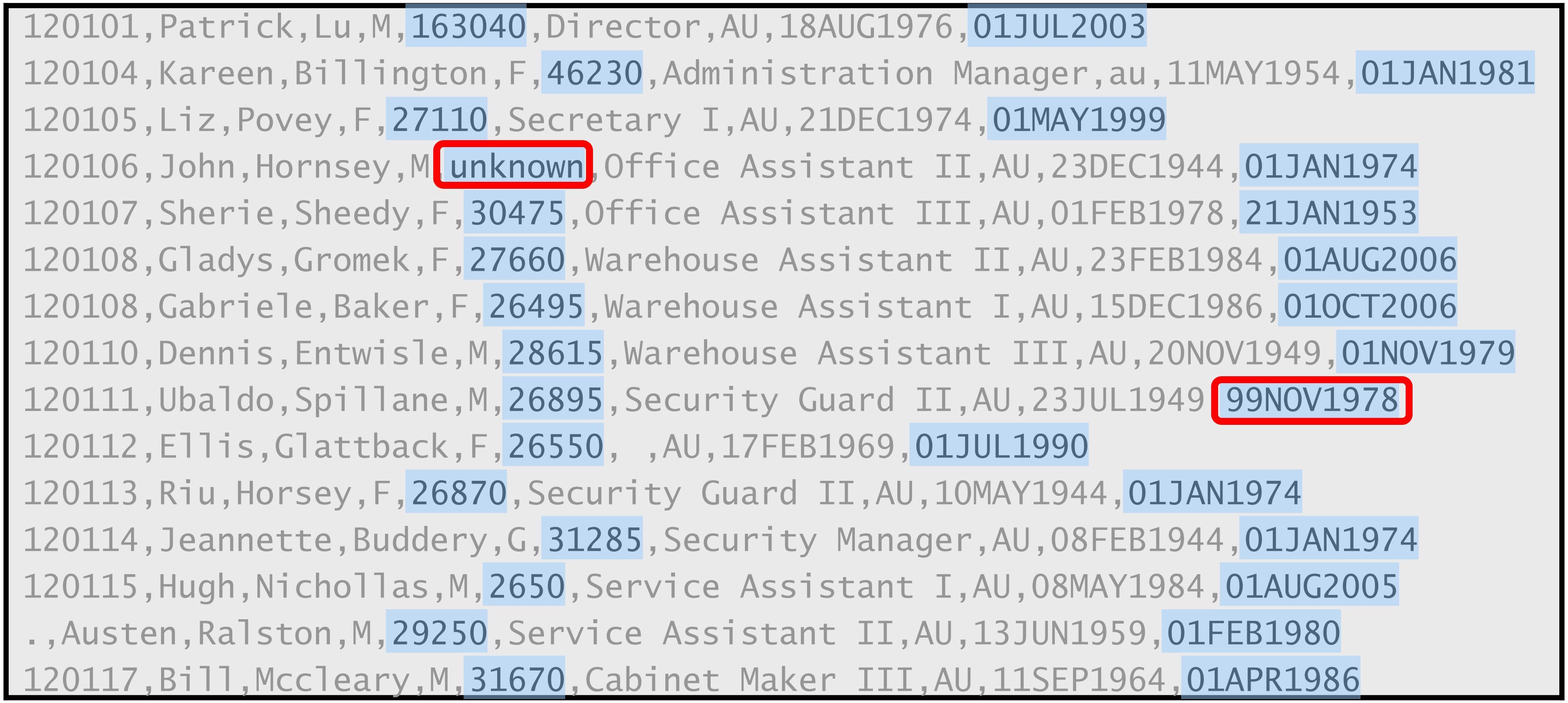
Partial **nonsales.csv**

```
120101,Patrick,Lu,M,163040,Director,AU,18AUG1976,01JUL2003
120104,Kareen,Billington,F,46230,Administration Manager,au,11MAY1954,01JAN1981
120105,Liz,Povey,F,27110,Secretary I,AU,21DEC1974,01MAY1999
120106,John,Hornsey,M,unknown,Office Assistant II,AU,23DEC1944,01JAN1974
120107,Sherie,Sheedy,F,30475,Office Assistant III,AU,01FEB1978,21JAN1953
120108,Gladys,Gromek,F,27660,Warehouse Assistant II,AU,23FEB1984,01AUG2006
120108,Gabriele,Baker,F,26495,Warehouse Assistant I,AU,15DEC1986,01OCT2006
120110,Dennis,Entwisle,M,28615,Warehouse Assistant III,AU,20NOV1949,01NOV1979
120111,Ubaldo,Spillane,M,26895,Security Guard II,AU,23JUL1949,99NOV1978
120112,Ellis,Glattback,F,26550, ,AU,17FEB1969,01JUL1990
120113,Riu,Horsey,F,26870,Security Guard II,AU,10MAY1944,01JAN1974
120114,Jeannette,Buddery,G,31285,Security Manager,AU,08FEB1944,01JAN1974
120115,Hugh,Nichollas,M,2650,Service Assistant I,AU,08MAY1984,01AUG2005
.,Austen,Ralston,M,29250,Service Assistant II,AU,13JUN1959,01FEB1980
120117,Bill,Mccleary,M,31670,Cabinet Maker III,AU,11SEP1964,01APR1986
```

8.01 Quiz – Correct Answer

What problems will SAS have reading the numeric data **`Salary`** and **`Hire_Date`**?

Partial **`nonsales.csv`**



```
120101,Patrick,Lu,M,163040,Director,AU,18AUG1976,01JUL2003
120104,Kareen,Billington,F,46230,Administration Manager,au,11MAY1954,01JAN1981
120105,Liz,Povey,F,27110,Secretary I,AU,21DEC1974,01MAY1999
120106,John,Hornsey,M,unknown,Office Assistant II,AU,23DEC1944,01JAN1974
120107,Sherie,Sheedy,F,30475,Office Assistant III,AU,01FEB1978,21JAN1953
120108,Gladys,Gromek,F,27660,Warehouse Assistant II,AU,23FEB1984,01AUG2006
120108,Gabriele,Baker,F,26495,Warehouse Assistant I,AU,15DEC1986,01OCT2006
120110,Dennis,Entwisle,M,28615,Warehouse Assistant III,AU,20NOV1949,01NOV1979
120111,Ubaldo,Spillane,M,26895,Security Guard II,AU,23JUL1949,99NOV1978
120112,Ellis,Glattback,F,26550, ,AU,17FEB1969,01JUL1990
120113,Riu,Horsey,F,26870,Security Guard II,AU,10MAY1944,01JAN1974
120114,Jeannette,Buddery,G,31285,Security Manager,AU,08FEB1944,01JAN1974
120115,Hugh,Nichollas,M,2650,Service Assistant I,AU,08MAY1984,01AUG2005
.,Austen,Ralston,M,29250,Service Assistant II,AU,13JUN1959,01FEB1980
120117,Bill,Mccleary,M,31670,Cabinet Maker III,AU,11SEP1964,01APR1986
```

Data Errors

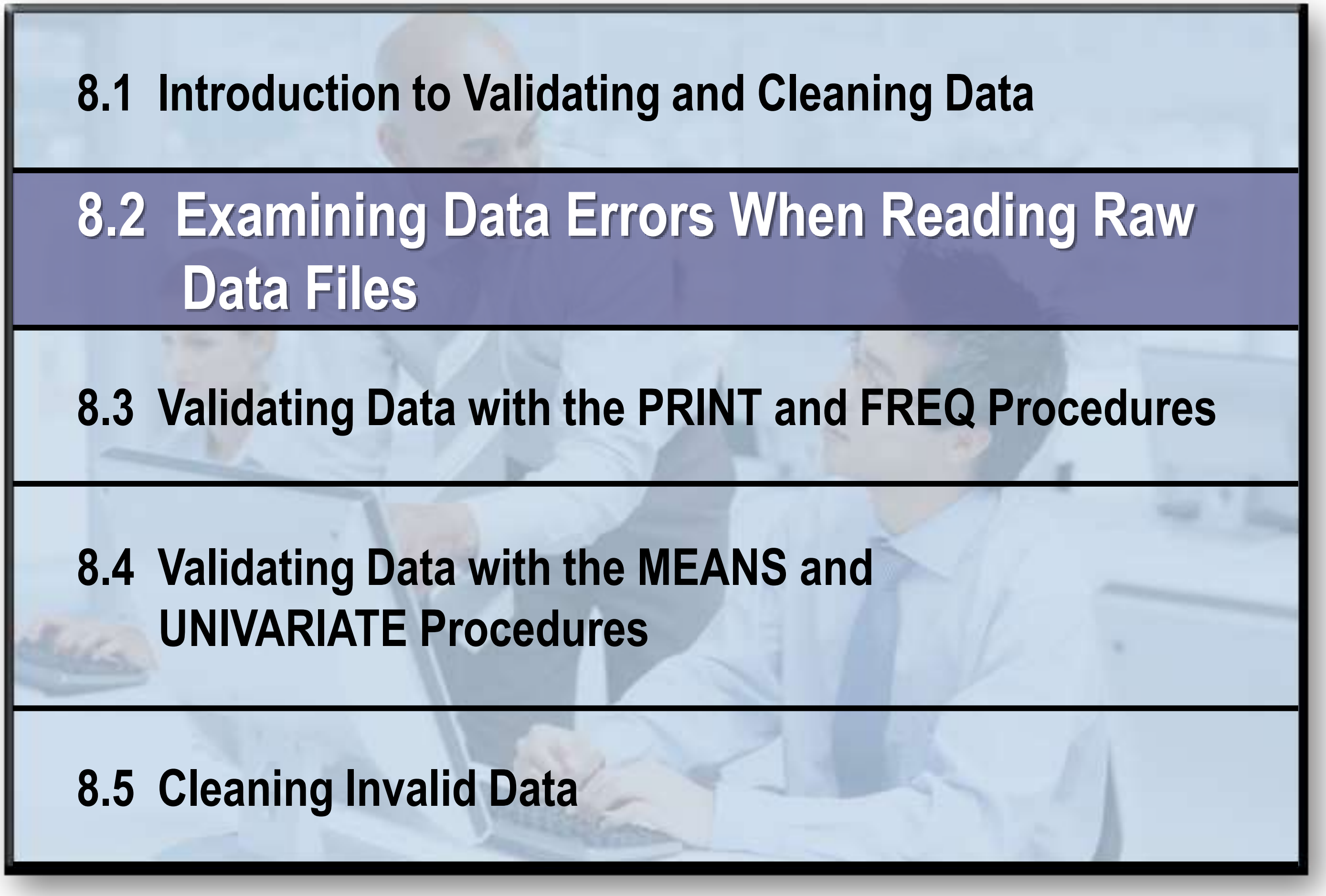
Data errors occur when data values are not appropriate for the SAS statements that are specified in a program.

- SAS detects data errors during program execution.
- When a data error is detected, SAS continues to execute the program.

```
NOTE: Invalid data for Salary in line 4 23-29.
RULE:     ----+----1----+----2----+----3----+----4----+----5----+----6
4         120106,John,Hornsey,M,unknown,Office Assistant II,AU,23DEC19
      61  44,01JAN1974 72
Employee_ID=120106 First=John Last=Hornsey Gender=M Salary=.
Job_Title=Office Assistant II Country=AU Birth_Date=23/12/1944
Hire_Date=01/01/1974 _ERROR_=1 _N_=4
NOTE: Invalid data for Hire_Date in li
9         120111,Ubaldo,Spillane,M,268                              94
      61  9,99NOV1978 71
Employee_ID=120111 First=Ubaldo Last=S
Job_Title=Security Guard II Country=AU
Hire_Date=. _ERROR_=1 _N_=9
```

A data error example is defining a variable as numeric, but the data value is actually character.

Chapter 8: Validating and Cleaning Data



8.1 Introduction to Validating and Cleaning Data

8.2 Examining Data Errors When Reading Raw Data Files

8.3 Validating Data with the PRINT and FREQ Procedures

8.4 Validating Data with the MEANS and UNIVARIATE Procedures

8.5 Cleaning Invalid Data

Objectives

- Identify data errors.
- Demonstrate what happens when a data error is encountered.
- Direct the observations with data errors to a different data set than the observations without data errors. (Self-Study)

Business Scenario

A delimited raw data file containing information on Orion Star non-sales employees from Australia and the United States needs to be read to create a data set.

Requirements of non-sales employee data:

- **`Employee_ID`**, **`Salary`**, **`Birth_Date`**, and **`Hire_Date`** must be numeric variables.
- **`First`**, **`Last`**, **`Gender`**, **`Job_Title`**, and **`Country`** must be character variables.

Data Errors

One type of data error is when the INPUT statement encounters invalid data in a field.

When SAS encounters a data error, these events occur:

- A note that describes the error is printed in the SAS log.
- The input record (contents of the input buffer) being read is displayed in the SAS log.
- The values in the SAS observation (contents of the PDV) being created are displayed in the SAS log.
- A missing value is assigned to the appropriate SAS variable.
- Execution continues.

Data Errors

A note that describes the error is printed in the SAS log.

Partial SAS Log

```
NOTE: Invalid data for Salary in line 4 23-29.
RULE:     ----+----1----+----2----+----3----+----4----+----5----+----6
4         120106,John,Hornsey,M,unknown,Office Assistant II,AU,23DEC19
      61  44,01JAN1974 72
Employee_ID=120106 First=John Last=Hornsey Gender=M Salary=.
Job_Title=Office Assistant II Country=AU Birth_Date=23/12/1944
Hire_Date=01/01/1974 _ERROR_=1 _N_=4
```

This note indicates that invalid data was found for variable **`Salary`** in line 4 of the raw data file in columns 23-29.

Data Errors

The input record (contents of the input buffer) being read is displayed in the SAS log.

Partial SAS Log

```
NOTE: Invalid data for Salary in line 4 23-29.
RULE:     ----+----1----+----2----+----3----+----4----+----5----+----6
4         120106,John,Hornsey,M,unknown,Office Assistant II,AU,23DEC19
      61  44,01JAN1974 72
Employee_ID=120106 First=John Last=Hornsey Gender=M Salary=.
Job_Title=Office Assistant II Country=AU Birth_Date=23/12/1944
Hire_Date=01/01/1974 _ERROR_=1 _N_=4
```

A ruler is drawn above the raw data record that contains the invalid data.

Data Errors

The values in the SAS observation (contents of the PDV) being created are displayed in the SAS log.

Partial SAS Log

```
NOTE: Invalid data for Salary in line 4 23-29.
RULE:     ----+----1----+----2----+----3----+----4----+----5----+----6
4         120106,John,Hornsey,M,unknown,Office Assistant II,AU,23DEC19
      61  44,01JAN1974 72
Employee_ID=120106 First=John Last=Hornsey Gender=M Salary=.
Job_Title=Office Assistant II Country=AU Birth_Date=23/12/1944
Hire_Date=01/01/1974 _ERROR_=1 _N_=4
```

Data Errors

A missing value is assigned to the appropriate SAS variable.

Partial SAS Log

```
NOTE: Invalid data for Salary in line 4 23-29.
RULE:     ----+----1----+----2----+----3----+----4----+----5----+----6
4         120106,John,Hornsey,M,unknown,Office Assistant II,AU,23DEC19
      61  44,01JAN1974 72
Employee_ID=120106 First=John Last=Hornsey Gender=M Salary=.
Job_Title=Office Assistant II Country=AU Birth_Date=23/12/1944
Hire_Date=01/01/1974 _ERROR_=1 _N_=4
```

Data Errors

During the processing of every DATA step, SAS automatically creates the following temporary variables:

- the _N_ variable, which counts the number of times the DATA step begins to iterate
- the _ERROR_ variable, which signals the occurrence of an error caused by the data during execution

  0 indicates that no errors exist.

  1 indicates that one or more errors occurred.

```
NOTE: Invalid data for Salary in line 4 23-29.
RULE:     ----+----1----+----2----+----3----+----4----+----5----+----6
4         120106,John,Hornsey,M,unknown,Office Assistant II,AU,23DEC19
      61  44,01JAN1974 72
Employee_ID=120106 First=John Last=Hornsey Gender=M Salary=.
Job_Title=Office Assistant II Country=AU Birth_Date=23/12/1944
Hire_Date=01/01/1974 _ERROR_=1 _N_=4
```

Poll

Quiz

8.04 Multiple Choice Poll

Which statement best describes the invalid data?

a. The data in the raw data file is bad.

b. The programmer incorrectly read the data.

Partial SAS Log

```
404      input Employee_ID First $ Last;
405   run;

NOTE: Invalid data for Last in line 1 16-17.
RULE:      ----+----1----+----2----+----3----+----4----+----5----+----6
1          120101,Patrick,Lu,M,163040,Director,AU,18AUG1976,01JUL2003 58
Employee_ID=120101 First=Patrick Last=. _ERROR_=1 _N_=1
NOTE: Invalid data for Last in line 2 15-24.
2          120104,Kareen,Billington,F,46230,Administration Manager,au,1
      61   1MAY1954,01JAN1981 78
Employee_ID=120104 First=Kareen Last=. _ERROR_=1 _N_=2
```

8.04 Multiple Choice Poll – Correct Answer

Which statement best describes the invalid data?

a. The data in the raw data file is bad.

[✓ b.] The programmer incorrectly read the data.

Last was read as numeric but needs to be read as character.

Partial SAS Log

```
404      input Employee_ID First $ Last;
405   run;

NOTE: Invalid data for Last in line 1 16-17.
RULE:     ----+----1----+----2----+----3----+----4----+----5----+----6
1         120101,Patrick,Lu,M,163040,Director,AU,18AUG1976,01JUL2003 58
Employee_ID=120101 First=Patrick Last=. _ERROR_=1 _N_=1
NOTE: Invalid data for Last in line 2 15-24.
2         120104,Kareen,Billington,F,46230,Administration Manager,au,1
      61  1MAY1954,01JAN1981 78
Employee_ID=120104 First=Kareen Last=. _ERROR_=1 _N_=2
```

Outputting to Multiple Data Sets

The DATA statement can specify multiple output data sets.

```
data work.baddata work.gooddata;
   length Employee_ID 8 First $ 12 Last $ 18
          Gender $ 1 Salary 8 Job_Title $ 25
          Country $ 2 Birth_Date Hire_Date 8;
   infile 'nonsales.csv' dlm=',';
   input Employee_ID First $ Last $
         Gender $ Salary Job_Title $ Country $
         Birth_Date :date9.
         Hire_Date :date9.;
   format Birth_Date Hire_Date ddmmyy10.;
   if _error_=1 then output work.baddata;
   else output work.gooddata;
run;
```

p108d03

Outputting to Multiple Data Sets

An OUTPUT statement can be used in a conditional statement to write the current observation to a specific data set that is listed in the DATA statement.

```
data work.baddata work.gooddata;
   length Employee_ID 8 First $ 12 Last $ 18
          Gender $ 1 Salary 8 Job_Title $ 25
          Country $ 2 Birth_Date Hire_Date 8;
   infile 'nonsales.csv' dlm=',';
   input Employee_ID First $ Last $
         Gender $ Salary Job_Title $ Country $
         Birth_Date :date9.
         Hire_Date :date9.;
   format Birth_Date Hire_Date ddmmyy10.;
   if _error_=1 then output work.baddata;
   else output work.gooddata;
run;
```

p108d03

Outputting to Multiple Data Sets

Partial SAS Log

```
NOTE: Invalid data for Salary in line 4 23-29.
RULE:     ----+----1----+----2----+----3----+----4----+----5----+----6
4         120106,John,Hornsey,M,unknown,Office Assistant II,AU,23DEC19
      61  44,01JAN1974 72
Employee_ID=120106 First=John Last=Hornsey Gender=M Salary=.
Job_Title=Office Assistant II Country=AU Birth_Date=23/12/1944
Hire_Date=01/01/1974 _ERROR_=1 _N_=4
NOTE: Invalid data for Hire_Date in line 9 63-71.
9         120111,Ubaldo,Spillane,M,26895,Security Guard II,AU,23JUL194
      61  9,99NOV1978 71
Employee_ID=120111 First=Ubaldo Last=Spillane Gender=M Salary=26895
Job_Title=Security Guard II Country=AU Birth_Date=23/07/1949
Hire_Date=. _ERROR_=1 _N_=9
NOTE: 235 records were read from the infile
      's:\workshop\nonsales.csv'.
      The minimum record length was 55.
      The maximum record length was 82.
NOTE: The data set WORK.BADDATA has 2 observations and 9 variables.
NOTE: The data set WORK.GOODDATA has 233 observations and 9 variables.
```