UCD School of Mathematics and Statistics

STAT40840: Data programming with SAS
Laura Kirwan

Lecture 6

6.1 Introduction to Reading Raw Data Files

6.2 Reading Standard Delimited Data

6.3 Using Informats to Read Delimited Data

6.4 Handling Missing Data

Objectives – Part 1

- Identify the types of raw data files and input styles.
- Define the terms *standard* and *nonstandard* data.

Information about Orion Star sales employees from Australia and the United States is stored in a raw data file.

Raw data file

Programmers need to be able to identify the layout and type of information in the raw data file.

Raw Data Files

- A raw data file is also known as a *flat file*.
  - They are text files that contain one record per line.
  - A record typically contains multiple fields.
  - Flat files do not have internal metadata.
  - External documentation, known as a *record layout*, should exist.
  - A record layout describes the fields and locations within each record.

Raw Data Files

Fields in a raw data file can be delimited or arranged in fixed columns.

Delimited File

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
```

Fixed Column File

```
         1    1    2    2    3    3    4    4    5    5    6
1---5----0----5----0----5----0----5----0----5----0----5----0--
120102Tom        Zhou            Sales Manager       108255AU
120103Wilson     Dawes           Sales Manager        87975AU
120121Irenie     Elvish          Sales Rep. II        26600AU
120122Christina  Ngan            Sales Rep. II        27475AU
```

Fields in Raw Data Files

- In order for SAS to read a raw data file, you must specify the following information about each field:
  - the location of the data value in the record
  - the name of the SAS variable in which to store the data
  - the type of the SAS variable

Reading Raw Data Files

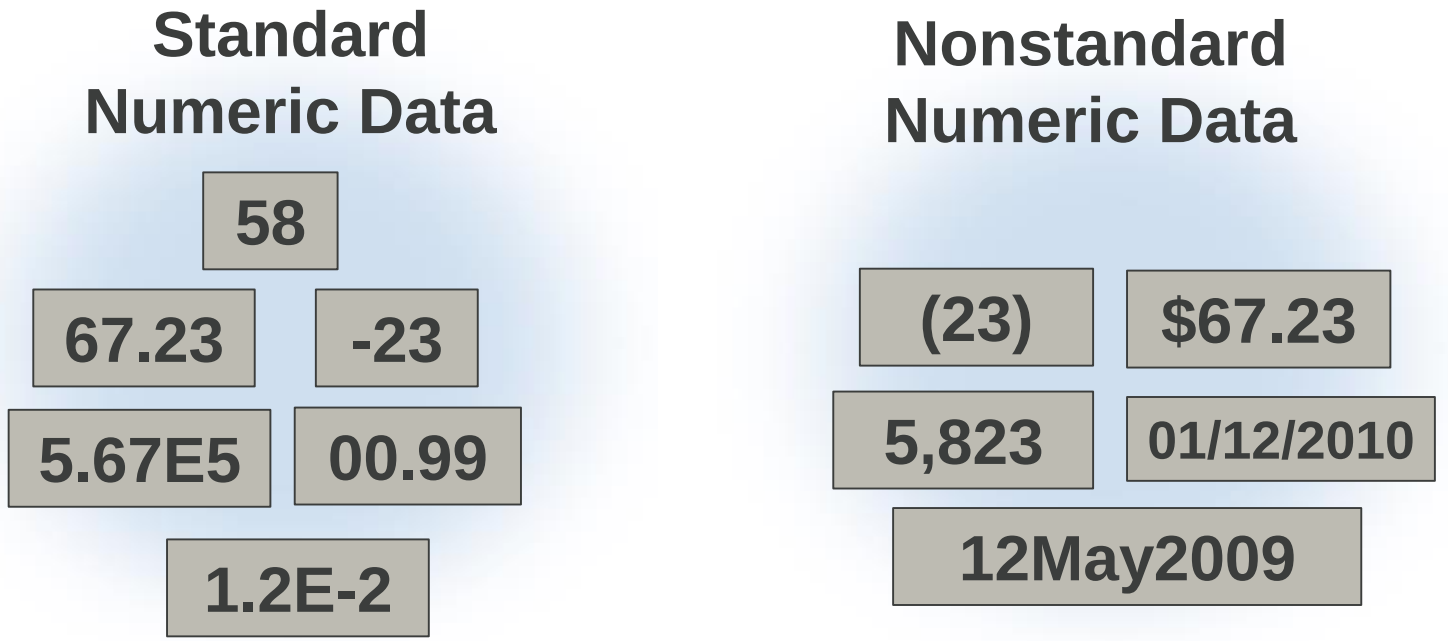
- There are different techniques, or *input styles*, for reading raw data files in SAS.

Input 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

Standard data is data that SAS can read without any additional instruction.

- Character data is always standard.
- Some numeric values are standard and some are not.



6.1 Introduction to Reading Raw Data Files

6.2 Reading Standard Delimited Data

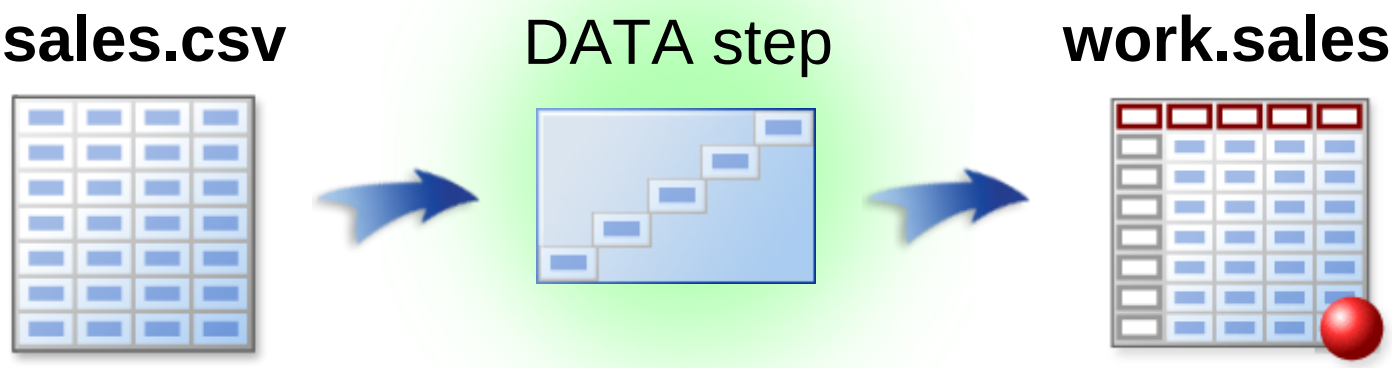
6.3 Using Informats to Read Delimited Data

6.4 Handling Missing Data

Objectives – Part 2

- Use list input to create a SAS data set from a delimited raw data file.
- Examine the compilation and execution phases of the DATA step when you read a raw data file.
- Explicitly define the length of a variable.
- Examine behavior when a data error is encountered.

Information about Orion Star sales employees is stored in a comma-delimited raw data file. The file contains both standard and nonstandard data fields.



List Input

Use list input to read delimited raw data files.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1968,10/01/1989
```

- SAS considers a space (blank) to be the default delimiter.
- Both standard and nonstandard data can be read.
- Fields must be read sequentially, left to right.

Exercise 1

Which fields in this file can be read as standard numeric values?

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1968,10/01/1989
```

Reading a Delimited Raw Data File

Use *INFILE* and *INPUT statements* in a DATA step to read a raw data file.

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $
         Last_Name $ Gender $ Salary
         Job_Title $ Country $;
run;
```

DATA *output-data-set*;
    INFILE "*raw-data-file*" <**DLM='***delimiter***'**>;
    INPUT *variable* <**$**> *variable* <**$**> … ;
RUN;

L6_D1.sas

INFILE Statement

The INFILE statement identifies the raw data file to be read.

```
infile "&path\sales.csv" dlm=',';
```

INFILE "*raw-data-file***" <DLM='***delimiter***'>;**

- A full path is recommended.
- Using the **&path** macro variable reference makes the program more flexible.
- The DLM= option specifies alternate delimiters.

Be sure to use double quotation marks when you reference a macro variable within a quoted string.

INPUT Statement

The INPUT statement reads the data fields sequentially, left to right. Standard data fields require only a variable name and type.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1969,06/01/1989
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1949,01/01/1974
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1944,01/01/1974
```

```
input Employee_ID First_Name $ Last_Name $
      Gender $ Salary Job_Title $ Country $;
```

INPUT *variable* <$> *variable* <$> …;

- The dollar sign indicates a character variable.
- Default length for ***all*** variables is eight bytes, regardless of type.

Viewing the Log

Partial SAS Log

```
249  data work.subset;
250     infile "&path\sales.csv" dlm=',';
251     input Employee_ID First_Name $ Last_Name $
252           Gender $ Salary Job_Title $ Country $;
253  run;

NOTE: The infile "s:\workshop\sales.csv" is:
      Filename=s:\workshop\sales.csv,
      RECFM=V,LRECL=256,File Size (bytes)=11340

NOTE: 165 records were read from the infile "s:\workshop\sales.csv".
      The minimum record length was 61.
      The maximum record length was 80.
NOTE: The data set WORK.SUBSET has 165 observations and 7 variables.
```

L6_D1.sas

Viewing the Output

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

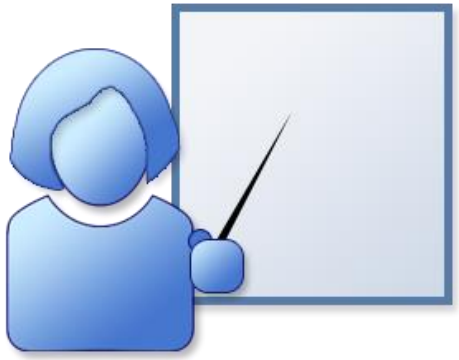
- Partial PROC PRINT Output

Employee_ ID	First_ Name	Last_ Name	Gender	Salary	Job_ Title	Country
120102	Tom	Zhou	M	108255	Sales Ma	AU
120103	Wilson	Dawes	M	87975	Sales Ma	AU
120121	Irenie	Elvish	F	26600	Sales Re	AU
120122	Christin	Ngan	F	27475	Sales Re	AU
120123	Kimiko	Hotstone	F	26190	Sales Re	AU

Some character values are truncated.

L6_D1.sas

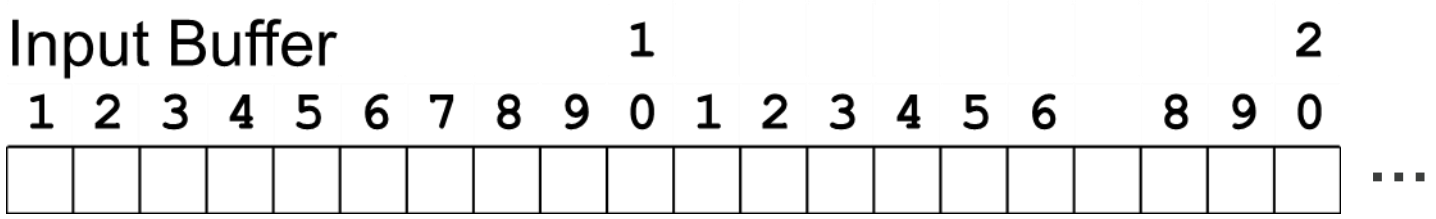
It is important to understand the processing that occurs when a DATA step reads a raw data file.



Compilation Phase

During compilation, SAS does the following:

- scans the step for syntax errors
- translates each statement into machine language
- creates an *input buffer* to hold one record at a time from the raw data file



- creates the program data vector (PDV) to hold one observation
- creates the descriptor portion of the output data set

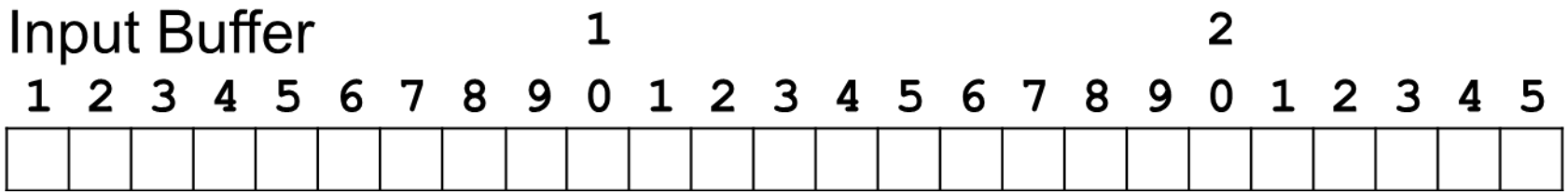
Compilation

```
data work.subset;
    infile "&path\sales.csv" dlm=',';
    input Employee_ID First_Name $ Last_Name $
          Gender $ Salary Job_Title $ Country $;
run;
```

L6_D1.sas

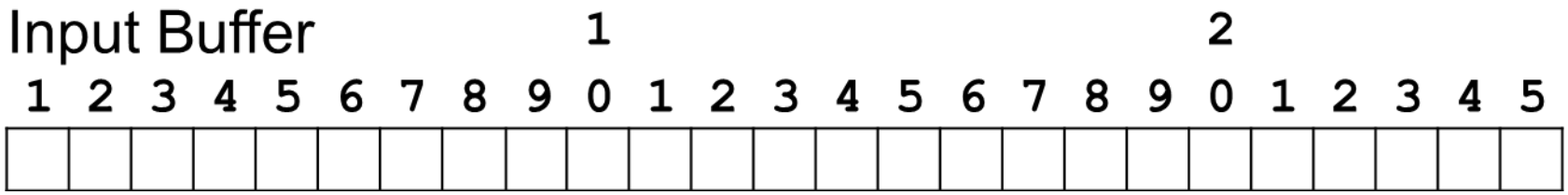
Compilation

```
data work.subset;
    infile "&path\sales.csv" dlm=',';
    input Employee_ID First_Name $ Last_Name $
          Gender $ Salary Job_Title $ Country $;
run;
```



Compilation

```
data work.subset;
    infile "&path\sales.csv" dlm=',';
    input Employee_ID First_Name $ Last_Name $
          Gender $ Salary Job_Title $ Country $;
run;
```



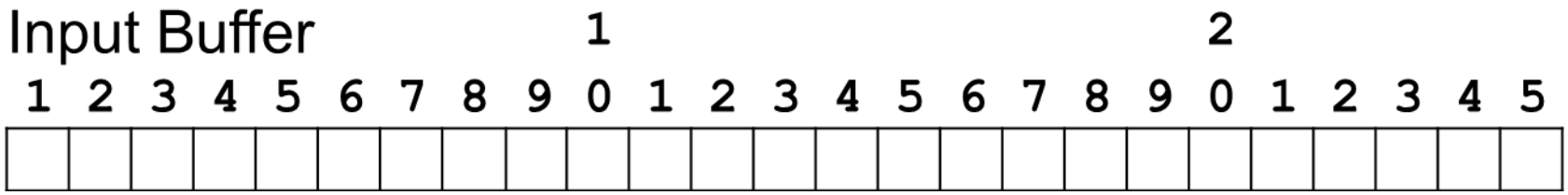
PDV

Employee_ID
N 8

Attributes are based on the INPUT statement.

Compilation

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```



PDV

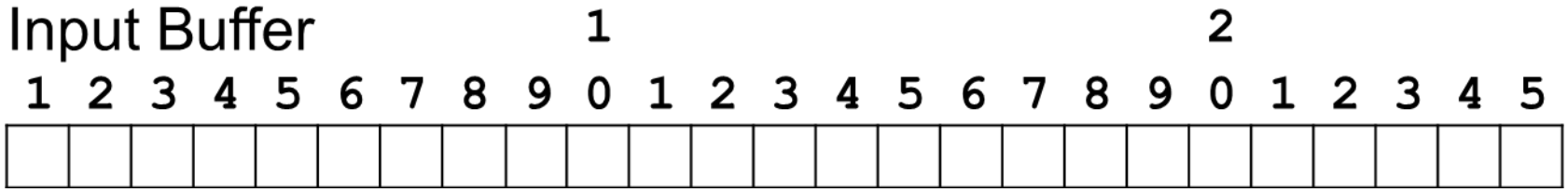
Employee_ID N 8	First_Name $ 8

With list input, the default length for character variables is eight bytes.

Compilation

```
data work.subset;
    infile "&path\sales.csv" dlm=',';
    input Employee_ID First_Name $ Last_Name $
          Gender $ Salary Job_Title $ Country $;
run;
```

Input Buffer



PDV

Employee_ID N 8	First_ Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8

Compilation

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8

Descriptor Portion of **work.subset**

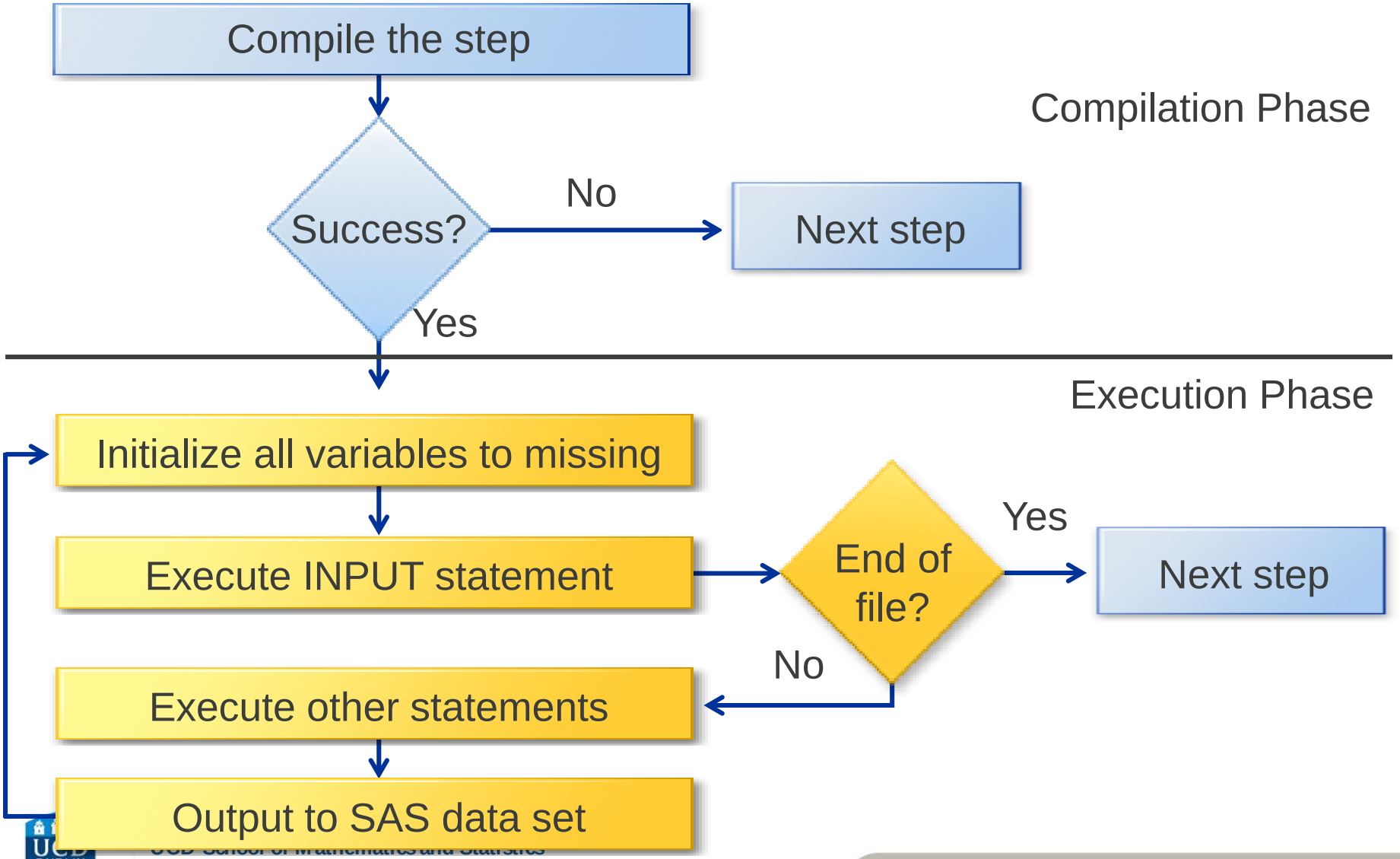
Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8

Exercise 2

Which statement is true?

a. An input buffer is created only if you are reading data from a raw data file.
b. The PDV at compile time holds the variable name, type, byte size, and initial value.
c. The descriptor portion is the first item that is created at compile time.

DATA Step Processing



Execution

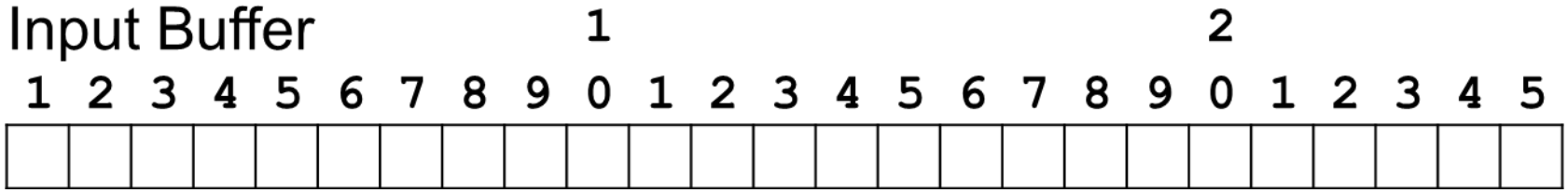
Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

Initialize PDV

```
data work.subset;
   infile "&path\sales.csv"
          dlm=',';
   input Employee_ID First_Name $
         Last_Name $ Gender $
         Salary Job_Title $
         Country $;
run;
```

Input Buffer



PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
.				.		

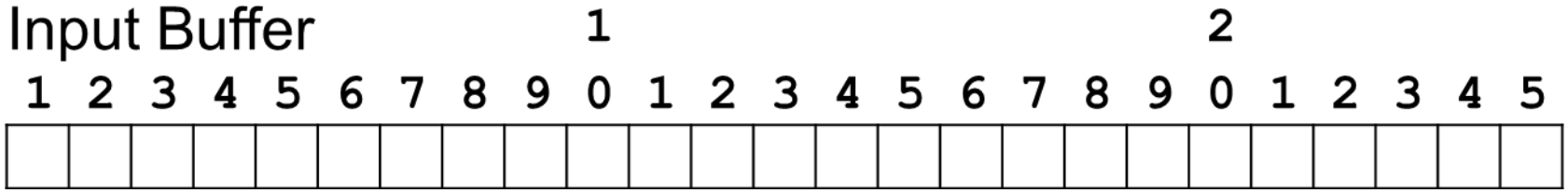
Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer



PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
.				.		

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

SAS reads a record into the input buffer.

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	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
.				.		

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

SAS scans until it reaches a delimiter.

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	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
.				.		

Execution

Partial **sales.csv**

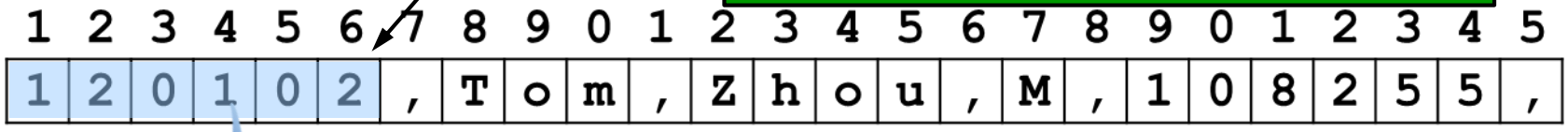
```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $

run
```

The value is converted from text to a floating-point numeric value and copied to the PDV.

Input Buffer



PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
120102				.		

Execution

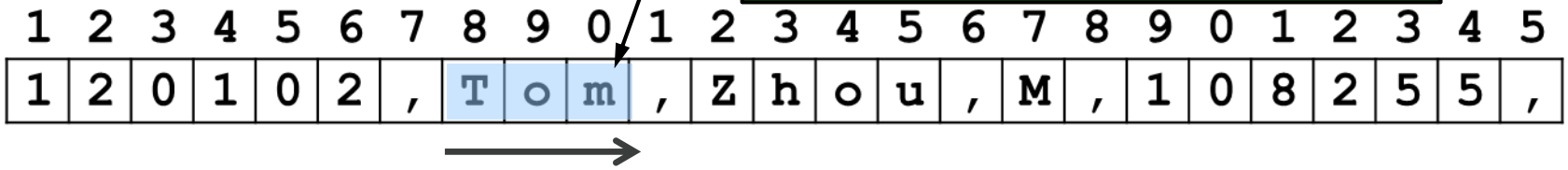
Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

SAS skips the delimiter and scans to the next delimiter.

Input Buffer



PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
120102				.		

Execution

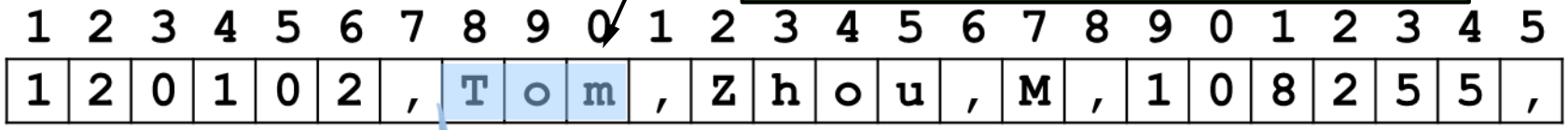
Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

The text value is copied to the PDV without conversion.

Input Buffer



PDV

Employee _ID N 8	First _Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _Title $ 8	Country $ 8
120102	Tom			.		

Execution

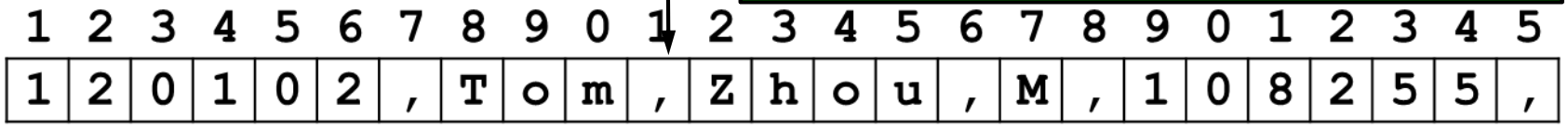
Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

These actions continue for all variables in the INPUT statement.

Input Buffer



PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
120102	Tom	Zhou	M	108255	Sales Ma	AU

Execution

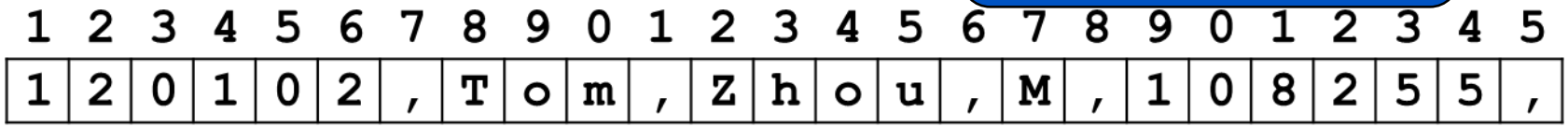
Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Implicit OUTPUT;
Implicit RETURN;

Input Buffer



PDV

Employee _ID N 8	First _Name $ 8	Last _Name $ 8	Gender $ 8	Salary N 8	Job _Title $ 8	Country $ 8
120102	Tom	Zhou	M	108255	Sales Ma	AU

Execution

- Here is the output data set after the first iteration of the DATA step.

work.subset

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	Country
120102	Tom	Zhou	M	108255	Sales Ma	AU

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Implicit OUTPUT;
Implicit RETURN;

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
1 2 0 1 0 2 , T o m , Z h o u , M , 1 0 8 2 5 5 ,
```

PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
120102	Tom	Zhou	M	108255	Sales Ma	AU

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

Reinitialize PDV

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

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
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

All variables in the PDV are reinitialized.

PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
.				.		

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
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
1	2	0	1	0	2	,	T	o	m	,	Z	h	o	u	,	M	,	1	0	8	2	5	5	,

PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
.				.		

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

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	2	0	1	0	3	,	W	i	l	s	o	n	,	D	a	w	e	s	,	M	,	8	7	9

PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
.				.		

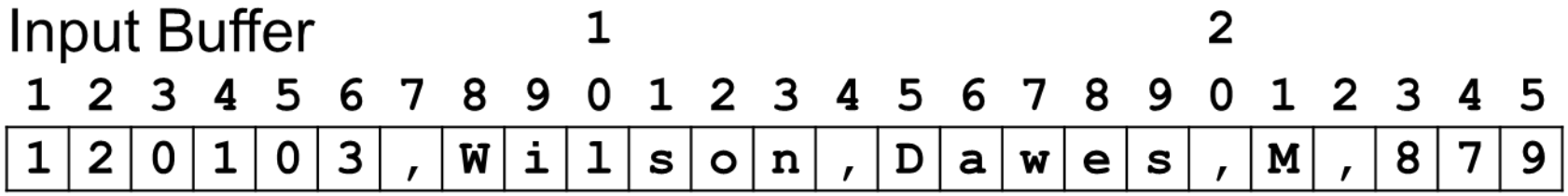
Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Input Buffer



PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
120103	Wilson	Dawes	M	87975	Sales Ma	AU

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
  infile "&path\sales.csv"
         dlm=',';
  input Employee_ID First_Name $
        Last_Name $ Gender $
        Salary Job_Title $
        Country $;
run;
```

Implicit OUTPUT;
Implicit RETURN;

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
1 2 0 1 0 3 , W i l s o n , D a w e s , M , 8 7 9
```

work.subset

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	County
120102	Tom	Zhou	M	108255	Sales Ma	AU
120103	Wilson	Dawes	M	87975	Sales Ma	AU

Execution

Partial **sales.csv**

```
120102,Tom,Zhou, ...
120103,Wilson,Dawes, ...
120121,Irenie,Elvish, ...
120122,Christina,Ngan, ...
120123,Kimiko,Hotstone, ...
120124,Lucian,Daymond, ...
120125,Fong,Hofmeister, ...
```

```
data work.subset;
   infile "$path\sales.csv"
          Continue until EOF
   input Employee_ID First_Name $
         Last_Name $ Gender $
         Salary Job_Title $
         Country $;
run;
```

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	7
1	2	0	1	0	3	,	W	i	l	s	o	n	,	D	a	w	e	s	,	M	,	8	7	9	,	

PDV

Employee_ID N 8	First_Name $ 8	Last_Name $ 8	Gender $ 8	Salary N 8	Job_Title $ 8	Country $ 8
.				.		

Viewing the Output

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

- Partial PROC PRINT Output

Employee_ ID	First_ Name	Last_ Name	Gender	Salary	Job_ Title	Country
120102	Tom	Zhou	M	108255	Sales Ma	AU
120103	Wilson	Dawes	M	87975	Sales Ma	AU
120121	Irenie	Elvish	F	26600	Sales Re	AU
120122	Christin	Ngan	F	27475	Sales Re	AU
120123	Kimiko	Hotstone	F	26190	Sales Re	AU
120124	Lucian	Daymond	M	26480	Sales Re	AU
120125	Fong	Hofmeist	M	32040	Sales Re	AU

Some character values are truncated.

L6_D1.sas

LENGTH Statement

The *LENGTH statement* defines the type and length of a variable.

LENGTH *variable(s)* <$> *length …*;

```
data work.subset;
    length First_Name $ 12 Last_Name $ 18
           Gender $ 1 Job_Title $ 25
           Country $ 2;
    infile "&path\sales.csv" dlm=',';
    input Employee_ID First_Name $ Last_Name $
          Gender $ Salary Job_Title $ Country $;
run;
```

Put the LENGTH statement before the INPUT statement.

L6_D2.sas

Compilation

```
data work.subset;
   length First_Name $ 12 Last_Name $ 18
          Gender $ 1 Job_Title $ 25
          Country $ 2;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

Attributes are based on the LENGTH statement.

PDV

First_ Name $ 12	Last _Name $ 18	Gender $ 1	Job_Title $ 25	Country $ 2

Compilation

```
data work.subset;
   length First_Name $ 12 Last_Name $ 18
          Gender $ 1 Job_Title $ 25
          Country $ 2;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $ Last_Name $
         Gender $ Salary Job_Title $ Country $;
run;
```

PDV

First_ Name $ 12	Last _Name $ 18	Gender $ 1	Job_Title $ 25	Country $ 2	Employee_ ID N 8	Salary N 8

Viewing the Output

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

L6_D2.sas

Partial PROC PRINT Output

First_ Name	Last_Name	Gender	Job_Title	Country	Employee_ ID	Salary
Tom	Zhou	M	Sales Manager	AU	120102	108255
Wilson	Dawes	M	Sales Manager	AU	120103	87975
Irenie	Elvish	F	Sales Rep. II	AU	120121	26600
Christina	Ngan	F	Sales Rep. II	AU	120122	27475
Kimiko	Hotstone	F	Sales Rep. I	AU	120123	26190

The character values are no longer truncated, but the order of the variables changed.

Exercise 3

Suppose you want the order of the variables to match the order of the fields. You can include the numeric variables in the LENGTH statement. Which of the following produces the correct results?

- a.

```
length Employee_ID First_Name $ 12
       Last_Name $ 18 Gender $ 1
       Salary Job_Title $ 25
       Country $ 2;
```

- b.

```
length Employee_ID 8 First_Name $ 12
       Last_Name $ 18 Gender $ 1
       Salary 8 Job_Title $ 25
       Country $ 2;
```

Using a LENGTH Statement

The LENGTH statement identifies the character variables, so dollar signs can be omitted from the INPUT statement.

```
data work.subset;
   length Employee_ID 8 First_Name $ 12
          Last_Name $ 18 Gender $ 1
          Salary 8 Job_Title $ 25
          Country $ 2;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name Last_Name
         Gender Salary Job_Title Country;
run;
```

L6_D3.sas

Viewing the Output

Display the variables in creation order.

```
proc contents data=work.subset varnum;
run;
```

Partial PROC CONTENTS Output

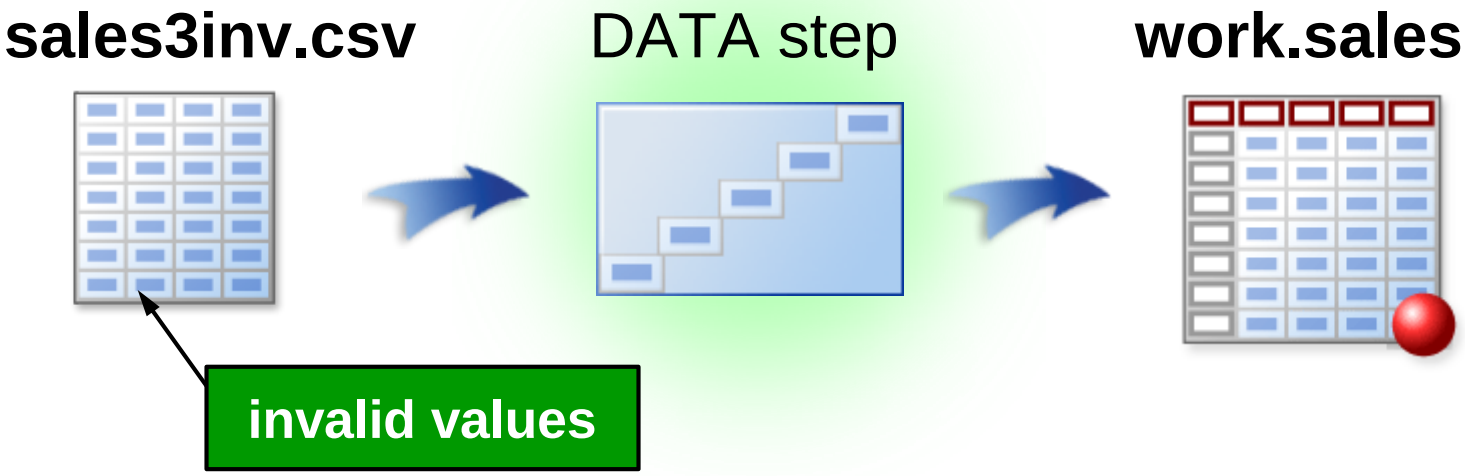
Variables in Creation Order

#	Variable	Type	Len
1	Employee_ID	Num	8
2	First_Name	Char	12
3	Last_Name	Char	18
4	Gender	Char	1
5	Salary	Num	8
6	Job_Title	Char	25
7	Country	Char	2

Viewing the Output

Employee_ ID	First_ Name	Last_Name	Gender	Salary	Job_Title	Country
120102	Tom	Zhou	M	108255	Sales Manager	AU
120103	Wilson	Dawes	M	87975	Sales Manager	AU
120121	Irenie	Elvish	F	26600	Sales Rep. II	AU
120122	Christina	Ngan	F	27475	Sales Rep. II	AU
120123	Kimiko	Hotstone	F	26190	Sales Rep. I	AU
120124	Lucian	Daymond	M	26480	Sales Rep. I	AU
120125	Fong	Hofmeister	M	32040	Sales Rep. IV	AU

A raw data file contains information about Orion Star sales employees. It includes some invalid data values.



Exercise 4

What problems do you see with the data values for the last two data fields, **Salary** and **Country**?

Partial **sales3inv.csv**

```
120102,Tom,Zhou,Manager,108255,AU
120103,Wilson,Dawes,Manager,87975,AU
120121,Irenie,Elvish,Rep. II,26600,AU
120122,Christina,Ngan,Rep. II,n/a,AU
120123,Kimiko,Hotstone,Rep. I,26190,AU
120124,Lucian,Daymond,Rep. I,26480,12
120125,Fong,Hofmeister,Rep. IV,32040,AU
```

Reading a Raw Data File with Data Errors

```
data work.sales;
   infile "&path\sales3inv.csv" dlm=',';
   input Employee_ID First $ Last $
         Job_Title $ Salary Country $;
run;

proc print data=work.sales;
run;
```

L6_D4.sas

Salary is defined as numeric and **Country** as character.

Viewing the Output

Obs	Employee_ ID	First	Last	Job_ Title	Salary	Country
1	120102	Tom	Zhou	Manager	108255	AU
2	120103	Wilson	Dawes	Manager	87975	AU
3	120121	Irenie	Elvish	Rep. II	26600	AU
4	120122	Christin	Ngan	Rep. II	.	AU
5	120123	Kimiko	Hotstone	Rep. I	26190	AU
6	120124	Lucian	Daymond	Rep. I	26480	12
7	120125	Fong	Hofmeist	Rep. IV	32040	AU

- A missing value was stored in **Salary** for the input value *n/a*.
- The value *12* was successfully stored in **Country**.
- A data error occurred on observation 4, but not on observation 6.

Data Errors

A data error occurs when a data value does not match the field specification. The following information is written to the SAS log:

- a note describing the error
- a column ruler
- the input record
- the contents of the PDV

```
NOTE: Invalid data for Salary in line 4 31-33.
RULE:     ----+----1----+----2----+----3----+----4----+----5-
4         120122,Christina,Ngan,Rep. II,n/a,AU 36
Employee_ID=120122 First=Christin Last=Ngan Job_Title=Rep. II Salary=.
Country=AU _ERROR_=1 _N_=4
```

A missing value is assigned to the corresponding variable, and execution continues.

Data Errors

Two temporary variables are created during the processing of every DATA step.

- _**N**_ is the DATA step iteration counter.
- _**ERROR**_ indicates data error status.
  - *0* indicates that no data error occurred on that record.
  - *1* indicates that one or more data errors occurred on that record.

```
NOTE: Invalid data for Salary in line 4 31-33.
RULE:      ----+----1----+----2----+----3----+----4----+----5-
4          120122,Christina,Ngan,Rep. II,n/a,AU 36
Employee_ID=120122 First=Christin Last=Ngan Job_Title=Rep. II Salary=.
Country=AU _ERROR_=1 _N_=4
```

Exercise 5

Submit program **L6_E5.sas** and examine the log.

Which statement best describes the reason for the error?

a. The data in the raw data file is invalid.
b. The programmer incorrectly read the data.

6.1 Introduction to Reading Raw Data Files

6.2 Reading Standard Delimited Data

6.3 Using Informats to Read Delimited Data

6.4 Handling Missing Data

Objectives – Part 3

- Use informats to read character data.
- Use informats to read nonstandard data.
- Subset observations and add permanent attributes.

Create a temporary SAS data set by reading both standard and nonstandard values from a comma-delimited raw data file.

The new data set should contain a subset of the input data and include permanent attributes.

Considerations

Use modified list input to read all the fields from **sales.csv**. Store the date fields as SAS dates.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1968,10/01/1989
```

Modified List Input

This DATA step uses *modified list input.* Instead of a LENGTH statement, an informat specifies the length for each character variable.

input *variable* <**:***informat.*> …;

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name :$12.
         Last_Name :$18. Gender :$1. Salary
         Job_Title :$25. Country :$2.;
run;
```

- The **$12.** informat defines a length of 12 for **First_Name** and enables up to 12 characters to be read.
- The **:** format modifier tells SAS to read until it encounters a delimiter.

L6_D5.sas

Modified List Input

- ⚠ Omitting the colon modifier causes unexpected results.


```
input Employee_ID First_Name $12.
      Last_Name :$18. Gender :$1. Salary
      Job_Title :$25. Country :$2.;
```

PDV

Employee_ID N 8	First_Name $ 12	Last_Name $ 18	Gender $ 1
120102	Tom,Zhou,M,1	08255	S

Salary N 8	Job_Title $ 25	Country $ 2
.	11AUG1973	06

Reading Nonstandard Data

An informat is ***required*** to read nonstandard numeric data.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1968,10/01/1989
```

In this example, informats are needed to specify the style of the date fields so that they can be read and converted to SAS dates.

Exercise 6

- A *format* is an instruction that tells SAS how to display data values. What formats could you specify to display a SAS date in the styles shown below?

a) 01JAN2000

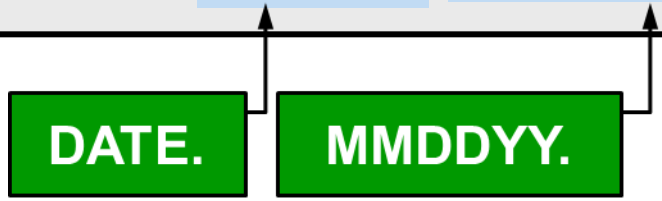
b) 01/16/2000

Informats for Nonstandard Data

An *informat* is an instruction that SAS uses to ***read*** data values into a variable.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1968,10/01/1989
```



The informat describes the data value and tells SAS how to convert it.

SAS Informats

- SAS informats have the following form:

<$><*informat*><*w*>.

$	Indicates a character informat.
informat	Names the SAS informat or user-defined informat.
w	Specifies the width or number of columns to read or specifies the length of a character variable.
.	Is required syntax.

- The width is typically not used with list input because SAS reads each field until it encounters a delimiter.

SAS Informats

Informat	Definition
COMMA. DOLLAR.	Reads nonstandard numeric data and removes embedded commas, blanks, dollar signs, percent signs, and dashes.
COMMAX. DOLLARX.	Reads nonstandard numeric data and removes embedded non-numeric characters; reverses the roles of the decimal point and the comma.
EUROX.	Reads nonstandard numeric data and removes embedded non-numeric characters in European currency.
$CHAR.	Reads character values and preserves leading blanks.
$UPCASE.	Reads character values and converts them to uppercase.

SAS Informats

Informats are used to read and convert raw data.

Informat	Raw Data Value	SAS Data Value
COMMA. DOLLAR.	*$12,345*	*12345*
COMMAX. DOLLARX.	*$12.345*	*12345*
EUROX.	*€12.345*	*12345*
$CHAR.	*##Australia*	*##Australia*
$UPCASE.	*au*	*AU*

The character # represents a blank space.

SAS Informats

Use date informats to read and convert dates to SAS date values.

Informat	Raw Data Value	SAS Data Value
MMDDYY.	*010160* *01/01/60* *01/01/1960* *1/1/1960*	*0*
DDMMYY.	*311260* *31/12/60* *31/12/1960*	*365*
DATE.	*31DEC59* *31DEC1959*	*-1*

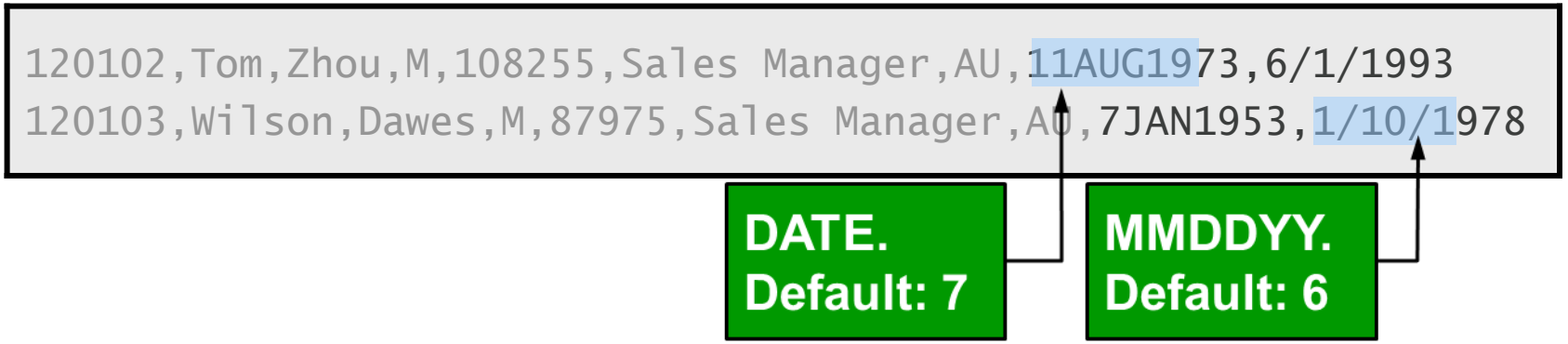
Exercise 7

Use the SAS Help Facility or documentation to investigate the **DATE*w*.** informat and answer the following questions:

a) What does the ***w*** represent?

b) What is the default width of this informat?

Using Informats to Read Nonstandard Data



- An informat is needed to read a nonstandard value.
- The width is optional when you use list input.
- If the width and the colon format modifier are omitted, the default width for that informat is used.

Modified List Input

The colon format modifier (:) tells SAS to read until it encounters a delimiter.

```
input Employee_ID First_Name :$12.
      Last_Name :$18. Gender :$1.
      Salary Job_Title :$25. Country :$2.
      Birth_Date :date. Hire_Date :mmddyy.;
```

INPUT *variable* <**$**> *variable* <**:***informat*> …**;**

colon format modifier

L6_D6.sas

Viewing the Log

```
37   data work.sales;
38      infile "&path\sales.csv" dlm=',';
39      input Employee_ID First_Name :$12. Last_Name :$18.
40            Gender :$1. Salary Job_Title :$25. Country :$2.
41            Birth_Date :date. Hire_Date :mmddyy.;
42   run;

NOTE: The infile "s:\workshop\sales.csv" is:
      Filename=s:\workshop\sales.csv,
      RECFM=V,LRECL=256,File Size (bytes)=11340,

NOTE: 165 records were read from the infile "s:\workshop\sales.csv".
NOTE: The data set WORK.SALES has 165 observations and 9 variables.
```

L6_D6.sas

Viewing the Output

```
proc print data=work.sales;
run;
```

- Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Gender	Job_Title	Country	Employee_ ID	Salary	Birth_ Date	Hire_ Date
1	Tom	Zhou	M	Sales Manager	AU	120102	108255	4971	12205
2	Wilson	Dawes	M	Sales Manager	AU	120103	87975	-2535	6575
3	Irenie	Elvish	F	Sales Rep. II	AU	120121	26600	-4169	6575
4	Christina	Ngan	F	Sales Rep. II	AU	120122	27475	-523	8217
5	Kimiko	Hotstone	F	Sales Rep. I	AU	120123	26190	3193	10866

L6_D6.sas

Additional SAS Statements

Additional SAS statements can be added to perform further processing in the DATA step.

```
data work.sales;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name :$12. Last_Name :$18.
         Gender :$1. Salary Job_Title :$25. Country :$2.
         Birth_Date :date. Hire_Date :mmddyy.;
   if Country='AU';
   keep First_Name Last_Name Salary
        Job_Title Hire_Date;
   label Job_Title='Sales Title'
         Hire_Date='Date Hired';
   format Salary dollar12. Hire_Date monyy7.;
run;
```

A subsetting IF statement is used to subset observations when SAS reads from a raw data file.

L6_D7.sas

Viewing the Output

```
proc print data=work.sales label;
run;
```

Partial PROC PRINT Output

Obs	First_ Name	Last_Name	Sales Title	Salary	Date Hired
1	Tom	Zhou	Sales Manager	$108,255	JUN1993
2	Wilson	Dawes	Sales Manager	$87,975	JAN1978
3	Irenie	Elvish	Sales Rep. II	$26,600	JAN1978
4	Christina	Ngan	Sales Rep. II	$27,475	JUL1982
5	Kimiko	Hotstone	Sales Rep. I	$26,190	OCT1989

L6_D7.sas

WHERE versus Subsetting IF Statement

Step and Usage	WHERE	IF
PROC step	Yes	No
DATA step (source of variable)		
SET statement	Yes	Yes
assignment statement	No	Yes
INPUT statement	No	Yes

You are working on a new project, but the raw data file is not created yet. You can include in-stream data in a DATA step.

DATALINES Statement

The DATALINES statement supplies data within a

```
data work.newemps;
   input First_Name :$12. Last_Name :$18.
         Job_Title :$15. Salary :dollar.;
datalines;
Steven Worton Auditor $40,450
Marta-Lyn Bamberger Manager $32,000
Merle Hieds Trainee $24,025
;
```

```
DATALINES;
...
;
```

- DATALINES is the last statement in the DATA step and immediately precedes the first data line.
- A null statement (a single semicolon) indicates the end of the input data.

Viewing the Output

```
proc print data=work.newemps;
run;
```

PROC PRINT Output

Obs	First_ Name	Last_Name	Job_ Title	Salary
1	Steven	Worton	Auditor	40450
2	Marta-Lyn	Bamberger	Manager	32000
3	Merle	Hieds	Trainee	24025

L6_D8.sas

6.1 Introduction to Reading Raw Data Files

6.2 Reading Standard Delimited Data

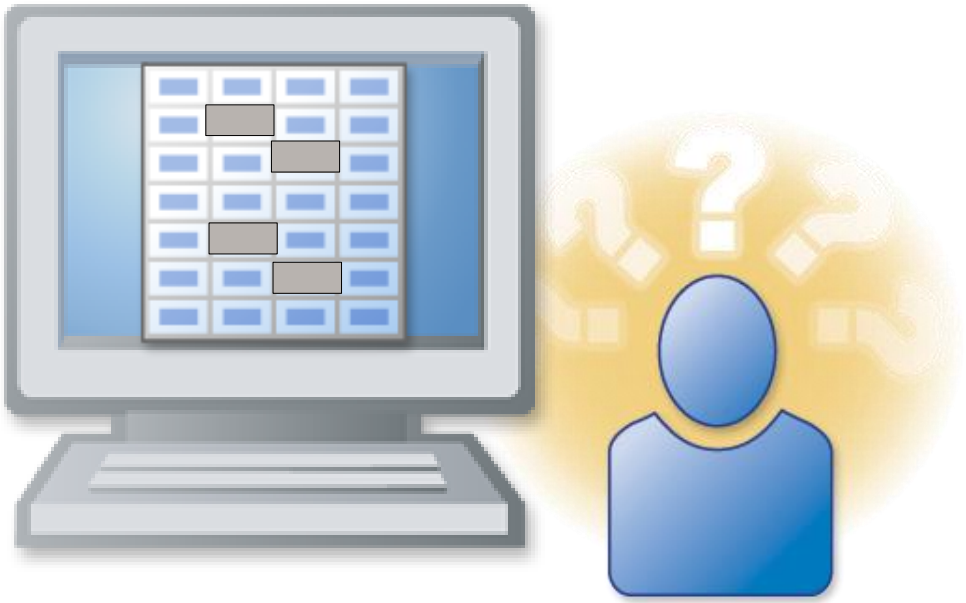
6.3 Using Informats to Read Delimited Data

6.4 Handling Missing Data

Objectives – Part 4

- Use the DSD option to read consecutive delimiters as missing values.
- Use the MISSOVER option to recognize missing values at the end of a record.

Orion Star programmers discovered that some files have records with missing data in one or more fields.



Missing Values in the Middle of the Record

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

Missing data is indicated by consecutive delimiters.

phone2.csv

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

Consecutive Delimiters in List Input

List input treats two or more consecutive delimiters as a single delimiter and not as a missing value.

Partial phone2.csv

```
          1    1    2    2    3    3    4    4
1---5----0----5----0----5----0----5----0----5
Sandrina Stephano,, (919) 271-4592
Cornelia Krahl,(212) 891-3241,(212) 233-5413
```

When there is missing data in a record, SAS does the following:

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

Exercise 8

- Submit **L6_E8.sas** and examine the log and output.
- How many input records were read and how many observations were created?
- Does the output look correct?

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

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

L6_E8.sas

DSD Option

Use the DSD option to correctly read **phone2.csv**.

```
data work.contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile "&path\phone2.csv" dsd;
   input Name $ Phone $ Mobile $;
run;
```

INFILE "*raw-data-file***" <DLM=> DSD**;

The DSD option does the following:

- 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 enclosed in quotation marks

L6_D9.sas

Viewing the Output

Adding the DSD option gives the correct results.

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

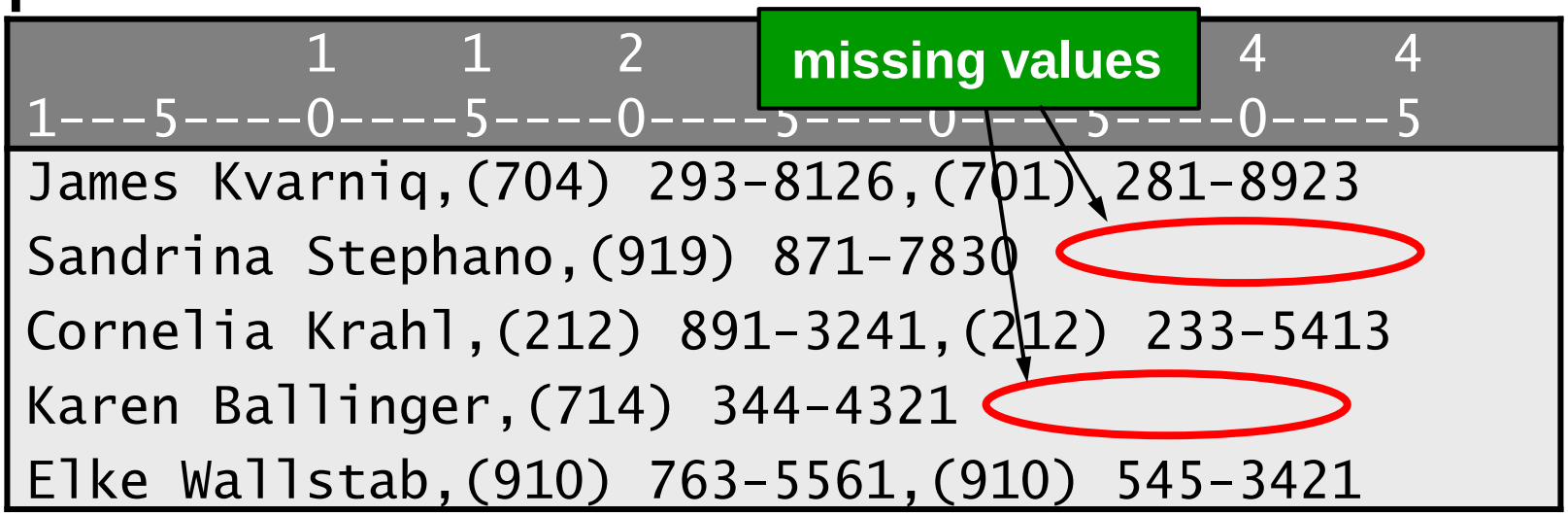
```
NOTE: 5 records were read from the infile "S:\workshop\phone2.csv".
      The minimum record length was 31.
      The maximum record length was 44.
NOTE: The data set WORK.CONTACTS has 5 observations and 3
variables.
```

Orion Star programmers discovered that some files have observations with missing data at the end of the record. As a result, there are fewer fields in the record than specified in the INPUT statement.

Missing Values at the End of a Record

The raw data file **phone.csv** contains missing values at the end of some records.

phone.csv



The DSD option is not appropriate because the missing data is not marked by consecutive delimiters.

MISSOVER Option

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

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

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

INFILE "*raw-data-file***" <DLM=> MISSOVER;**

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

L6_D10.sas

Viewing the Output

```
NOTE: 5 records were read from the infile "S:\workshop\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.
```

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

INFILE Options

INFILE "*raw-data-file***" <DLM=> <DSD> <MISSOVER>;**

Option	Description
DLM=	Specifies an alternate delimiter.
DSD	Sets the default delimiter to a comma, treats consecutive delimiters as missing values, and allows embedded delimiters when the data value is enclosed in quotation marks.
MISSOVER	Sets variables to missing if the end of the record is reached before finding values for all fields.