Chapter 17

Reading Free-Format Data

Overview

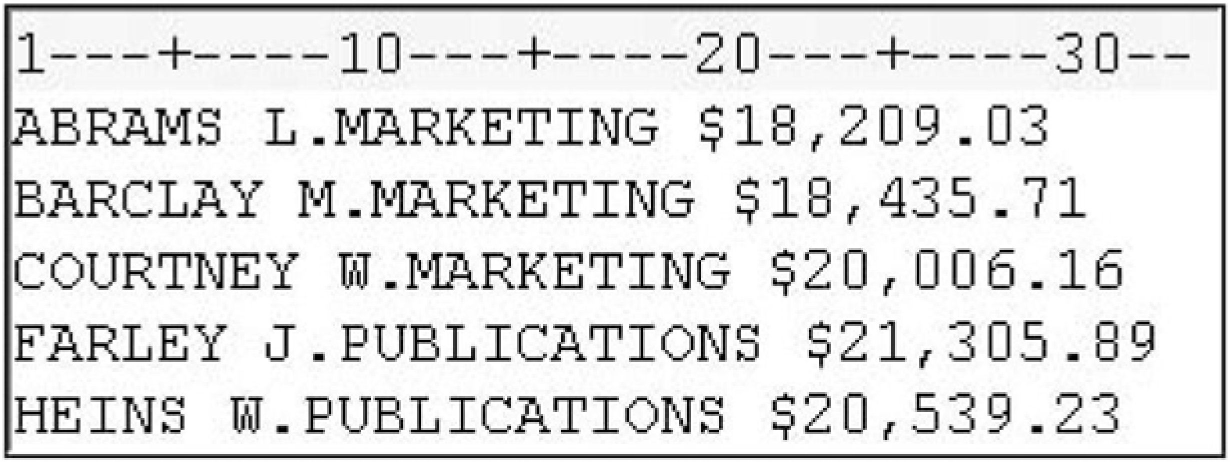
This chapter teaches you how to use **list input** to read free-format data that is not arranged in fixed fields. You will learn to use the INPUT statement with list input to read:

- normal free-format data.
- free-format data that is separated by nonblank delimiters, such as commas.
- free-format data that contains missing values.
- character values that exceed eight characters.
- nonstandard free-format data.
- character values that contain embedded blanks.

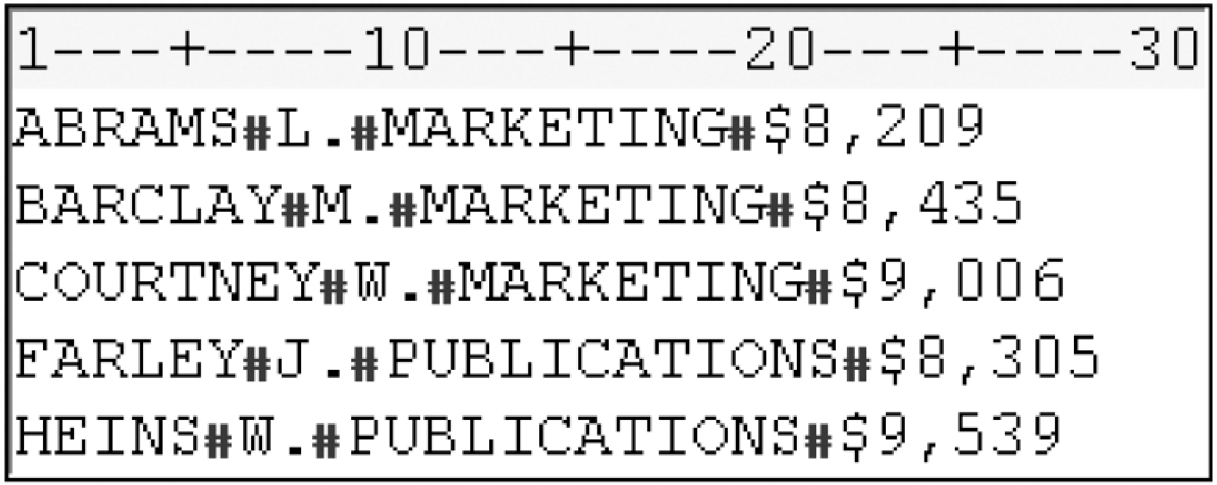
You will also learn how to mix column, formatted, and list input styles in a single INPUT statement.

Free-Format Data

The fields of free-format data are often separated by blanks or by other delimiter, as shown below



```
1---+----10---+----20---+----30--
ABRAMS L.MARKETING $18,209.03
BARCLAY M.MARKETING $18,435.71
COURTNEY W.MARKETING $20,006.16
FARLEY J.PUBLICATIONS $21,305.89
HEINS W.PUBLICATIONS $20,539.23
```

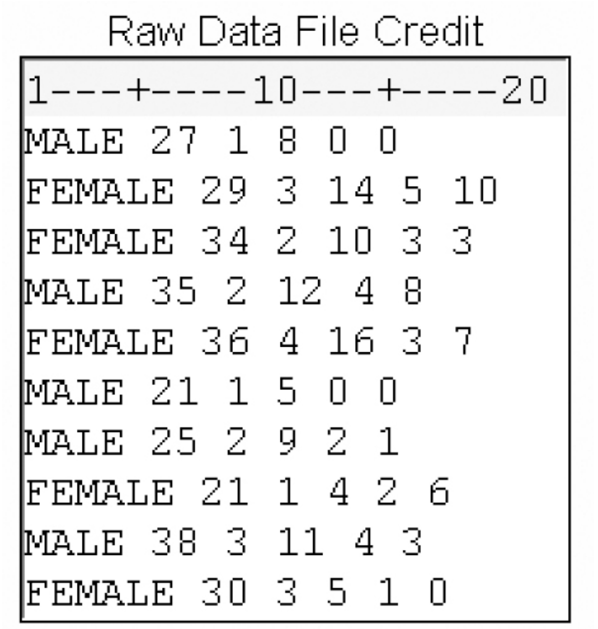


```
1---+----10---+----20---+----30
ABRAMS#L.#MARKETING#$8,209
BARCLAY#M.#MARKETING#$8,435
COURTNEY#W.#MARKETING#$9,006
FARLEY#J.#PUBLICATIONS#$8,305
HEINS#W.#PUBLICATIONS#$9,539
```

Using List Input

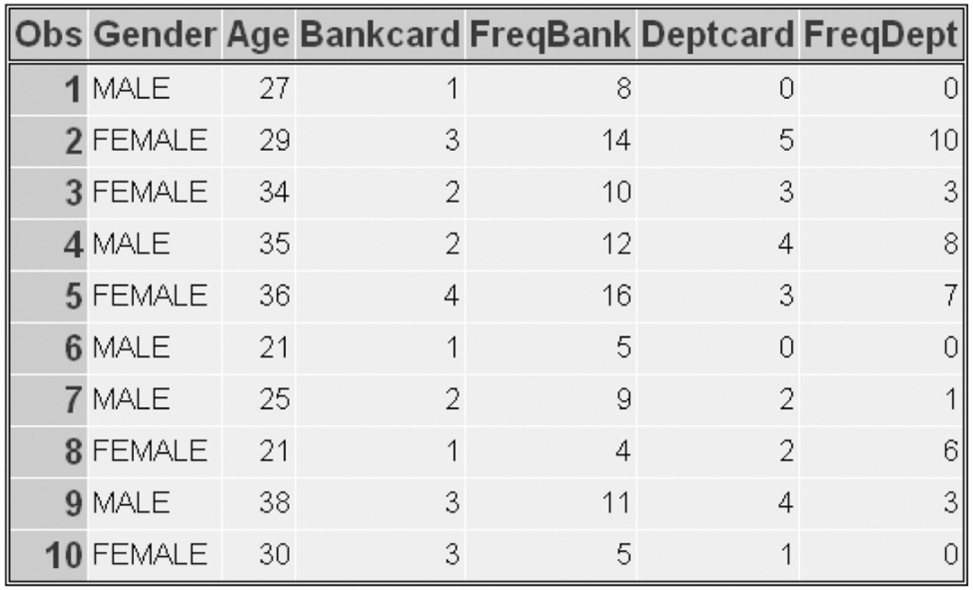
List input is a powerful tool for reading both standard and nonstandard free-format data. By default, list input does not specify column locations; all fields must be separated by at least one blank or other delimiters; fields must be read in order from left to right; you cannot skip or re-read fields.

```
data sasuser.creditsurvey;
  infile credit;
  input Gender $ Age Bankcard FreqBank Deptcard FreqDept;
run;
```



Raw Data File Credit

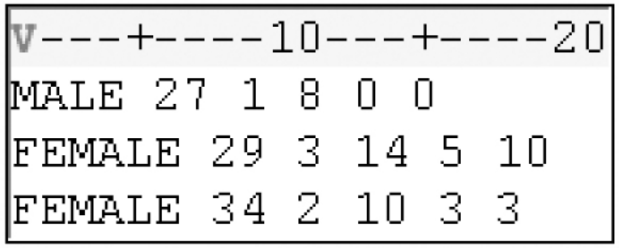
```
1---+----10---+----20
MALE 27 1 8 0 0
FEMALE 29 3 14 5 10
FEMALE 34 2 10 3 3
MALE 35 2 12 4 8
FEMALE 36 4 16 3 7
MALE 21 1 5 0 0
MALE 25 2 9 2 1
FEMALE 21 1 4 2 6
MALE 38 3 11 4 3
FEMALE 30 3 5 1 0
```



Obs	Gender	Age	Bankcard	FreqBank	Deptcard	FreqDept
1	MALE	27	1	8	0	0
2	FEMALE	29	3	14	5	10
3	FEMALE	34	2	10	3	3
4	MALE	35	2	12	4	8
5	FEMALE	36	4	16	3	7
6	MALE	21	1	5	0	0
7	MALE	25	2	9	2	1
8	FEMALE	21	1	4	2	6
9	MALE	38	3	11	4	3
10	FEMALE	30	3	5	1	0

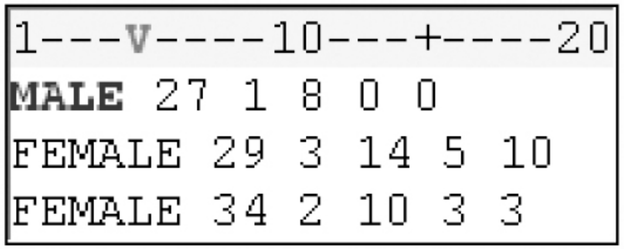
Processing List Input

List input scans the input lines for **values** rather than reading from specific columns. When the INPUT statement is submitted for processing, the input pointer is positioned at column 1 of the raw data file.



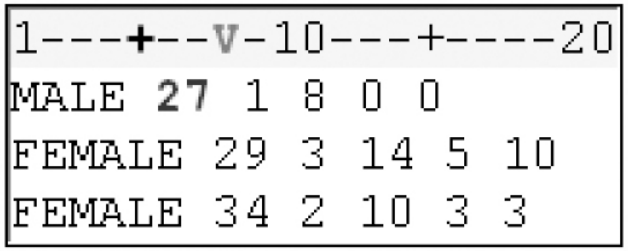
```
V---+----10---+----20
MALE 27 1 8 0 0
FEMALE 29 3 14 5 10
FEMALE 34 2 10 3 3
```

SAS reads the first field until it encounters a **blank** space (or other delimiter if any), which indicates the end of the field, and the data value is assigned to the PDV for the first variable in the INPUT statement.



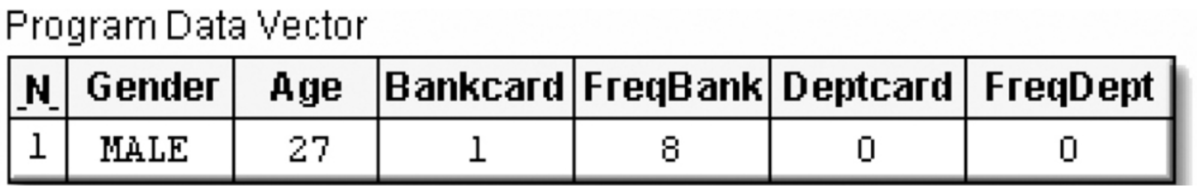
```
1---V----10---+----20
MALE 27 1 8 0 0
FEMALE 29 3 14 5 10
FEMALE 34 2 10 3 3
```

Next, SAS scans the record until the next non-delimiter is found, and the second value is read until another delimiter is encountered. Then the value is assigned to its corresponding variable in the PDV.



```
1---+--V-10---+----20
MALE 27 1 8 0 0
FEMALE 29 3 14 5 10
FEMALE 34 2 10 3 3
```

This process of scanning ahead to the next non-delimiter column, reading the data value until a delimiter is encountered, and assigning the value to a variable in the PDV continues until all of the fields have been read and values have been assigned to variables in the PDV.



Program Data Vector

_N_	Gender	Age	Bankcard	FreqBank	Deptcard	FreqDept
1	MALE	27	1	8	0	0

What are the lengths of the variables created?

```
proc  contents data= sasuser.creditsurvey;
Run;
```

#	Variable	Type	Len
1	Gender	Char	8
2	Age	Num	8
3	Bankcard	Num	8
4	FreqBank	Num	8
5	Deptcard	Num	8
6	FreqDept	Num	8

Working with Delimiters

Most free-format data fields are separated by blanks (default), but fields can also be separated by other delimiters, such as commas. You can tell SAS which field delimiter to use. Use the DLM= option in the INFILE statement to specify a delimiter other than a blank. DELIMITER is an alias for the DLM option.

General form : **DLM** = *delimiter*(*s*)

where *delimiter*(*s*) specifies a delimiter(s) for list input in either of the following forms:

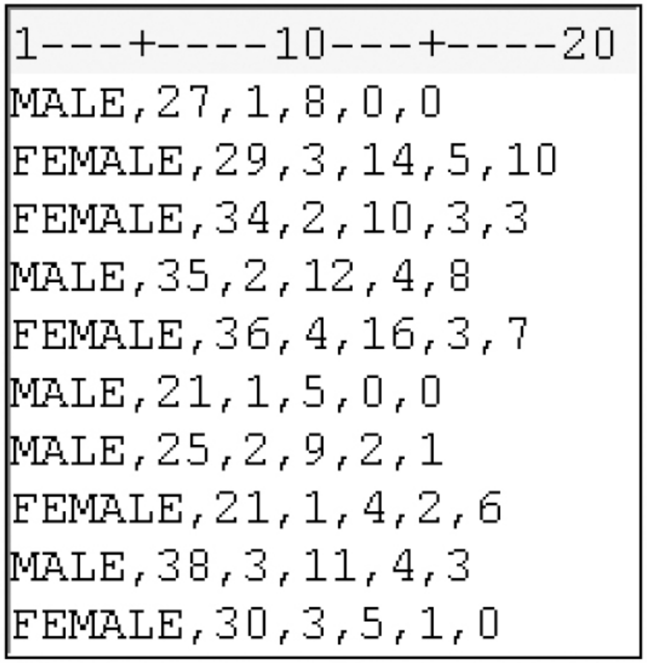
- *'list-of-delimiting-characters'* specifies one or more characters (up to 200) to read as delimiters. The list of characters must be enclosed in quotation marks.
- *character-variable* whose value becomes the delimiter.

Example: Working with Delimiters

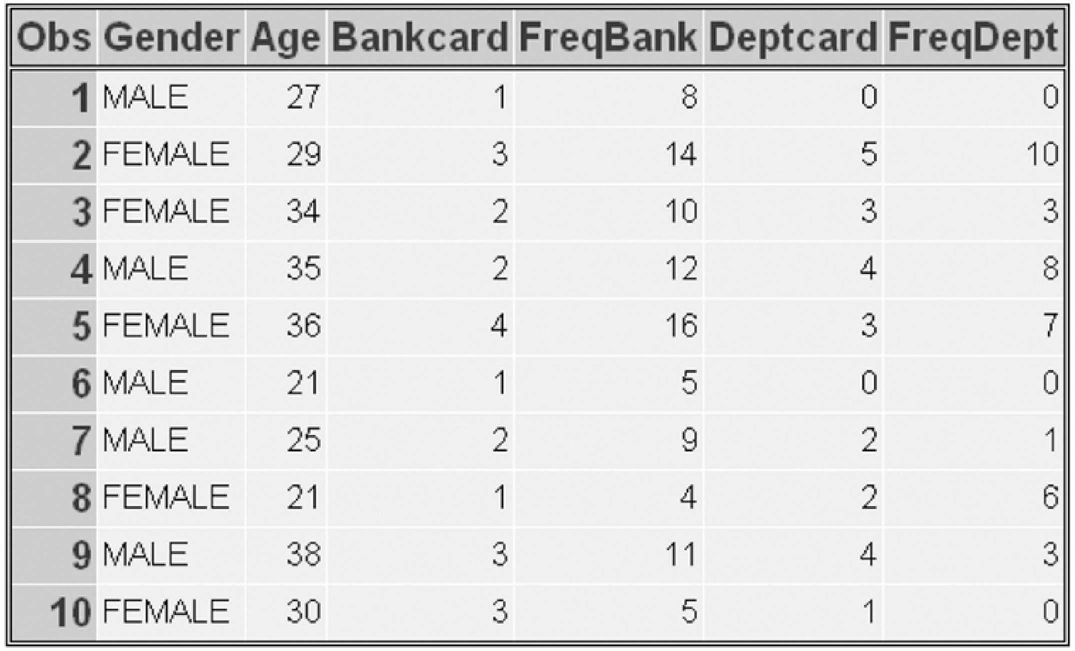
```
data sasuser.creditsurvey;
    infile credit dlm=',#*&';
    input Gender $ Age Bankcard
        FreqBank Deptcard FreqDept;
run;
proc print data=sasuser.creditsurvey;
run;
```

Note:
The field delimiter must *not* be a character that occurs in a data value; otherwise, the fields are identified incorrectly.

Raw Data File Credit



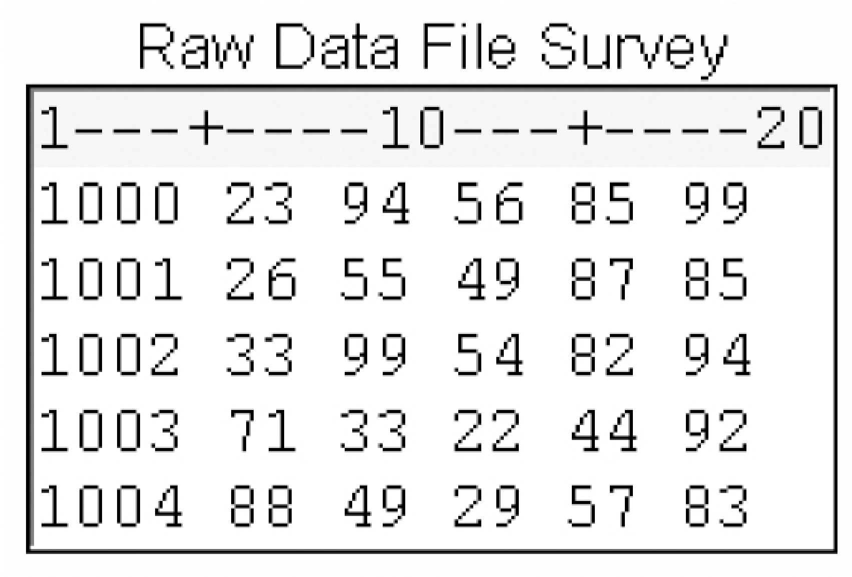
```
1---+----10---+----20
MALE,27,1,8,0,0
FEMALE,29,3,14,5,10
FEMALE,34,2,10,3,3
MALE,35,2,12,4,8
FEMALE,36,4,16,3,7
MALE,21,1,5,0,0
MALE,25,2,9,2,1
FEMALE,21,1,4,2,6
MALE,38,3,11,4,3
FEMALE,30,3,5,1,0
```



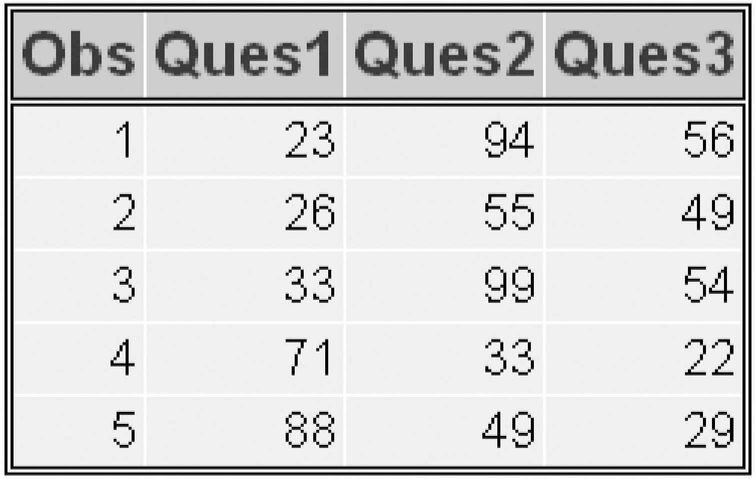
Obs	Gender	Age	Bankcard	FreqBank	Deptcard	FreqDept
1	MALE	27	1	8	0	0
2	FEMALE	29	3	14	5	10
3	FEMALE	34	2	10	3	3
4	MALE	35	2	12	4	8
5	FEMALE	36	4	16	3	7
6	MALE	21	1	5	0	0
7	MALE	25	2	9	2	1
8	FEMALE	21	1	4	2	6
9	MALE	38	3	11	4	3
10	FEMALE	30	3	5	1	0

Reading a Range of Variables

```
data sasuser.phonesurvey;
     infile phonesurvey;
     input IDnum $ Ques1-Ques5;
run;
proc print data=sasuser.phonesurvey;
       var ques1-ques3;
run;
```


Raw Data File Survey

```
1---+----10---+----20
1000 23 94 56 85 99
1001 26 55 49 87 85
1002 33 99 54 82 94
1003 71 33 22 44 92
1004 88 49 29 57 83
```



Obs	Ques1	Ques2	Ques3
1	23	94	56
2	26	55	49
3	33	99	54
4	71	33	22
5	88	49	29

Reading a Range of Variables

- When specifying a range of character variables, **both** the variable list and the $ sign must be enclosed in parentheses:

```
data  survey.stores;
      infile stordata;
      input Age (Store1-Store3) ($);
run;
```

- When specifying a range using formatted input, **both** the **variable list** and the **informat** must be enclosed in parentheses, regardless of the variable's type:

```
data  test.scores;
      infile group3;
      input Age (Score1-Score4) (6.);
run;
```

Limitations of List Input

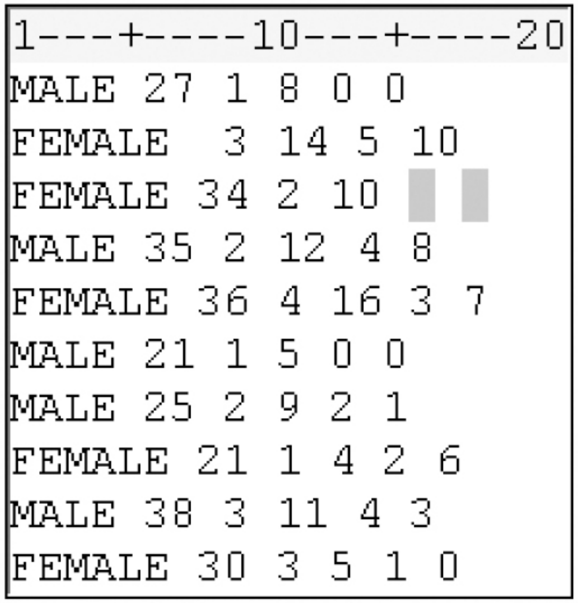
In its default form, list input places several restrictions on the types of data that can be read:

- Both character and numeric variables have a default length of **8** bytes. Character values that are longer than eight characters will be truncated.
- Data must be in standard numeric or character format.
- Character values cannot contain embedded delimiters.
- Missing numeric and character values must be represented by a place holder (period or some other character).

There are ways to work around these limitations by using **modified list input**.

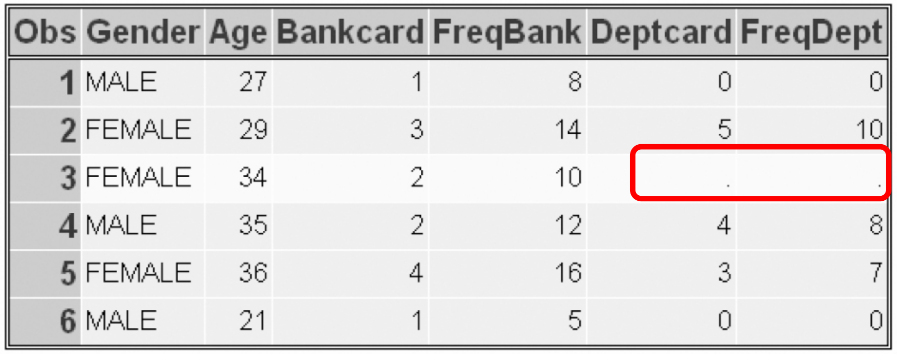
List Input With Missing Data

When missing data appear at the end of the record and no placeholder is present, you can use the MISSOVER option in the INFILE statement to assign the missing values to variables with missing data. The MISSOVER option prevents SAS from reading the next record.



```
1---+----10---+----20
MALE 27 1 8 0 0
FEMALE  3 14 5 10
FEMALE 34 2 10
MALE 35 2 12 4 8
FEMALE 36 4 16 3 7
MALE 21 1 5 0 0
MALE 25 2 9 2 1
FEMALE 21 1 4 2 6
MALE 38 3 11 4 3
FEMALE 30 3 5 1 0
```

```
data sasuser.creditsurvey;
  infile credit missover;
  input Gender $ Age Bankcard FreqBank
      Deptcard FreqDept;
run;
proc print data=sasuser.creditsurvey;
run;
```

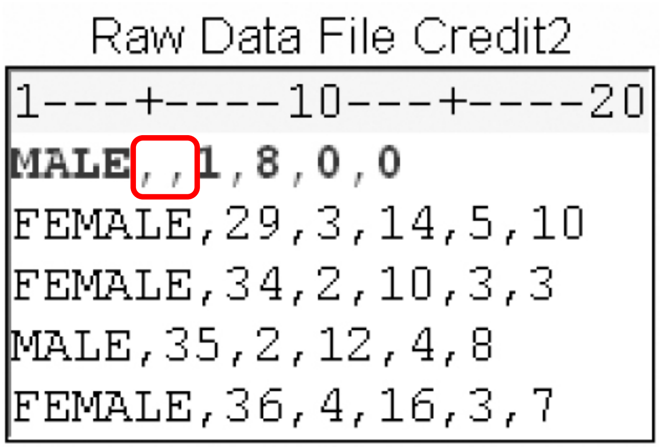


Obs	Gender	Age	Bankcard	FreqBank	Deptcard	FreqDept
1	MALE	27	1	8	0	0
2	FEMALE	29	3	14	5	10
3	FEMALE	34	2	10	.	.
4	MALE	35	2	12	4	8
5	FEMALE	36	4	16	3	7
6	MALE	21	1	5	0	0

List Input With Missing Data

When missing data appear in the middle of a record, you can use the Delimiter Sensitive Data (**DSD**) option in the INFILE statement to correctly read the raw data. The DSD option changes how SAS treats delimiters. Specifically, the DSD option

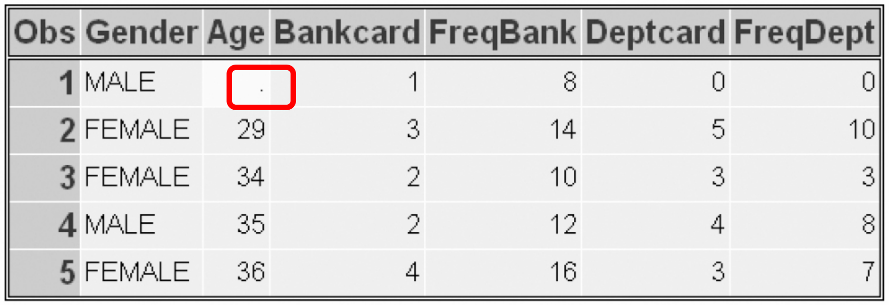
- sets the default delimiter to a comma, treats two consecutive delimiters as a missing value,
- removes quotation marks (if any) from values.



Raw Data File Credit2

```
1---+----10---+----20
MALE,,1,8,0,0
FEMALE,29,3,14,5,10
FEMALE,34,2,10,3,3
MALE,35,2,12,4,8
FEMALE,36,4,16,3,7
```

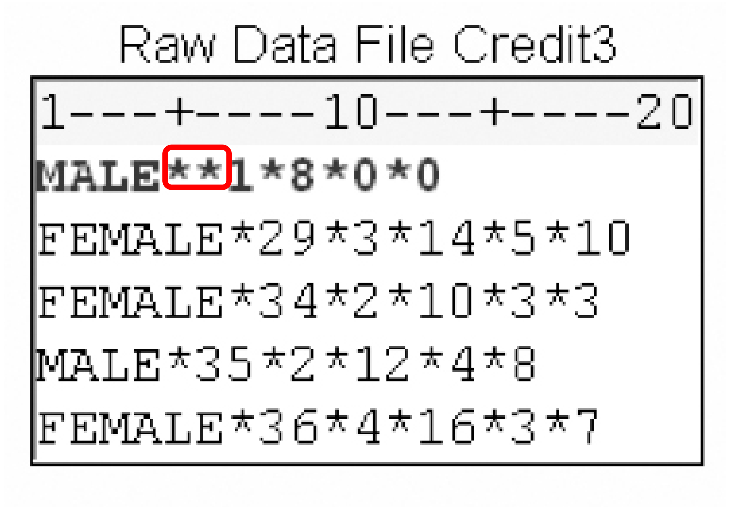
```
data sasuser.creditsurvey;
   infile credit dsd;
   input Gender $ Age Bankcard FreqBank
         Deptcard FreqDept;
run;
proc print data=sasuser.creditsurvey;
run;
```



Obs	Gender	Age	Bankcard	FreqBank	Deptcard	FreqDept
1	MALE	.	1	8	0	0
2	FEMALE	29	3	14	5	10
3	FEMALE	34	2	10	3	3
4	MALE	35	2	12	4	8
5	FEMALE	36	4	16	3	7

List Input With Missing Data: Use DSD With DLM=

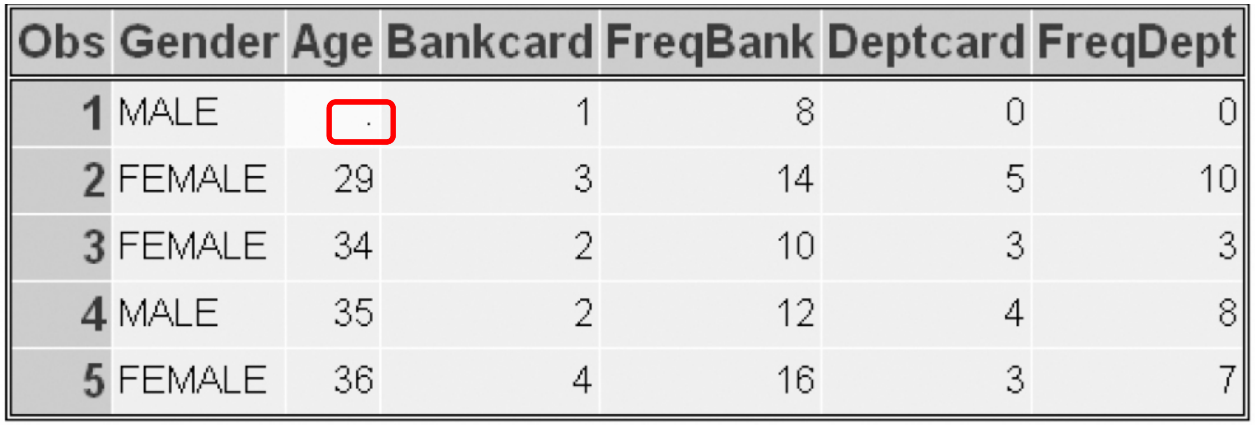
If the data uses **multiple delimiters** or a single delimiter **other than a comma**, specify the delimiter value(s) with the DLM= option.



Raw Data File Credit3

```
1---+----10---+----20
MALE**1*8*0*0
FEMALE*29*3*14*5*10
FEMALE*34*2*10*3*3
MALE*35*2*12*4*8
FEMALE*36*4*16*3*7
```

```
data sasuser.creditsurvey;
  infile credit3 dsd DLM='*';
  input Gender $ Age Bankcard FreqBank
        Deptcard FreqDept;
run;
proc print data=sasuser.creditsurvey;
run;
```



Obs	Gender	Age	Bankcard	FreqBank	Deptcard	FreqDept
1	MALE	.	1	8	0	0
2	FEMALE	29	3	14	5	10
3	FEMALE	34	2	10	3	3
4	MALE	35	2	12	4	8
5	FEMALE	36	4	16	3	7

List Input With Missing Data: Use DSD With DLM=

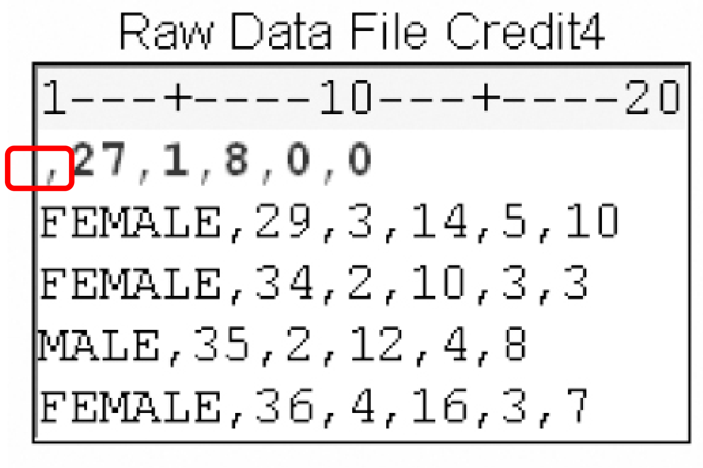
You can still use the DSD and DLM= options to read fields that are delimited by blanks.

```
data sasuser.creditsurvey;
  infile credit5 dsd DLM=' ';
  input Gender $ Age Bankcard FreqBank
       Deptcard FreqDept;
run;
proc print data=sasuser.creditsurvey;
run;
```

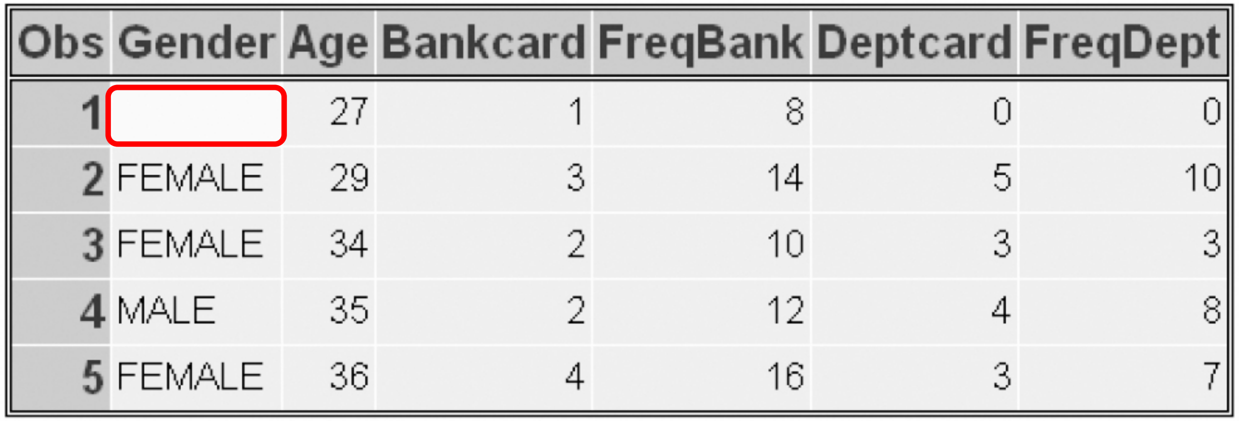
List Input With Missing Data: Use DSD With DLM=

The DSD option can also be used to read raw data when there is a missing value at the beginning of a record, as long as a delimiter precedes the first value in the record.

```
data sasuser.creditsurvey;
  infile credit4 dsd;
  input Gender $ Age Bankcard FreqBank
       Deptcard FreqDept;
run;
proc print data=sasuser.creditsurvey;
run;
```


Raw Data File Credit4

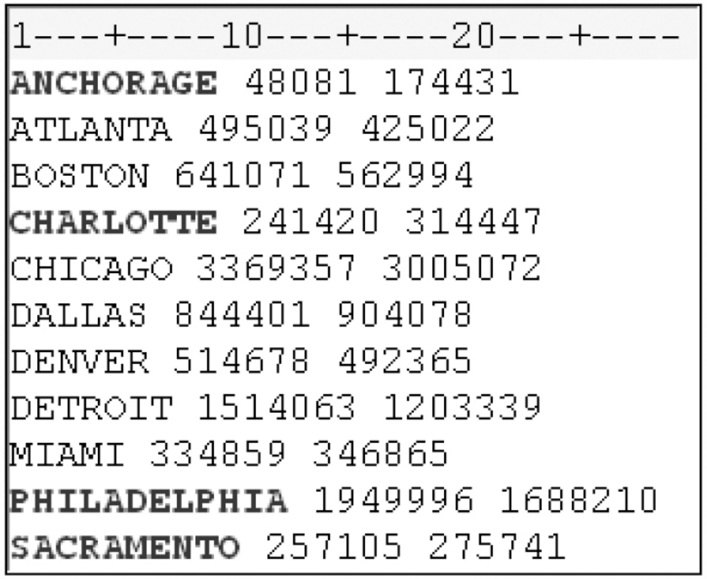
```
1---+----10---+----20
,27,1,8,0,0
FEMALE,29,3,14,5,10
FEMALE,34,2,10,3,3
MALE,35,2,12,4,8
FEMALE,36,4,16,3,7
```



Obs	Gender	Age	Bankcard	FreqBank	Deptcard	FreqDept
1		27	1	8	0	0
2	FEMALE	29	3	14	5	10
3	FEMALE	34	2	10	3	3
4	MALE	35	2	12	4	8
5	FEMALE	36	4	16	3	7

Specifying the Length of Character Variables

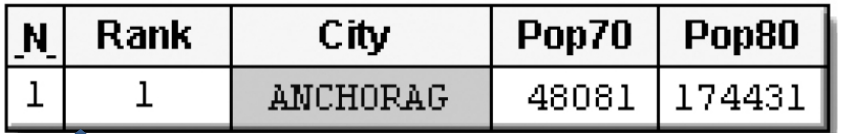
When you use list input to read raw data, character variables are assigned a default length of **8**. Let's see what happens when list input is used to read character variables whose values are longer than 8.



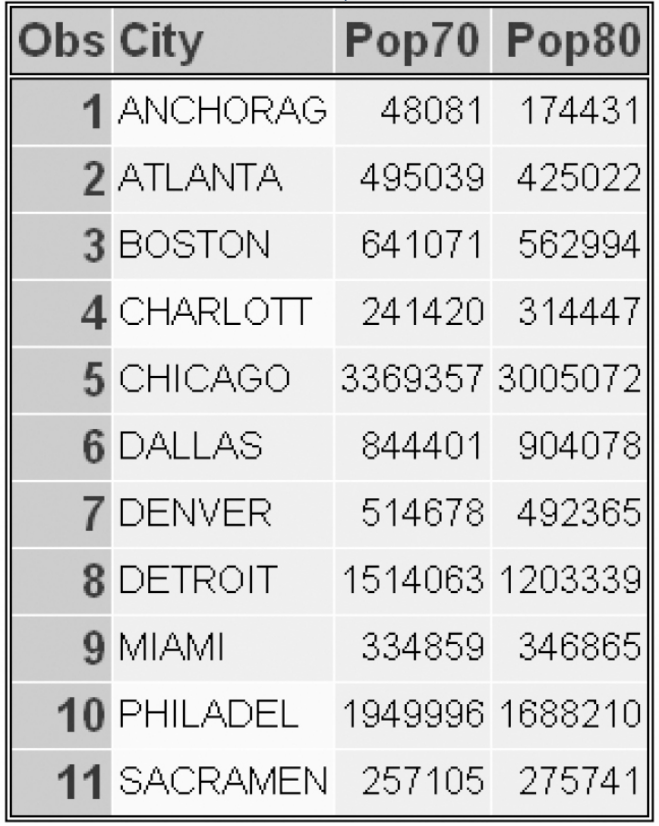
```
1---+----10---+----20---+----
ANCHORAGE 48081 174431
ATLANTA 495039 425022
BOSTON 641071 562994
CHARLOTTE 241420 314447
CHICAGO 3369357 3005072
DALLAS 844401 904078
DENVER 514678 492365
DETROIT 1514063 1203339
MIAMI 334859 346865
PHILADELPHIA 1949996 1688210
SACRAMENTO 257105 275741
```

```
data sasuser.growth;
  infile citydata;
  input City $ Pop70 Pop80;
run;
proc print data=sasuser.growth;
run;
```

Longer character values are truncated when read into PDV



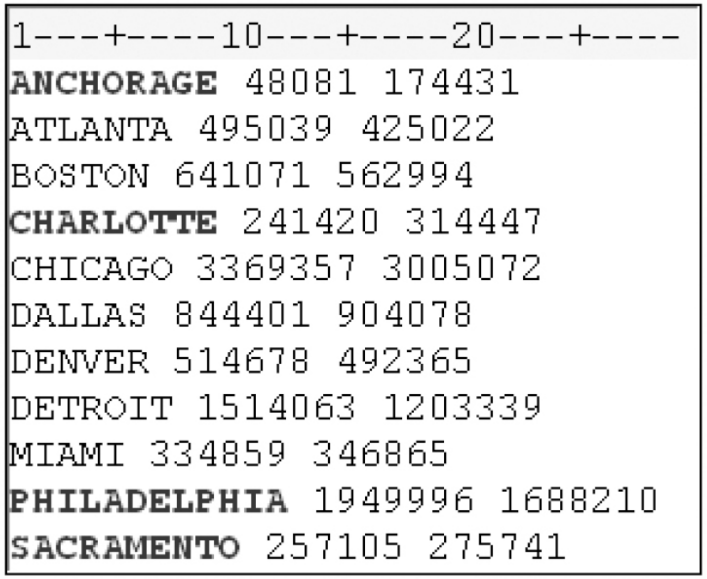
N	Rank	City	Pop70	Pop80
1	1	ANCHORAG	48081	174431



Obs	City	Pop70	Pop80
1	ANCHORAG	48081	174431
2	ATLANTA	495039	425022
3	BOSTON	641071	562994
4	CHARLOTT	241420	314447
5	CHICAGO	3369357	3005072
6	DALLAS	844401	904078
7	DENVER	514678	492365
8	DETROIT	1514063	1203339
9	MIAMI	334859	346865
10	PHILADEL	1949996	1688210
11	SACRAMEN	257105	275741

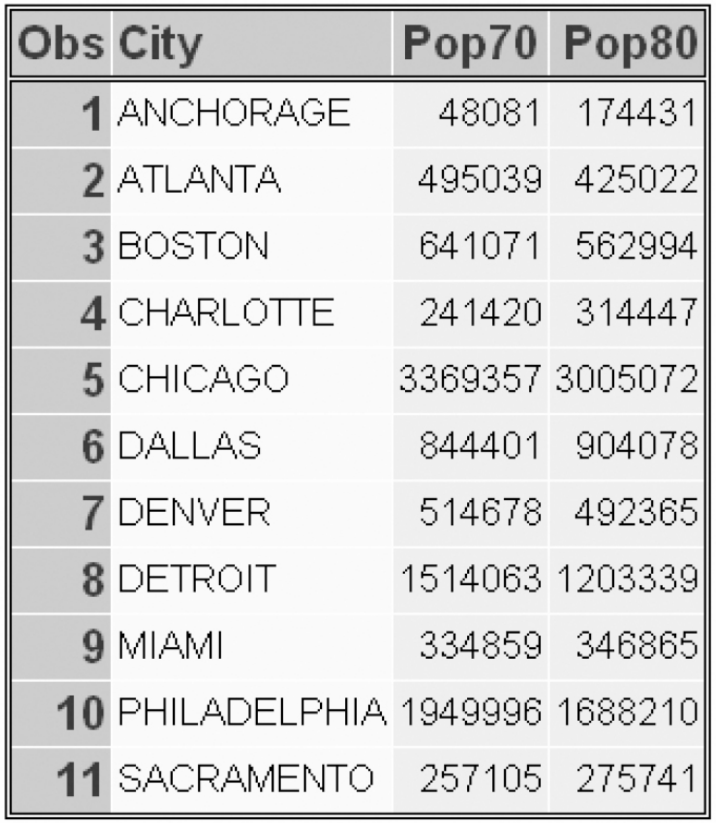
Specifying the Length of Character Variables

Variable attributes are defined when the variable is first encountered in the DATA step. So, the solution is to use the **LENGTH statement** before the INPUT statement to define both the length and type of the variable(s).



```
1---+----10---+----20---+----
ANCHORAGE 48081 174431
ATLANTA 495039 425022
BOSTON 641071 562994
CHARLOTTE 241420 314447
CHICAGO 3369357 3005072
DALLAS 844401 904078
DENVER 514678 492365
DETROIT 1514063 1203339
MIAMI 334859 346865
PHILADELPHIA 1949996 1688210
SACRAMENTO 257105 275741
```

```
data sasuser.growth;
  infile citydata;
  length City $ 12;
  input city $ Pop70 Pop80;
run;
proc print data=sasuser.growth;
run;
```



Obs	City	Pop70	Pop80
1	ANCHORAGE	48081	174431
2	ATLANTA	495039	425022
3	BOSTON	641071	562994
4	CHARLOTTE	241420	314447
5	CHICAGO	3369357	3005072
6	DALLAS	844401	904078
7	DENVER	514678	492365
8	DETROIT	1514063	1203339
9	MIAMI	334859	346865
10	PHILADELPHIA	1949996	1688210
11	SACRAMENTO	257105	275741

Note: A variable defined in a LENGTH statement before an INPUT statement appears first in the data set, regardless of the order of the variables in the INPUT statement.

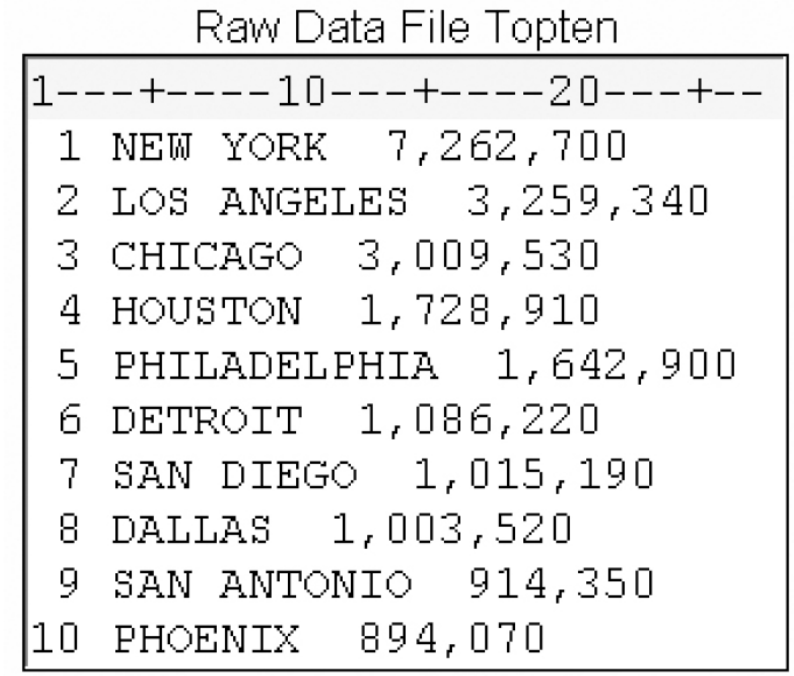
Modified List Input

List input can be more versatile by using **modifiers**:

- The **ampersand (&) modifier:** to read character values that contain a **single embedded blank**, but you must use at least two consecutive blanks as delimiters to separate the values. The & indicates that a character value that is read with list input might contain one or more single embedded blanks. The value is read until two or more consecutive blanks are encountered. *The & modifier **precedes** a specified informat if one is used.*
- The **colon (:) modifier:** to read nonstandard data values and character values that are longer than 8 characters, but which contain no embedded blanks.

A dataset example:

Some city names contain embedded single blanks and all are followed by two blanks to separate the numeric values; the numeric values are nonstandard numeric values (they contain commas).

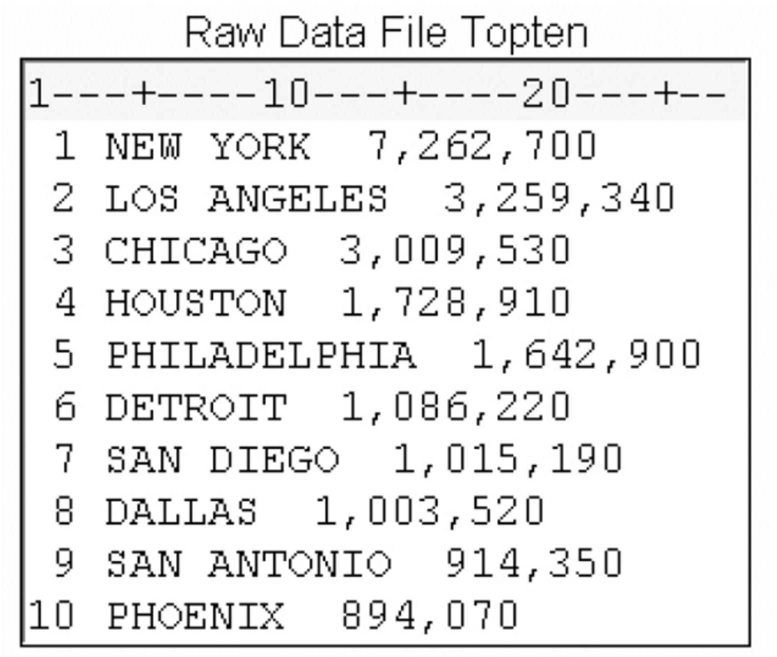

Raw Data File Topten

```
1---+----10---+----20---+--
 1 NEW YORK  7,262,700
 2 LOS ANGELES  3,259,340
 3 CHICAGO  3,009,530
 4 HOUSTON  1,728,910
 5 PHILADELPHIA  1,642,900
 6 DETROIT  1,086,220
 7 SAN DIEGO  1,015,190
 8 DALLAS  1,003,520
 9 SAN ANTONIO  914,350
10 PHOENIX  894,070
```

Modified List Input

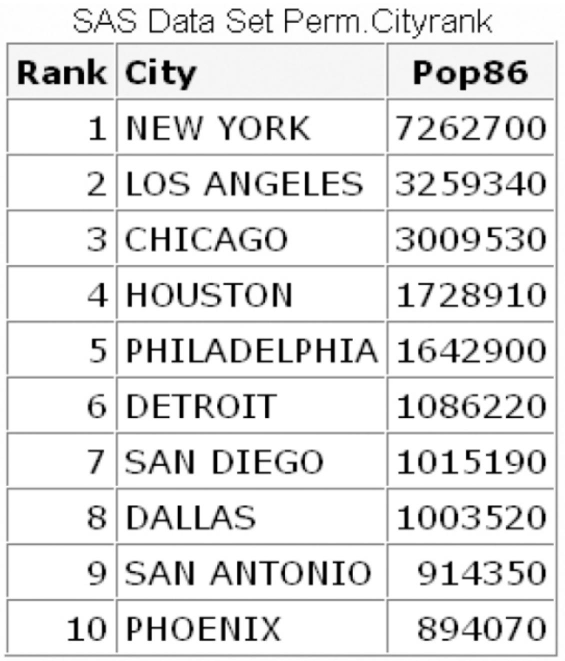
Often you use the modifiers with an informat. Note that in modified list input the informats are used differently from those in the formatted input style, e.g., if *COMMAw.d* is used with :, *w* and *d* values are not specified.

```
data Perm.cityrank;
  infile top10;
  input Rank City & $12. Pop86 : comma.;
run;
```



Raw Data File Topten

```
1---+----10---+----20---+--
 1 NEW YORK  7,262,700
 2 LOS ANGELES  3,259,340
 3 CHICAGO  3,009,530
 4 HOUSTON  1,728,910
 5 PHILADELPHIA  1,642,900
 6 DETROIT  1,086,220
 7 SAN DIEGO  1,015,190
 8 DALLAS  1,003,520
 9 SAN ANTONIO  914,350
10 PHOENIX  894,070
```



SAS Data Set Perm.Cityrank

Rank	City	Pop86
1	NEW YORK	7262700
2	LOS ANGELES	3259340
3	CHICAGO	3009530
4	HOUSTON	1728910
5	PHILADELPHIA	1642900
6	DETROIT	1086220
7	SAN DIEGO	1015190
8	DALLAS	1003520
9	SAN ANTONIO	914350
10	PHOENIX	894070

Modified List Input

You can also use the & Modifier together with a LENGTH Statement to determine the length of a variable.

```
data Perm.cityrank;
  infile top10;
  length City $ 12;
  input Rank City & Pop86 : comma.;
run;
```

City	Rank	Pop86
NEW YORK	1	7262700
LOS ANGELES	2	3259340
CHICAGO	3	3009530
HOUSTON	4	1728810
PHILADELPHIA	5	1642900
DETROIT	6	1086220
SAN DIEGO	7	1015190
DALLAS	8	1003520
SAN ANTONIO	9	914350
PHOENIX	10	890070

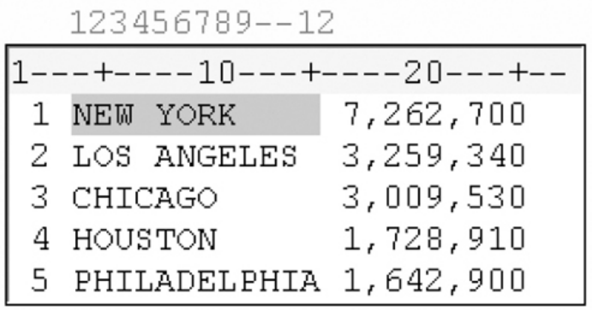
```
data Perm.cityrank;
  infile top10;
  length Rank City $ 12;
  input Rank City & Pop86 : comma.;
run;
```

Rank	City	Pop86
1	NEW YORK	7262700
2	LOS ANGELES	3259340
3	CHICAGO	3009530
4	HOUSTON	1728810
5	PHILADELPHIA	1642900
6	DETROIT	1086220
7	SAN DIEGO	1015190
8	DALLAS	1003520
9	SAN ANTONIO	914350
10	PHOENIX	890070

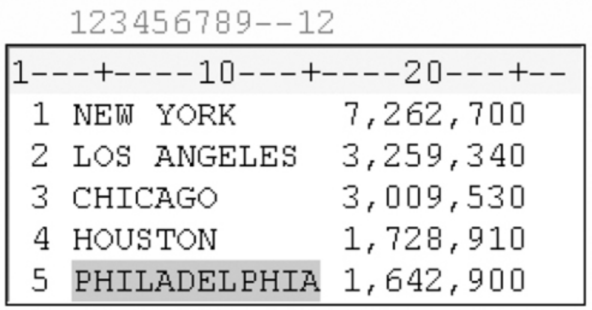
Comparing Formatted Input and Modified List Input

- Informats work differently in modified list input than they do in formatted input. With formatted input, the informat determines both the length of character variables and the number of columns that are read. The same number of columns are read from each record.

input **@4** City **$12.**;



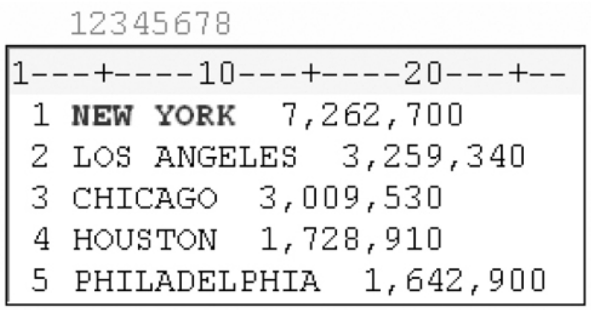
```
   123456789--12
1---+----10---+----20---+--
 1 NEW YORK    7,262,700
 2 LOS ANGELES 3,259,340
 3 CHICAGO     3,009,530
 4 HOUSTON     1,728,910
 5 PHILADELPHIA 1,642,900
```



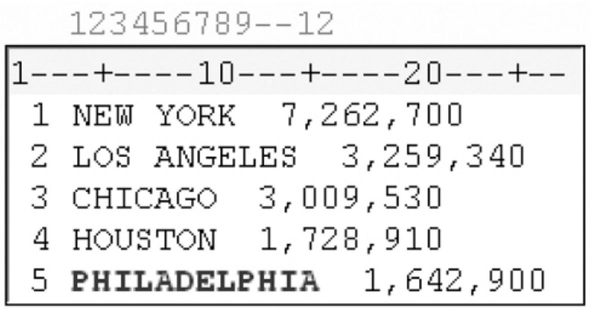
```
   123456789--12
1---+----10---+----20---+--
 1 NEW YORK     7,262,700
 2 LOS ANGELES  3,259,340
 3 CHICAGO      3,009,530
 4 HOUSTON      1,728,910
 5 PHILADELPHIA 1,642,900
```

- The informat in modified list input determines only the length of the variable, not the number of columns that are read. Here, the raw data values are read until two consecutive blanks are encountered.

Input Rank City **&** **$12.**;



```
   12345678
1---+----10---+----20---+--
 1 NEW YORK  7,262,700
 2 LOS ANGELES  3,259,340
 3 CHICAGO  3,009,530
 4 HOUSTON  1,728,910
 5 PHILADELPHIA  1,642,900
```



```
   123456789--12
1---+----10---+----20---+--
 1 NEW YORK  7,262,700
 2 LOS ANGELES  3,259,340
 3 CHICAGO  3,009,530
 4 HOUSTON  1,728,910
 5 PHILADELPHIA  1,642,900
```

List Output: Creating Free-Format Raw Data

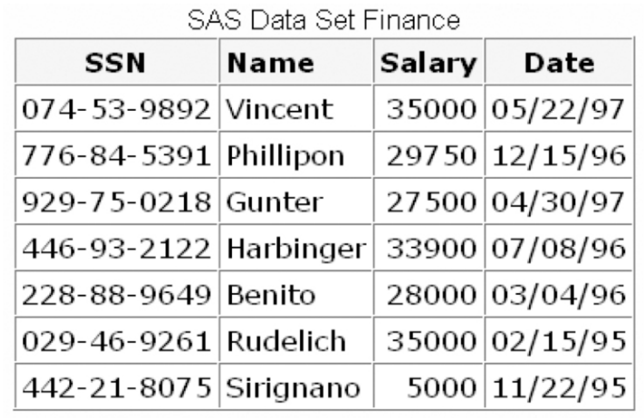
With list output, you simply list the names of the variables whose values you want to write. The PUT statement writes a value, leaves a blank, and then writes the next value.

General form:

PUT variable(s) <: format>;

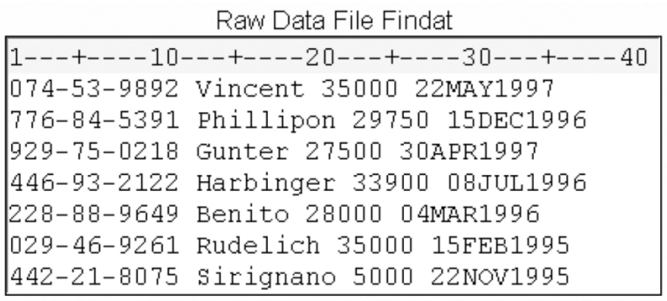
where

variable	specifies the variable(s) whose value(s) you want to write
:	precedes a format
format	specifies a format to use for writing the data values


SAS Data Set Finance

SSN	Name	Salary	Date
074-53-9892	Vincent	35000	05/22/97
776-84-5391	Phillipon	29750	12/15/96
929-75-0218	Gunter	27500	04/30/97
446-93-2122	Harbinger	33900	07/08/96
228-88-9649	Benito	28000	03/04/96
029-46-9261	Rudelich	35000	02/15/95
442-21-8075	Sirignano	5000	11/22/95

```
data _null_;
   set sasuser.finance;
   file 'c:\data\findat.txt';
   put ssn name salary
          date : date9.;
run;
```


Raw Data File Findat

```
1---+----10---+----20---+----30---+----40
074-53-9892 Vincent 35000 22MAY1997
776-84-5391 Phillipon 29750 15DEC1996
929-75-0218 Gunter 27500 30APR1997
446-93-2122 Harbinger 33900 08JUL1996
228-88-9649 Benito 28000 04MAR1996
029-46-9261 Rudelich 35000 15FEB1995
442-21-8075 Sirignano 5000 22NOV1995
```

List Output: Creating Free-Format Raw Data

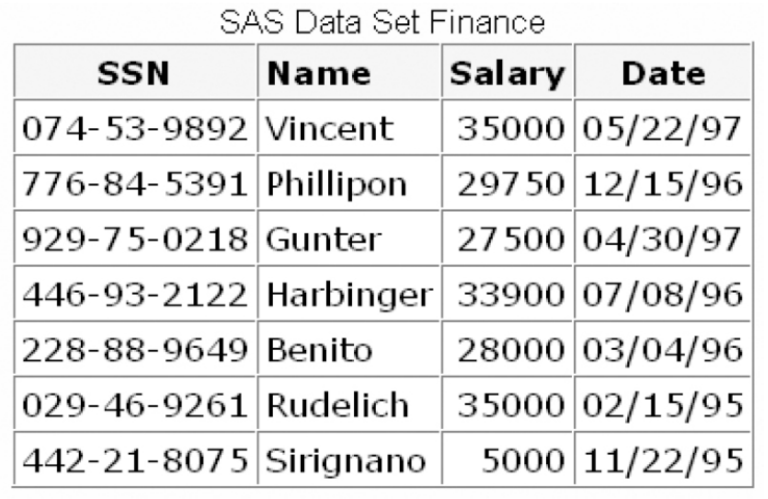
Specifying a Delimiter

You can use the DLM= option with a FILE statement to create a character-delimited raw data file.

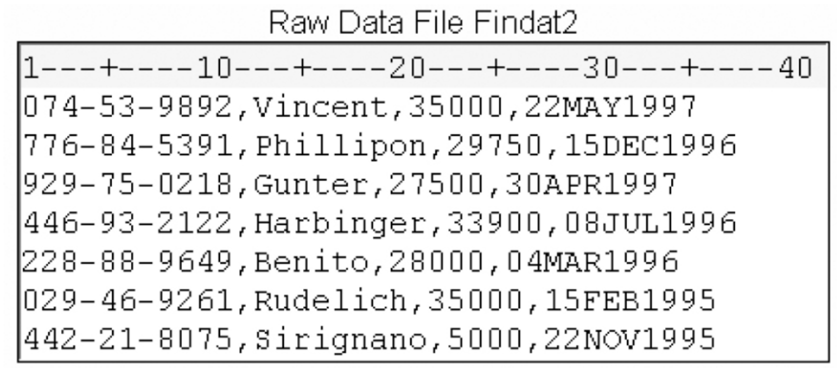
```
data _null_;
  set sasuser.finance;
  file 'c:\data\findat2.txt' dlm=',';
  put ssn name salary date : date9.;
run;
```

Alternatively:

```
proc export data=sasuser.finance;
  outfile ='c:\data\findat2.txt' delimiter=',';
run;
```


SAS Data Set Finance

SSN	Name	Salary	Date
074-53-9892	Vincent	35000	05/22/97
776-84-5391	Phillipon	29750	12/15/96
929-75-0218	Gunter	27500	04/30/97
446-93-2122	Harbinger	33900	07/08/96
228-88-9649	Benito	28000	03/04/96
029-46-9261	Rudelich	35000	02/15/95
442-21-8075	Sirignano	5000	11/22/95


Raw Data File Findat2

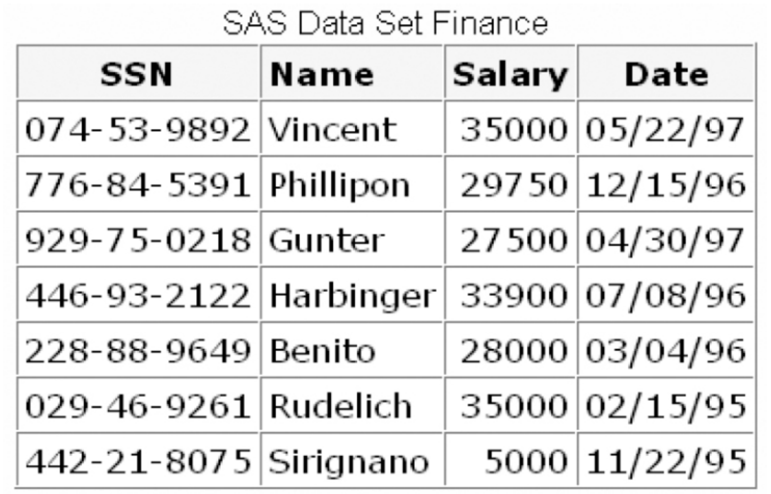
```
1---+----10---+----20---+----30---+----40
074-53-9892,Vincent,35000,22MAY1997
776-84-5391,Phillipon,29750,15DEC1996
929-75-0218,Gunter,27500,30APR1997
446-93-2122,Harbinger,33900,08JUL1996
228-88-9649,Benito,28000,04MAR1996
029-46-9261,Rudelich,35000,15FEB1995
442-21-8075,Sirignano,5000,22NOV1995
```

List Output: Creating Free-Format Raw Data

Using the DSD Option

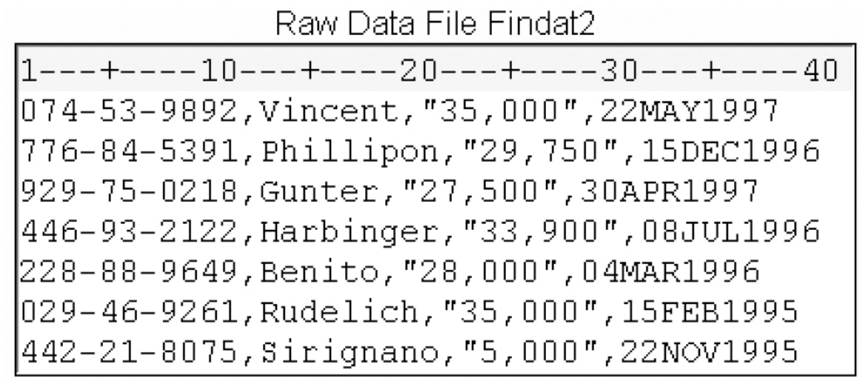
You can use the **DSD** option in the FILE statement to specify that data values containing commas should be enclosed in quotation marks. Since the DSD option uses a comma as a delimiter, so a DLM= option isn't necessary here.

```
data _null_;
   set sasuser.finance;
   file 'c:\data\findat2.txt' dsd;
   put ssn name salary : comma. date : date9.;
run;
```



SAS Data Set Finance

SSN	Name	Salary	Date
074-53-9892	Vincent	35000	05/22/97
776-84-5391	Phillipon	29750	12/15/96
929-75-0218	Gunter	27500	04/30/97
446-93-2122	Harbinger	33900	07/08/96
228-88-9649	Benito	28000	03/04/96
029-46-9261	Rudelich	35000	02/15/95
442-21-8075	Sirignano	5000	11/22/95



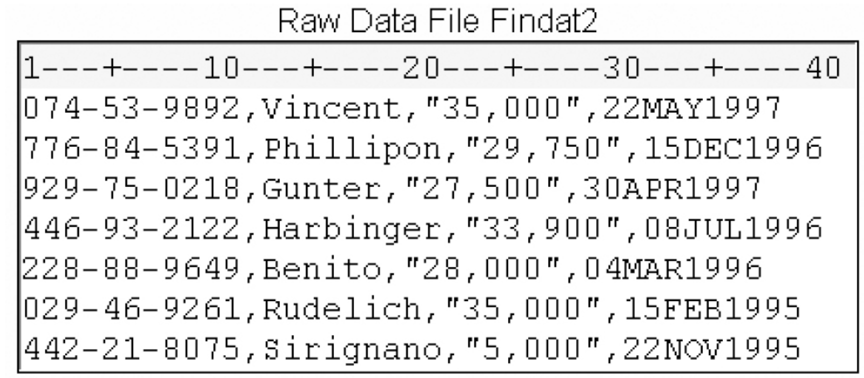
Raw Data File Findat2

```
1---+----10---+----20---+----30---+----40
074-53-9892,Vincent,"35,000",22MAY1997
776-84-5391,Phillipon,"29,750",15DEC1996
929-75-0218,Gunter,"27,500",30APR1997
446-93-2122,Harbinger,"33,900",08JUL1996
228-88-9649,Benito,"28,000",04MAR1996
029-46-9261,Rudelich,"35,000",15FEB1995
442-21-8075,Sirignano,"5,000",22NOV1995
```

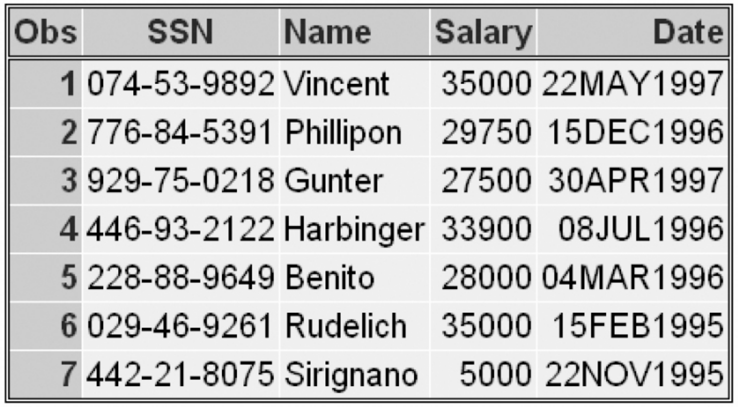
Reading Values That Contain Delimiters Within a Quoted String

You can also use the **DSD** option in an INFILE statement to read values that contain delimiters within a quoted string. The INPUT statement interprets the commas within the values not as delimiters, and removes the quotation marks from the character strings before the value is stored.

```
data work.finance2;
  infile findat2 dsd;
  length SSN $ 11 Name $ 9;
  input SSN Name Salary : comma. Date : date9.;
run;
proc print data=work.finance2;
  format date date9.;
run;
```


Raw Data File Findat2

```
1---+----10---+----20---+----30---+----40
074-53-9892,Vincent,"35,000",22MAY1997
776-84-5391,Phillipon,"29,750",15DEC1996
929-75-0218,Gunter,"27,500",30APR1997
446-93-2122,Harbinger,"33,900",08JUL1996
228-88-9649,Benito,"28,000",04MAR1996
029-46-9261,Rudelich,"35,000",15FEB1995
442-21-8075,Sirignano,"5,000",22NOV1995
```

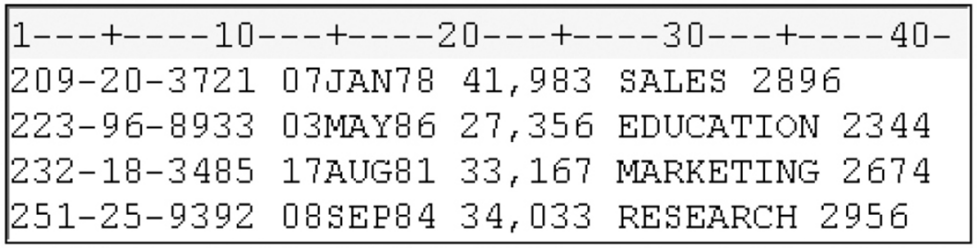


Obs	SSN	Name	Salary	Date
1	074-53-9892	Vincent	35000	22MAY1997
2	776-84-5391	Phillipon	29750	15DEC1996
3	929-75-0218	Gunter	27500	30APR1997
4	446-93-2122	Harbinger	33900	08JUL1996
5	228-88-9649	Benito	28000	04MAR1996
6	029-46-9261	Rudelich	35000	15FEB1995
7	442-21-8075	Sirignano	5000	22NOV1995

Mixing Input Styles

Input Style	Reads
Column	standard data values in fixed fields
Formatted	standard and nonstandard data values in fixed fields
(Modified) List	data values that are not arranged in fixed fields, but are separated by blanks or other delimiters

With some file layouts (e.g., the one shown below), you might need to mix input styles in the same INPUT statement in order to read the data correctly.



```
1---+----10---+----20---+----30---+----40-
209-20-3721 07JAN78 41,983 SALES 2896
223-96-8933 03MAY86 27,356 EDUCATION 2344
232-18-3485 17AUG81 33,167 MARKETING 2674
251-25-9392 08SEP84 34,033 RESEARCH 2956
```

```
data sasuser.mixedstyles;
  infile rawdata;
  input SSN $ 1-11 @13 HireDate date7.
      @21 Salary comma6. Department : $9.
       Phone $;
run;
proc print data=sasuser.mixedstyles;
run;
```

Field Description	Starting column	Field Width	Data Type	Input Style
SSN	1	11	character	column
HireDate	13	7	date	formatted
Salary	21	6	numeric	formatted
Department	28	5 to 9	character	list (modified)
Phone	?	4	character	list

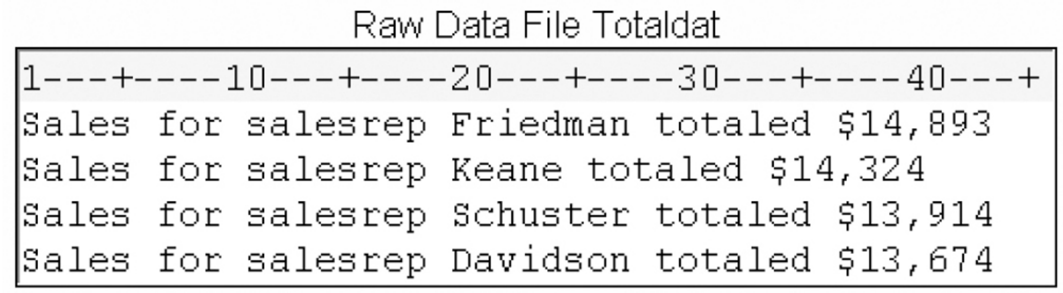
Obs	SSN	HireDate	Salary	Department	Phone
1	209-20-3721	6581	41983	Sales	2897
2	223-96-8933	9619	27356	Education	2344
3	232-18-3485	7899	33167	MARKETING	2674
4	251-25-9392	9017	34033	RESEARCH	2956

Writing Character Strings and Variable Values

You can use a **PUT** statement to write both character strings and variable values to a raw data file. To write out a character string, simply add a character string, enclosed in quotation marks, to the PUT statement. It's a good idea to include a blank space as the last character in the string to avoid spacing problems.

```
filename totaldat 'c:\records\junsales.txt';
data _null_;
  set work.totals;
  file totaldat;
  put 'Sales for salesrep ' SalesRep
     'totaled ' Sales : dollar9.;
run;
```

Obs	SalesRep	Sales
1	Friedman	14893
2	Keane	14324
3	Schuster	13914
4	Davidson	13674


Raw Data File Totaldat

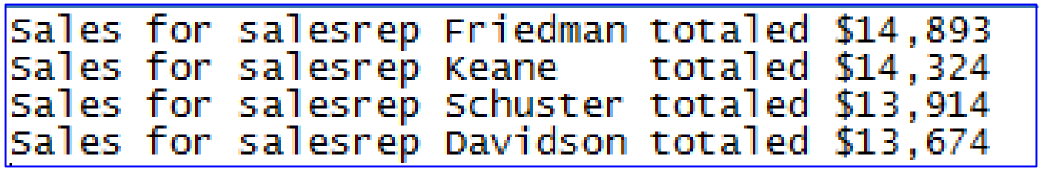
```
1---+----10---+----20---+----30---+----40---+
Sales for salesrep Friedman totaled $14,893
Sales for salesrep Keane totaled $14,324
Sales for salesrep Schuster totaled $13,914
Sales for salesrep Davidson totaled $13,674
```

Writing Character Strings and Variable Values

You can use a format to specify the length of the fields you want to write the raw data file.

```
filename totaldat 'c:\records\junsales.txt';
data _null_;
  set work.totals;
  file totaldat;
  put 'Sales for salesrep ' SalesRep $9.
     'totaled ' Sales : dollar9.;
run;
```

Obs	SalesRep	Sales
1	Friedman	14893
2	Keane	14324
3	Schuster	13914
4	Davidson	13674



```
Sales for salesrep Friedman totaled $14,893
Sales for salesrep Keane    totaled $14,324
Sales for salesrep Schuster totaled $13,914
Sales for salesrep Davidson totaled $13,674
```