SAS Certification Prep Guide: Base Programming for SAS 9, Third Edition

by SAS Institute

Chapter 21: Reading Hierarchical Files

Overview

Introduction

Raw data files can be hierarchical in structure, consisting of a header record and one or more detail records. Typically, each record contains a field that identifies the record type.

Here, the P indicates a header record that contains a patient's ID number. The C indicates a detail record that contains the date of the patient's appointment and the charges that the patient has incurred.

Raw Data File

```
1---+----10---+----20
P   1095
C   01-08-89   $45.0
C   01-17-89   $37.5
P   1096
C   01-09-89   $156.5
P   1097
C   01-02-89   $109.0
P   1099
C   01-03-89   $45.0
C   01-05-89   $45.0
P   1201
C   01-05-89   $37.0
C   01-10-89   $45.0
```

You can build a SAS data set from a hierarchical file by creating one observation per detail record, retaining the patient ID from the header record.

SAS Data Set

Obs	ID	Date	Amount
1	1095	01/08/89	$45.00
2	1095	01/17/89	$37.50
3	1096	01/09/89	$156.50
4	1097	01/02/89	$109.00
5	1099	01/03/89	$45.00
6	1099	01/05/89	$45.00
7	1201	01/05/89	$37.00
8	1201	01/10/89	$45.00

You can also build a SAS data set from a hierarchical file by creating one observation per header record and combining the information from detail records into summary variables.

SAS Data Set		
Obs	ID	Total
1	1095	$82.50
2	1096	$156.50
3	1097	$109.00
4	1099	$90.00
5	1201	$82.00

In this chapter, you learn how to read from a hierarchical file and create a SAS data set that contains either one observation per detail record or one observation per header record.

Objectives

In this chapter, you learn to

- retain the value of a variable
- conditionally execute a SAS statement
- determine when the last observation is being processed
- conditionally execute multiple SAS statements.

You can also review how to

- use a line-hold specifier to hold the current record
- explicitly write an observation to a data set.

Creating One Observation per Detail Record

Overview

In order to create one observation per detail record, it is necessary to distinguish between header and detail records. Having a field that identifies the type of the record makes this task easier.

In the partial raw data file Census, shown below, H indicates a header record that contains a street address, and P indicates a detail record that contains information about a person who lives at that address.

Raw Data File

```
1---+----10---+----
H 321 S. MAIN ST
P MARY E     21 F
P WILLIAM M  23 M
P SUSAN K     3 F
H 324 S. MAIN ST
P THOMAS H   79 M
P WALTER S   46 M
P ALICE A    42 F
P MARYANN A  20 F
P JOHN S     16 M
H 325A S. MAIN ST
```

Let's see how you can create a data set that contains one observation for each person who lives at a specific address.

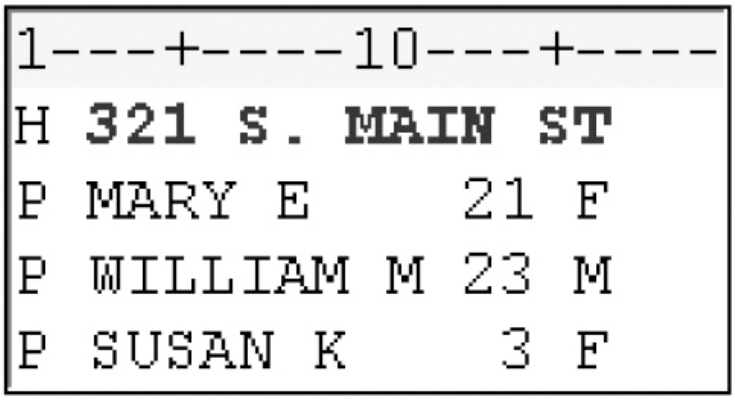
SAS Data Set

Obs	Address	Name	Age	Gender
1	321 S. MAIN ST	MARY E	21	F
2	321 S. MAIN ST	WILLIAM M	23	M
3	321 S. MAIN ST	SUSAN K	3	F
4	324 S. MAIN ST	THOMAS H	79	M
5	324 S. MAIN ST	WALTER S	46	M
6	324 S. MAIN ST	ALICE A	42	F
7	324 S. MAIN ST	MARYANN A	20	F
8	324 S. MAIN ST	JOHN S	16	M
9	325A S. MAIN ST	JAMES L	34	M
10	325A S. MAIN ST	LIZA A	31	F
11	325B S. MAIN ST	MARGO K	27	F

Retaining the Values of Variables

As you write the DATA step to read this file, remember that you want to keep the header record as a part of each observation until the next header record is encountered. To do this, you need to use a RETAIN statement to retain the values for Address across iterations of the DATA step.

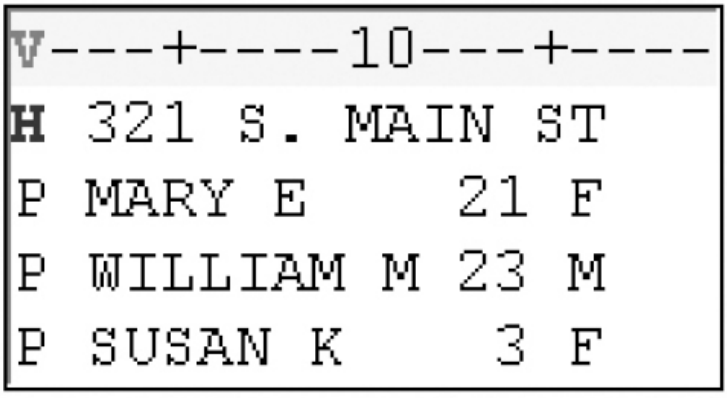
```
data perm.people;
   infile census;
   retain Address;
```



Additional Note Retain the variable Address so that its value will be available to subsequent iterations of the DATA step.

Next, read the first field in each record, which identifies the record's type. You also need to use the @ line-hold specifier to hold the current record so that the other values in the record can be read later.

```
data perm.people;
   infile census;
   retain Address;
   input type $1. @;
```

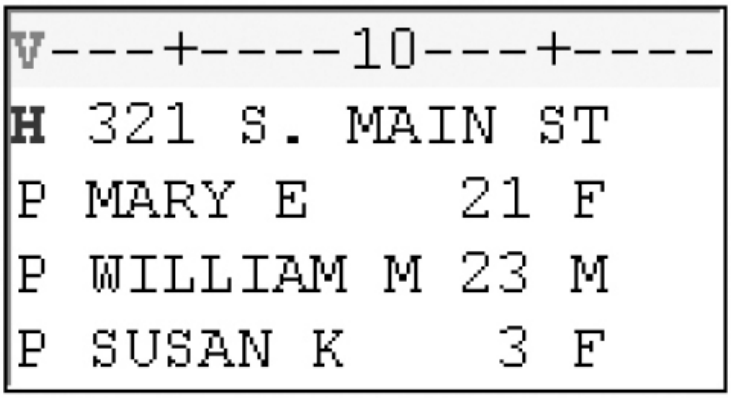


Conditionally Executing SAS Statements

You can use the value of type to identify each record. If type is **H**, you need to execute an INPUT statement to read the values for Address. However, if type is **P**, then execute an INPUT statement to read the values for Name, Age, and Gender.

You can tell SAS to perform a given task based on a specific condition by using an IF-THEN statement.

```
data perm.people;
   infile census;
   retain Address;
   input type $1. @;
   if type='H' then
      input @3 address $15.;
```



Expressions in conditional statements usually involve some kind of comparison. In the example shown above, a variable is compared to a constant. When the condition is met, the expression is evaluated as true, and the statement that follows the keyword THEN is executed.

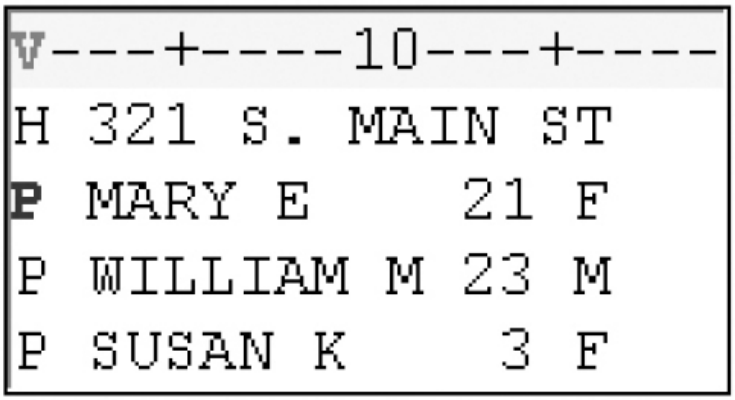
The expression defines a condition so that when the value of type is **H**, the INPUT statement reads the values for Address. However, when the value of type is not **H**, the expression is evaluated as false, and the INPUT statement is not executed. Notice that the value is enclosed in quotation marks because it is a character value.

Additional Note When you compare values, be sure to express the values exactly as they are coded in the data. For example, the expression below would evaluate to false because the values in the data are stored in uppercase letters.

```
if type='h' then … ;
```

Reading a Detail Record

Now think about what needs to happen when a detail record is read. Remember, you want to write an observation to the data set only when the value of type is **P**.



You can use a subsetting IF statement to check for the condition that type is **P**. The remaining DATA step statements execute only when the condition is true. If type is not **P**, then the values for Name, Age, and Gender are not read, the values in the program data vector are not written to the data set as an observation, and control returns to the top of the DATA step. However, Address is retained.

If type is **P**, Name, Age, and Gender are read, and an observation is written to the data set. Remember that you want to create an observation for detail records only.

```
data perm.people;
   infile census;
   retain Address;
   input type $1. @;
   if type='H' then input @3 address $15.;
   if type='P';
```

```
   input @3 Name $10. @13 Age 3. @16 Gender $1.;
run;
```

Dropping Variables

Because type is useful only for identifying a record's type, drop the variable from the data set. The DROP= option in the DATA statement shown here prevents the variable type from being written to the data set.

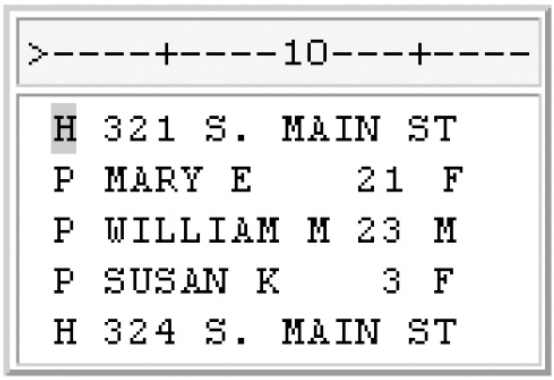
```
data perm.people (drop=type);
   infile census;
   retain Address;
   input type $1. @;
   if type='H' then input @3 address $15.;
   if type='P';
   input @3 Name $10. @13 Age 3. @16 Gender $1.;
run;
```

Processing a DATA Step That Creates One Observation per Detail Record

Let's see how this DATA step is processed.

```
data perm.people (drop=type);
   infile census;
   retain Address;
   input type $1. @;
   if type='H' then input @3 address $15.;
   if type='P';
   input @3 Name $10. @13 Age 3. @16 Gender $1.;
run;
```

At compile time, the variable type is flagged so that its values are not written to the data set. Address is flagged so that its value is retained across iterations of the DATA step.



Retain Drop

_N_	Address	type	Name	Age	Gender
•				•	

As the DATA step begins to execute, the INPUT statement reads the value for type and holds the first record.

```
>----+----10---+----
H 321 S. MAIN ST
P MARY E    21  F
P WILLIAM M 23  M
P SUSAN K    3  F
H 324 S. MAIN ST
```

Retain Drop

_N_	Address	type	Name	Age	Gender
1		H		.	

The condition type='H' is checked and found to be true, so the INPUT statement reads the value for Address in the first record.

```
>----+----10---+----
H 321 S. MAIN ST
P MARY E    21  F
P WILLIAM M 23  M
P SUSAN K    3  F
H 324 S. MAIN ST
```

Retain Drop

_N_	Address	type	Name	Age	Gender
1	321 S. MAIN ST	H		.	

Next, the subsetting IF statement checks for the condition type='P'. Because the condition is not true, the remaining statements are not executed and control returns to the top of the DATA step. The PDV is initialized but Address is retained.

```
>----+----10---+----
 H 321 S. MAIN ST
 P MARY E    21  F
 P WILLIAM M 23  M
 P SUSAN K    3  F
 H 324 S. MAIN ST
```

Retain Drop

_N_	Address	type	Name	Age	Gender
2	321 S. MAIN ST			.	

As the second iteration begins, the input pointer moves to the next record and a new value for type is read. The condition expressed in the IF-THEN statement is not true, so the statement following the THEN keyword is not executed.

```
>----+----10---+----
 H 321 S. MAIN ST
 P MARY E    21  F
 P WILLIAM M 23  M
 P SUSAN K    3  F
 H 324 S. MAIN ST
```

Retain Drop

_N_	Address	type	Name	Age	Gender
2	321 S. MAIN ST	P		.	

Now the subsetting IF statement checks for the condition type-'P'. In this iteration, the condition is true, so the final INPUT statement reads the values for Name, Age, and Gender.

```
>----+----10---+----
 H 321 S. MAIN ST
 P MARY E    21 F
 P WILLIAM M 23 M
 P SUSAN K    3 F
 H 324 S. MAIN ST
```

Retain Drop

_N_	Address	type	Name	Age	Gender
2	321 S. MAIN ST	P	MARY E	21	F

Then the values in the program data vector are written as the first observation, and control returns to the top of the DATA step. Notice that the values for type are not included.

```
>----+----10---+----
 H 321 S. MAIN ST
 P MARY E    21 F
 P WILLIAM M 23 M
 P SUSAN K    3 F
 H 324 S. MAIN ST
```

Retain Drop

_N_	Address	type	Name	Age	Gender
2	321 S. MAIN ST	P	MARY E	21	F

SAS Data Set Perm.People

Obs	Address	Name	Age	Gender
1	321 S. MAIN ST	MARY E	21	F

As execution continues, observations are produced from the third and fourth records. However, notice that the fifth record is a header record. During the fifth iteration, the condition type='H' is true, so a new Address is read into the program data vector, overlaying the previous value.

```
>----+----10---+----
 H 321 S. MAIN ST
 P MARY E     21  F
 P WILLIAM M 23  M
 P SUSAN K     3  F
 H 324 S. MAIN ST
```

Retain Drop

_N_	Address	type	Name	Age	Gender
5	324 S. MAIN ST	H		.	

SAS Data Set Perm.People

Obs	Address	Name	Age	Gender
1	321 S. MAIN ST	MARY E	21	F
2	321 S. MAIN ST	WILLIAM M	23	M
3	321 S. MAIN ST	SUSAN K	3	F

Displaying Your Results

When the execution phase is complete, you can display the data set by using the PRINT procedure. The first 10 observations are displayed.

Obs	Address	Name	Age	Gender
1	321 S. MAIN ST	MARY E	21	F
2	321 S. MAIN ST	WILLIAM M	23	M
3	321 S. MAIN ST	SUSAN K	3	F
4	324 S. MAIN ST	THOMAS H	79	M
5	324 S. MAIN ST	WALTER S	46	M
6	324 S. MAIN ST	ALICE A	42	F
7	324 S. MAIN ST	MARYANN A	20	F
8	324 S. MAIN ST	JOHN S	16	M
9	325AS. MAIN ST	JAMES L	34	M
10	325AS. MAIN ST	LIZA A	31	F

Figure 21.1: Output from the PRINT Procedure

Creating One Observation per Header Record

Overview

In the previous example, you learned how to create one observation per detail record. But suppose you only want to know how many people reside at each address. You can create a data set that reads each detail record, counts the number of people, and stores this number in a summary variable.

In the example below, this summary variable, Total, is computed for each address. As you can see, creating one observation per header record condenses a large amount of information into a concise data set.

Raw Data File

```
1---+----10---+----20
H 321 S. MAIN ST
P MARY E    21 F
P WILLIAM M 23 M
P SUSAN K    3 F
H 324 S. MAIN ST
P THOMAS H  79 M
P WALTER S  46 M
P ALICE A   42 F
P MARYANN A 20 F
P JOHN S    16 M
H 325A S. MAIN ST
P JAMES L 34 M
P LIZA A 31 F
H 325B S. MAIN ST
P MARGO K 27 F
P WILLIAM R 27 M
P ROBERT W 1 M
```

SAS Date Set

Address	total
321 S MAIN ST	3
324 S MAIN ST	5
325A S MAIN ST	2
325B S MAIN ST	3

As you write the DATA step to read this file, you need to think about performing several tasks. First, the value of Address must be retained as detail records are read and summarized.

```
data perm.people;
   infile census;
   retain Address;
```

```
1---+----10---+----
H 321 S. MAIN ST
P MARY E    21 F
P WILLIAM M 23 M
P SUSAN K    3 F
```

Next, the value of type must be read in order to determine whether the current record is a header record or a detail record. Add an @ to hold the record so that another INPUT statement can read the remaining values.

```
data perm.people;
   infile census;
```

```
retain Address;
input type $1. @;
```

```
V---+----10---+----
H 321 S. MAIN ST
P MARY E    21 F
P WILLIAM M 23 M
P SUSAN K    3 F
```

When the value of type indicates a header record, several statements need to be executed. When the value of type indicates a detail record, you need to define an alternative set of actions. Let's look at executing different sets of statements for each value of type.

DO Group Actions for Header Records

To execute multiple SAS statements based on the value of a variable, you can use a simple DO group with an IF-THEN statement. When the condition type='H' is true, several statements need to be executed.

```
data perm.residnts;
   infile census;
   retain Address;
   input type $1. @;
   if type='H' then do;
```

- First, you need to determine whether this is the first header record in the external file. You do not want the first header record to be written as an observation until the related detail records have been read and summarized.

  _N_ is an automatic variable whose value is the number of times the DATA step has begun to execute. The expression _n_ > 1 defines a condition where the DATA step has executed more than once. Use this expression in conjunction with the previous IF-THEN statement to check for these two conditions:

  1. The current record is a header record.
  2. The DATA step has executed more than once.

  ```
  data perm.residnts;
     infile census;
     retain Address;
     input type $1. @;
     if type='H' then do;
        if _n_ > 1
  ```

- When the conditions type='H' and _n_ > 1 are true, an OUTPUT statement is executed. Thus, each header record except for the first one causes an observation to be written to the data set.

  ```
  data perm.residnts;
     infile census;
     retain Address;
     input type $1. @;
     if type='H' then do;
        if _n_ > 1 then output;
  ```

- An assignment statement creates the summary variable Total and sets its value to **0**.

  ```
  data perm.residnts;
     infile census;
  ```

```
      retain Address;
      input type $1. @;
      if type='H' then do;
         if _n_ > 1 then output;
         Total=0;
```

- An INPUT statement reads the values for Address.

```
data perm.residnts;
   infile census;
   retain Address;
   input type $1. @;
   if type='H' then do;
      if _n_ > 1 then output;
      Total=0;
      input address $ 3-17;
```

- An END statement closes the DO group.

```
data perm.residnts;
  infile census;
  retain Address;
  input type $1. @;
  if type='H' then do;
     if _n_ > 1 then output;
     Total=0;
     input address $ 3-17;
end;
```

Reading Detail Records

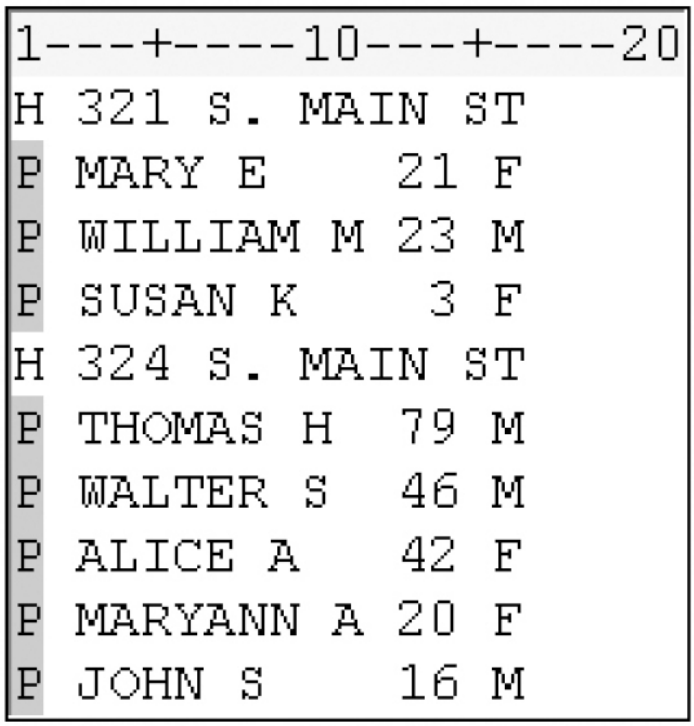
When the value of type is not **H**, you need to define an alternative action. You can do this by adding an ELSE statement after the DO group.

Remember that the IF-THEN statement executes a SAS statement when the condition specified in the IF clause is true. By adding an ELSE statement after the IF-THEN statement, you define an alternative action to be performed when the IF condition is false.

```
data perm.residnts;
   infile census;
   retain Address;
   input type $1. @;
   if type='H' then do;
      if _n_ > 1 then output;
      Total=0;
      input address $ 3-17;
   end;
else
```

The only other type of record is a detail record, represented by a P. You want to count each person who is represented by a detail record and store the accumulated value in the summary variable Total. You do not need to read the values for Name, Age, and Gender.

```
data perm.residnts;
   infile census;
   retain Address;
   input type $1. @;
   if type='H' then do;
      if _n_ > 1 then output;
      Total=0;
      input address $ 3-17;
   end;
   else if type='P' then
```



```
1---+----10---+----20
H 321 S. MAIN ST
P MARY E     21 F
P WILLIAM M  23 M
P SUSAN K     3 F
H 324 S. MAIN ST
P THOMAS H   79 M
P WALTER S   46 M
P ALICE A    42 F
P MARYANN A  20 F
P JOHN S     16 M
```

At this point, the value of Total to **0** has already been initialized each time a header record is read. Now, as each detail record is read, you can increment the value of Total by 1 when the value of type is **P**.

```
data perm.residnts;
   infile census;
   retain Address;
   input type $1. @;
   if type='H' then do;
      if _n_ > 1 then output;
      Total=0;
      input address $ 3-17;
   end;
   else if type='P' then total+1;
```

Additional Note Remember that a sum statement enables you to add any valid SAS expression to an accumulator variable.

```
else if type='B' then total+cost;
```

Additional Note Remember also that the value generated by a sum statement is automatically retained throughout the DATA step. That is why it is important to set the value of Total to **0** each time a header record is read.

Determining the End of the External File

Your program writes an observation to the data set only when another header record is read and the DATA step has executed more than once. But after the last detail record is read, there are no more header records to cause the last observation to be written to the data set.

```
data perm.residnts;
   infile census;
   retain Address;
   input type $1. @;
   if type='H' then do;
      if _n_ > 1 then
         output;
      Total=0;
      input address $ 3-17;
   end;
   else if type='P'
```

```
      then total+1;
```

```
1---+----10---+----20
H 321 S. MAIN ST
P MARY E    21 F
P WILLIAM M 23 M
P SUSAN K    3 F
H 324 S. MAIN ST
P THOMAS H  79 M
P WALTER S  46 M
P ALICE A   42 F
P MARYANN A 20 F
P JOHN S    16 M
H 325A S. MAIN ST
P JAMES L 34 M
P LIZA A 31 F
H 325B S. MAIN ST
P MARGO K 27 F
P WILLIAM R 27 M
P ROBERT W 1 M
```

You need to determine when the last record in the file is read so that you can then execute another explicit OUTPUT statement. You can determine when the current record is the last record in an external file by specifying the END= option in the INFILE statement. (You learned how to use the END= option with a SET statement in "Reading SAS Data Sets" on page 334.)

General form, INFILE statement with the END= option:

INFILE *file-specification* ***END***=*variable*;

where *variable* is a temporary numeric variable whose value is **`0`** until the last line is read and **`1`** after the last line is read.

Like automatic variables, the END= variable is not written to the data set.

In the following example, the END= variable is defined in the INFILE statement as last. When last has a value other than 0, the OUTPUT statement writes the final observation to the data set.

```
data perm.residnts;
   infile census end=last;
   retain Address;
   input type $1. @;
   if type='H' then do;
    if _n_ > 1 then output;
       Total=0;
       input address $ 3-17;
   end;
   else if type='P' then total+1;
   if last then output;
```

A DROP= option in the DATA statement drops the variable type from the data set, and a RUN statement completes the DATA step.

```
data perm.residnts (drop=type);
   infile census end=last;
```

```
   retain Address;
   input type $1. @;
   if type='H' then do;
      if _n_ > 1 then output;
         Total=0;
      input address $ 3-17;
   end;
   else if type='P' then total+1;
   if last then output;
run;
```

Processing a DATA Step That Creates One Observation per Header Record

During the compile phase, the variable type is flagged so that later it can be dropped. The value for Address and Total (SUM statement) are retained.

```
data perm.people (drop=type);
   infile census end=last;
   retain Address;
   input type $1. @;
   if type = 'H' then do;
      if _n_ > 1 then output;
      Total=0;
      input Address $3-17;
   end;
   else if type=' P ' then total+1;
   if last then output;
run;
```

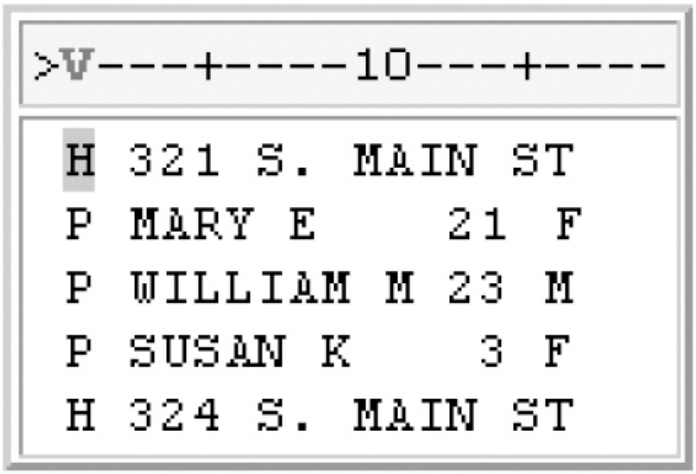
```
>----+----10---+----
 H 321 S. MAIN ST
 P MARY E     21  F
 P WILLIAM M 23  M
 P SUSAN K     3  F
 H 324 S. MAIN ST
```

		Retain		Drop	Retain
_N_	last	Address		type	Total
•	•				•

As the execution begins, _N_ is 1 and last is 0. Total is 0 because of the sum statement.

```
data perm.people (drop=type);
    infile census end=last;
    retain Address;
    input type $1. @;
```

```
    if type = 'H' then do ;
       if _n_ > 1 then output;
       Total=0;
       input Address $3-17;
    end;
    else if type=' P ' then total+1;
    if last then output;
run;
```



Now the value for type is read, the condition type='H' is true, and therefore the statements in the DO group execute.

```
data perm.people (drop=type);
   infile census end=last;
   retain Address;
   input type $1. @;
   if type = 'H' then do ;
       if _n_ > 1 then output;
       Total=0;
       input Address $3-17;
    end;
    else if type=' P ' then total+1;
    if last then output;
run;
```

```
>-V--+----1O---+----
 H 321 S. MAIN ST
 P MARY E     21  F
 P WILLIAM M 23  M
 P SUSAN K     3  F
 H 324 S. MAIN ST
```

	Retain		Drop	Retain
_N_	last	Address	type	Total
1	0		H	0

The condition N>1 is not true, so the OUTPUT statement is not executed. However, Total is assigned the value of 0 and the value for Address is read.

```
data perm.people (drop=type);
   infile census end=last;
   retain Address;
   input type $1. @;
   if type = 'H' then do ;
      if _n_ > 1 then output;
      Total=0;
      input Address $3-17;
   end;
   else if type=' P ' then total+1;
   if last then output;
run;
```

```
>----+----10---+--V-
 H 321 S. MAIN ST
 P MARY E     21  F
 P WILLIAM M 23  M
 P SUSAN K     3  F
 H 324 S. MAIN ST
```

		Retain		Drop	Retain
_N_	last	Address		type	Total
1	0	321 S. MAIN ST		H	0

The END statement closes the DO group. The alternative condition expressed in the ELSE statement is not checked because the first condition, type='H', was true.

```
data perm.people (drop=type);
   infile census end=last;
   retain Address;
   input type $1. @;
   if type = 'H' then do ;
      if _n_ > 1 then output;
      Total=0;
      input Address $3-17;
   end;
   else if type=' P ' then total+1;
   if last then output;
run;
```

```
>----+----10---+----
 H 321 S. MAIN ST
 P MARY E     21  F
 P WILLIAM M 23  M
 P SUSAN K     3  F
 H 324 S. MAIN ST
```

		Retain	Drop	Retain
_N_	last	Address	type	Total
1	0	321 S. MAIN ST	H	0

The value of last is still **0**, so the OUTPUT statement is not executed. Control returns to the top of the DATA step.

```
data perm.people (drop=type);
   infile census end=last;
   retain Address;
   input type $1. @;
   if type = 'H' then do ;
      if _n_ > 1 then output;
      Total=0;
      input Address $3-17;
   end;
   else if type=' P ' then total+1;
   if last then output;
run;
```

```
> ---+----10---+-V--
 H 321 S. MAIN ST
 P MARY E     21  F
 P WILLIAM M 23  M
 P SUSAN K     3  F
 H 324 S. MAIN ST
```

		Retain	Drop	Retain
_N_	last	Address	type	Total
2	0	321 S. MAIN ST		0

During the second iteration, the value of type is **'P'** and Total is incremented by 1. Again, the value of last is **0**, so control returns to the top of the DATA step.

```
data perm.people (drop=type);
   infile census end=last;
   retain Address;
```

```
   input type $1. @;
   if type = 'H' then do ;
      if _n_ > 1 then output;
      Total=0;
      input Address $3-17;
   end;
   else if type=' P ' then total+1;
   if last then output;
run;
```

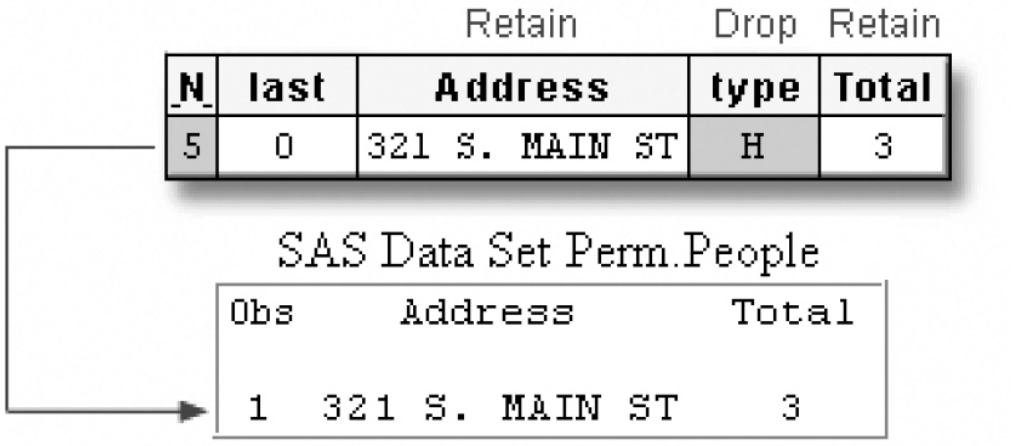
```
>-V--+----10---+----
 H 321 S. MAIN ST
 P MARY E     21  F
 P WILLIAM M  23  M
 P SUSAN K     3  F
 H 324 S. MAIN ST
```

		Retain		Drop	Retain
_N_	last	Address		type	Total
2	0	321 S. MAIN ST		P	1

During the fifth iteration, the value of type is **'H'** and _N_ is greater than 1, so the values for Address and Total are written to the data set as the first observation.

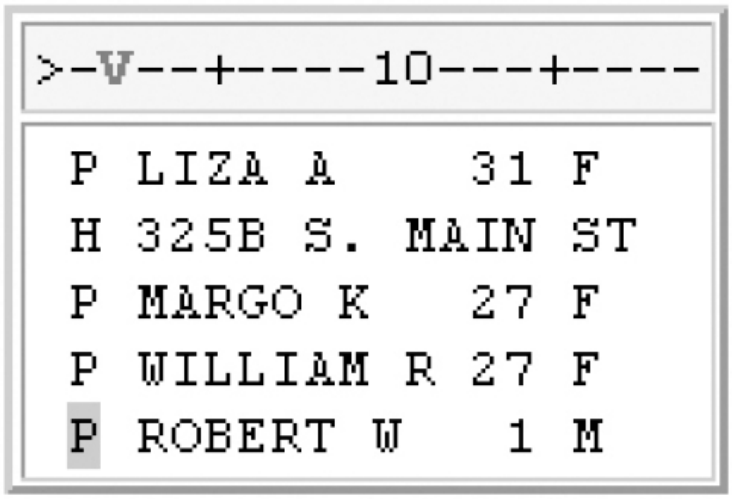
```
data perm.people (drop=type);
   infile census end=last;
   retain Address;
   input type $1. @;
         if type = 'H' then do ;
         if _n_ > 1 then output;
         Total=0;
         input Address $3-17;
    end;
    else if type=' P ' then total+1;
    if last then output;
run;
```

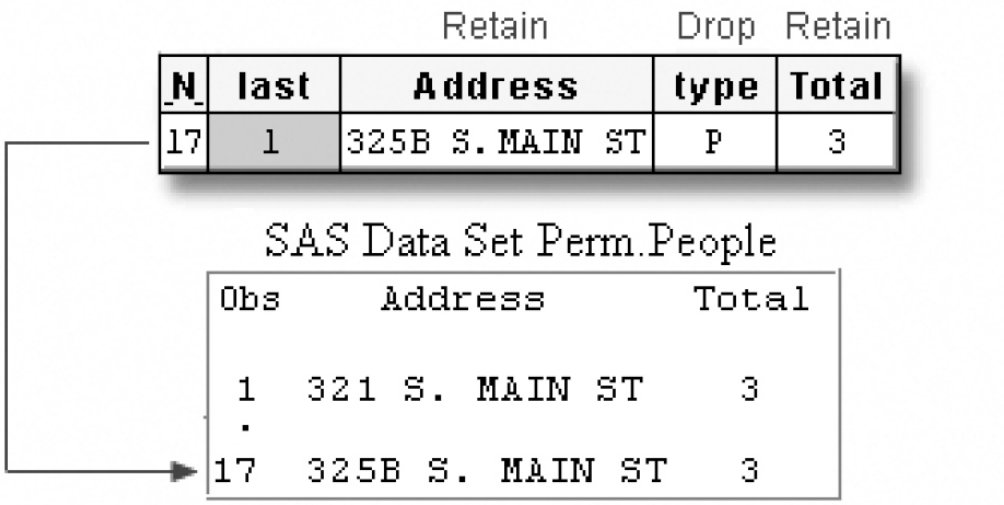
```
>-V--+----10---+----
 H 321 S. MAIN ST
 P MARY E     21  F
 P WILLIAM M 23  M
 P SUSAN K     3  F
 H 324 S. MAIN ST
```



As the last record in the file is read, the variable last is set to **1**. Now that the condition for last is true, the values in the program data vector are written to the data set as the final observation.

```
data perm.people (drop=type);
   infile census end=last;
   retain Address;
   input type $1. @;
         if type = 'H' then do ;
         if _n_ > 1 then outputt>;
         Total=0;
         input Address $3-17;
    end;
    else if type=' P ' then total+1;
    if last then output;
run>;
```





Chapter Summary

Text Summary

Hierarchical Raw Data Files

Raw data files can be hierarchical in structure, consisting of a header record and one or more detail records. You can build a SAS data set from a hierarchical file by creating one observation

- per detail record, storing header data with each observation
- per header record, combining the information from detail records into summary variables.

Creating One Observation per Detail Record

To create one observation per detail record, it is necessary to distinguish between header and detail records. Having a field that identifies the type of the record makes this task easier.

As you write the DATA step, use a RETAIN statement to retain header data in the PDV until the next header record is encountered.

Next, you need to read the field in each record that identifies the record's type. Remember to use the @ line-hold specifier to hold the current value of each record type so that the other values in the record can be read by subsequent INPUT

statements.

Use an IF-THEN statement to check for the condition that the record is a header record. If the record is a header record, you need to execute an INPUT statement to read values for that record.

You can use a subsetting IF statement to check for the condition that the record is a detail record. If the record is a detail record, use another INPUT statement to read values in that record.

Use the DROP= option to exclude the variable that identifies each record's type from the output data set.

Creating One Observation per Header Record

Creating one observation per header record condenses a large amount of information into a concise data set. As you write the DATA step, you need to think about performing several tasks.

As with creating one observation per detail record, use a RETAIN statement to retain header data in the PDV until the next header record is encountered. Then read the field in each record that identifies the record's type. Remember to use the @ line-hold specifier to hold the current record so that the other values in the record can be read by subsequent INPUT statements.

When the record is a header record, multiple statements need to be executed. You can do this by adding a simple DO group to an IF-THEN statement. Within the DO group, you need to

1. determine whether this is the first header record in the external file by using the automatic variable _N_
2. use an OUTPUT statement to write each header record except for the first one to the data set
3. use an assignment statement to create a summary variable, and set its value to *0*
4. add an INPUT statement to read values in the header record
5. close the DO group with an END statement.

When the record is a detail record, you need to define an alternative set of actions. You can do this by adding an ELSE statement after the DO group. As each detail record is read, increment the value of the summary variable by using a sum statement.

After the last detail record is read, there are no more header records to cause the last observation to be written to the data set. Determine when the current record is the last record in an external file by specifying the END= option in the INFILE statement. Again, you can use the DROP= option to exclude the variable that identifies each record's type from the output data set.

Syntax

Syntax to Create One Observation for Each Detail Record

```
LIBNAME libref 'SAS-data-library';
FILENAME fileref 'filename';
DATA SAS-data-set (DROP=variable);

   INFILE file-specification;
   RETAIN variable;
   INPUT variable;
   IF variable='condition' THEN SAS statement;
   IF variable='condition';
       SAS-statement;
RUN;
```

Syntax to Create One Observation for Each Header Record

```
LIBNAME libref 'SAS-data-library';
FILENAME fileref 'filename';
DATA SAS-data-set (DROP= variable);
    INFILE file-specification END=variable;
    RETAIN variable;
    INPUT variable;
    IF variable='condition' THEN DO;
```

```
      IF _N_ > 1 THEN OUTPUT;
      summary-variable=0;
      INPUT variable;
  END;
  ELSE IF variable='condition' THEN
                    summary-variable+expression;
  IF variable THEN OUTPUT;
RUN;
```

Sample Programs

Program to Create One Observation for Each Detail Record

```
libname perm 'c:\records\census2k';
filename census 'c:\records\census2k\survey.dat';
data perm.people(drop=type);
   infile census;
   retain Address;
   input type $1. @;
   if type='H' then input @3 address $15.;
   if type='P';
   input @3 Name $10. @13 Age 3. @16 Gender $1.;
run;
```

Program to Create One Observation for Each Header Record

```
libname perm 'c:\records\census2k';
filename census 'c:\records\census2k\survey.dat';
data perm.residnts (drop=type);
   infile census end=last;
   retain Address;
   input type $1. @;
   if type='H' then do;
      if _n_ > 1 then output;
      Total=0;
      input address $ 3-17;
   end;
   else if type='P' then total+1;
   if last then output;
run;
```

Points to Remember

- As with automatic variables, the END= variable is not written to the data set.
- Values are automatically retained when using a sum statement. Therefore, it may be necessary to set the value of the counter variable back to **0** when a new header is encountered.

Chapter Quiz

Select the best answer for each question. After completing the quiz, check your answers using the answer key in the appendix.

1. When you write a DATA step to create one observation per detail record you need to ?

   a. distinguish between header and detail records.

   b. keep the header record as a part of each observation until the next header record is encountered.

   c. hold the current value of each record type so that the other values in the record can be read.

   d. all of the above

2. Which SAS statement reads the value for code (in the first field), and then holds the value until an INPUT statement reads the remaining value in each observation in the same iteration of the DATA step? ?

```
1---+----10---+----20---+----30
01 Office Supplies
02 Paper Clips
02 Tape
01 Art Supplies
02 Crayons
02 Finger Paint
02 Construction Paper
```

a. `input code $2. @;`

b. `input code $2. @@;`

c. `retain code;`

d. none of the above

3. Which SAS statement checks for the condition that Record equals C and executes a single statement to read the values for Amount? ?

a. `if record=c then input @3 Amount comma7.;`

b. `if record='C' then input @3 Amount comma7.;`

c. `if record='C' then do input @3 Amount comma7.;`

d. `if record=C then do input @3 Amount comma7.;`

4. After the value for code is read in the sixth iteration, which illustration of the program data vector is correct? ?

```
1---+----10---+----20---+----30
H Lettuce
P Green Leaf   Quality Growers
P Iceberg      Pleasant Farm
P Romaine      Quality Growers
H Squash
P Yellow       Tasty Acres
P Zucchini     Pleasant Farm
```

```
data perm.produce (drop=code);
   infile orders;
   retain Vegetable;
```

```
   input code $1. @;
   if code='H' then input @3 vegetable $6.;
   if code='P';
   input @3 Variety : $10. @15 Supplier : $15.;
run;
proc print data=perm.produce;
run;
```

a.

_N_	Vegetable	code	Variety	Supplier
6		P		

b.

_N_	Vegetable	code	Variety	Supplier
6	Squash	P		

c.

_N_	Vegetable	code	Variety	Supplier
6	Squash	H		

d.

_N_	Vegetable	code	Variety	Supplier
6	Squash	H	Yellow	

5. What happens when the fourth iteration of the DATA step is complete? ?

```
1---+----10---+----20---+----30
F Apples
V Gala                  $2.50
V Golden Delicious      $1.99
V Rome                  $2.35
F Oranges
V Navel                 $2.79
V Temple                $2.99
```

```
data perm.orders (drop=type);
   infile produce;
   retain Fruit;
   input type $1. @;
   if type='F' then input @3 fruit $7.;
   if type='V';
   input @3 Variety : $16. @20 Price comma5.;
run;
```

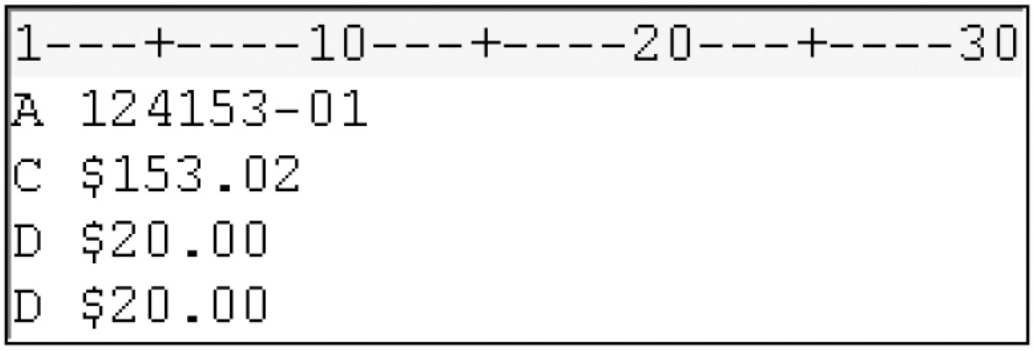
a. All of the values in the program data vector are written to the data set as the third observation.

b. All of the values in the program data vector are written to the data set as the fourth observation.

c. The values for Fruit, Variety, and Price are written to the data set as the third observation.

d. The values for Fruit, Variety, and Price are written to the data set as the fourth observation.

6. Which SAS statement indicates that several other statements should be executed when Record has a value of **A**? ?



a. `if record='A' then do;`

b. `if record=A then do;`

c. `if record='A' then;`

d. `if record=A then;`

7. Which is true for the following statements (X indicates a header record)? ?

```
if code='X' then do;
   if _n_ > 1 then output;
   Total=0;
   input Name $ 3-20;
end;
```

a. _N_ equals the number of times the DATA step has begun to execute.

b. When code='X' and _n_ > 1 are true, an OUTPUT statement is executed.

c. Each header record causes an observation to be written to the data set.

d. a and b

8. What happens when the condition type='P' is false? ?

```
if type='P' then input @3 ID $5. @9 Address $20.;
else if type='V' then input @3 Charge 6.;
```

a. The values for ID and Address are read.

b. The values for Charge are read.

c. type is assigned the value of **V**.

d. The ELSE statement is executed.

9. What happens when last has a value other than zero? ?

```
data perm.househld (drop=code);
   infile citydata end=last;
   retain Address;
   input type $1. @;
   if code='A' then do;
      if _n_ > 1 then output;
      Total=0;
      input address $ 3-17;
   end;
   else if code='N' then total+1;
   if last then output;
run;
```

a. last has a value of **1**.

b. The OUTPUT statement writes the last observation to the data set.

c. The current value of last is written to the DATA set.

d. a and b

10. Based on the values in the program data vector, what happens next? ?

_N_	last	Department	Extension	type	Total	Amount
5	0	Accounting	x3808	D	16.50	.

```
1---+----10---+----20---+----30
D Accounting   x3808
S Paper Clips  $2.50
S Paper        $4.95
S Binders      $9.05
D Personnel    x3810
S Markers      $8.98
S Pencils      $3.35
```

```
data work.supplies (drop=type amount);
   infile orders end=last;
   retain Department Extension;
   input type $1. @;
   if type='D' then do;
      if _n_ > 1 then output;
      Total=0;
      input @3 department $10. @16 extension $5.;
   end;
   else if type='S' then do;
      input @16 Amount comma5.;
      total+amount;
      if last then output;
       end;
run;
```

a. All the values in the program data vector are written to the data set as the first observation.

b. The values for Department, Total, and Extension are written to the data set as the first observation.

c. The values for Department, Total, and Extension are written to the data set as the fourth observation.

d. The value of last changes to **1**.

Answers

1. Correct answer: d

In order to create one observation per detail record, it is necessary to distinguish between header and detail records. Use a RETAIN statement to keep header data as part of each observation until the next header record is encountered. You also need to use the @ line-hold specifier to hold the current record so other values in the record can be read.

2. Correct answer: a

An INPUT statement is used to read the value for code. The single @ sign at the end of the INPUT statement holds the current record for a later INPUT statement in the same iteration of the DATA step.

3. Correct answer: b

The IF-THEN statement defines the condition that Record equals C and executes an INPUT statement to read the value for Amount when the condition is true. C must be enclosed in quotation marks and must be specified exactly as shown because it is a character value.

4. Correct answer: b

The value of Vegetable is retained across iterations of the DATA step. As the sixth iteration begins, the INPUT statement reads the value for code and holds the record, so that the values for Variety and Supplier can be read with an additional INPUT statement.

5. Correct answer: c

This program creates one observation for each detail record. The RETAIN statement retains the value for Fruit as part of each observation until the values for Variety and Price can be read. The DROP= option in the DATA statement prevents the variable for type from being written to the data set.

6. Correct answer: a

The IF-THEN statement defines the condition that Record equals A and specifies a simple DO group. The keyword DO indicates that several executable statements follow until the DO group is closed by an END statement. The value A must be enclosed in quotation marks and specified exactly as shown because it is a character value.

7. Correct answer: d

_N_ is an automatic variable whose value is the number of times the DATA step has begun to execute. The expression _n_ > 1 defines a condition where the DATA step has executed more than once. When the conditions code='X' and _n_ > 1 are true, an OUTPUT statement is executed, and Total is initialized to zero. Thus, each header record except the first one causes an observation to be written to the data set.

8. Correct answer: d

The condition is false, so the values for ID and Address are not read. Instead, the ELSE statement is executed and defines another condition which may or may not be true.

9. Correct answer: d

You can determine when the current record is the last record in an external file by specifying the END= option in the INFILE statement. last is a temporary numeric variable whose value is zero until the last line is read. last has a value of **1** after the last line is read. Like automatic variables, the END= variable is not written to the data set.

10. Correct answer: b

This program creates one observation for each header record and combines information from each detail record into the summary variable, Total. When the value of type is **D** and the value of _N_ is greater than **1**, the OUTPUT statement executes, and the values for Department, Total and Extension are written to the data set as the first observation. The variables _N_, last, type and Amount are not written to the data set.