STAT604

Lesson SAS 08

5.13 Quiz

In the DATA step below, the expression should calculate **TotalValue** by multiplying **SharePrice** by **MyShares**. Will SAS automatically convert SharePrice to a numeric value?

```
data Input_Quiz;
   SharePrice = "$130.25";
   MyShares = 125;
   TotalValue = SharePrice * MyShares

run;
```

5.13 Quiz – Correct Answer

Fill in the missing expression in the DATA step below. The expression should calculate **TotalValue** by multiplying **SharePrice** by **MyShares**.

```
data Input_Quiz;
   SharePrice = "$130.25";
   MyShares = 125;
   TotalValue = input(SharePrice,comma7.)*
                MyShares;
run;
```

The COMMAw.d informat reads numeric values and removes embedded commas, blanks, dollar signs, percent signs, hyphens, and close parentheses from the input data. The COMMAw.d informat converts an open parenthesis at the beginning of a field to a minus sign.

p205d17

Explicit Character-to-Numeric Conversion

Continue with the business scenario by creating the variables **EmpID**, **Bonus**, and **HireDate**.

Use the INPUT function to explicitly convert character values to numeric.

```
data hrdata;
   keep EmpID GrossPay Bonus HireDate;
   set orion.convert;
   EmpID = input(ID,5.)+11000;
   Bonus = input(GrossPay,comma6.)*.10;
   HireDate = input(Hired,mmddyy10.);
run;
```

p205d18

Explicit Character-to-Numeric Conversion

```
proc print data=hrdata noobs;
   var EmpID GrossPay Bonus HireDate;
run;
```

PROC PRINT Output

```
               Gross                       Hire
     EmpID     Pay            Bonus        Date

     11036     52,000          5200       16174
     11048     32,000          3200       17038
     11052     49,000          4900       16595
```

SAS date values

Explicit Character-to-Numeric Conversion

```
proc print data=hrdata noobs;
   var EmpID GrossPay Bonus HireDate;
   format HireDate mmddyy10.;
run;
```

PROC PRINT Output

```
                 Gross
       EmpID     Pay           Bonus        HireDate

       11036     52,000         5200      04/13/2004
       11048     32,000         3200      08/25/2006
       11052     49,000         4900      06/08/2005
```

What data type is **`GrossPay`**?

Converting a Variable to Another Data Type

```
proc contents data=hrdata;
run;
```

Partial PROC CONTENTS Output

```
Alphabetic List of Variables and Attributes

   #    Variable     Type     Len

   3    Bonus        Num        8
   2    EmpID        Num        8
   1    GrossPay     Char       6
   4    HireDate     Num        8
```

How can you convert **`GrossPay`** to a numeric variable with the same name?

Poll

Quiz

5.14 Quiz

Will this statement convert **GrossPay** to numeric?

```
GrossPay=input(GrossPay,comma6.);
```

Open and run the program **p205a02**. Did **GrossPay** become a numeric variable?

5.14 Quiz – Correct Answer

Will this statement convert **`GrossPay`** to numeric?

```
GrossPay=input(GrossPay,comma6.);
```

Open and run the program **p205a02**. Did **`GrossPay`** become a numeric variable?

No, `GrossPay` remained a character variable.

Converting a Variable to Another Data Type

```
GrossPay=input(GrossPay,comma6.);
```

⚠ This assignment statement does **not** change **`GrossPay`** from a character variable to a numeric variable.

A variable is character or numeric. After the variable’s type is established, it cannot be changed.

By following three steps, you can create a new variable with the same name and a different type.

Converting a Variable to Another Data Type

Step 1: Use the RENAME= data set option to rename the variable that you want to convert.

```
data hrdata;
   set orion.convert(rename=(GrossPay=
                               CharGross));
run;
```

General form of the RENAME data set option:

SAS-data-set(RENAME=(*old-name*=*new-name*))

Converting a Variable to Another Data Type

Step 2: Use the INPUT function in an assignment statement to create a new variable with the original name of the variable that you renamed.

```
data hrdata;
   set orion.convert(rename=(GrossPay=
                             CharGross));
   GrossPay=input(CharGross,comma6.);
run;
```

Converting a Variable to Another Data Type

Step 3: Use a DROP= data set option in the DATA statement to exclude the original variable from the output SAS data set.

```
data hrdata(drop=CharGross);
   set orion.convert(rename=(GrossPay=
                             CharGross));
   GrossPay=input(CharGross,comma6.);
run;
```

The compilation for this program shows the PDV being created with a numeric **`GrossPay`** variable.

p205d19

Converting a Variable: Compilation

```
data hrdata(drop=CharGross);
   set orion.convert(rename=(GrossPay=
                             CharGross));
   GrossPay=input(CharGross,comma6.);
run;
```

Partial PDV

ID	CharGross	Hired
$ 5	$ 6	$ 7

Converting a Variable: Compilation

```
data hrdata(drop=CharGross);
   set orion.convert(rename=(GrossPay=
                             CharGross));
   GrossPay=input(CharGross,comma6.);
run;
```

Partial PDV

ID $ 5	CharGross $ 6	Hired $ 7	GrossPay N 8

...

Converting a Variable: Compilation

```
data hrdata(drop=CharGross);
   set orion.convert(rename=(GrossPay=
                          CharGross));
   GrossPay=input(CharGross,comma6.);
run;
```

Partial PDV

ID	CharGross	Hired	GrossPay
$ 5	D $ 6	$ 7	N 8

Continue with the Business Scenario

The **`orion.convert`** data set contains a numeric variable **`Code`** (area code) and a character variable **`Mobile`** (mobile telephone number). Create a character variable, **`Phone`**, that contains the area code in parentheses followed by the mobile telephone number.

For the first try at creating the **`Phone`** variable, let SAS automatically handle the conversion.

Automatic Numeric-to-Character Conversion

Partial list of `orion.convert`

Code	Mobile
N 8	$ 8
303	393-0956
919	770-8292
301	449-5239

```
data hrdata;
   keep Phone Code Mobile;
   set orion.convert;
   Phone='(' !! Code !! ') ' !! Mobile;
run;
```

SAS automatically converts the numeric values in **Code** into character values.

p205d20

Automatic Numeric-to-Character Conversion

Partial Log

```
14   data hrdata;
15     keep Phone Code Mobile;
16     set orion.convert;
17     Phone='(' !! Code !! ') ' !! Mobile;
18   run;

NOTE: Numeric values have been converted to character values
    at the places given by:
      (Line):(Column).
      17:16
NOTE: There were 3 observations read from the data set
    ORION.CONVERT.
NOTE: The data set WORK.HRDATA has 3 observations and 3
    variables.
```

Automatic Numeric-to-Character Conversion

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

PROC PRINT Output

```
Code    Mobile               Phone

 303    393-0956    (         303) 393-0956
 919    770-8292    (         919) 770-8292
 301    449-5239    (         301) 449-5239
```

Why did SAS insert the extra blanks before the area code?

Automatic Numeric-to-Character Conversion

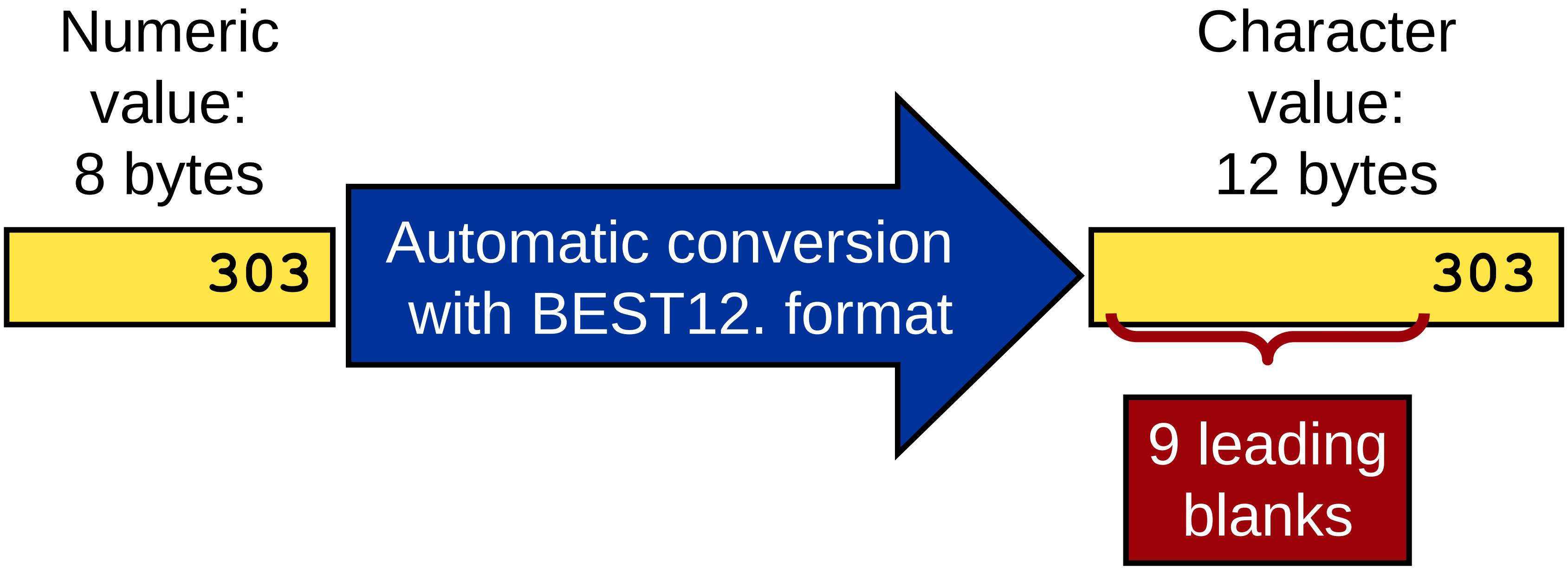
SAS automatically converts a numeric value to a character value when the numeric value is used in a character context, such as

- assignment to a character variable
- a concatenation operation
- a function that accepts character arguments.

Automatic Numeric-to-Character Conversion

The automatic conversion

- uses the BEST12. format
- right-aligns the resulting character value.



Numeric value: 8 bytes

303

Automatic conversion with BEST12. format

Character value: 12 bytes

303

9 leading blanks

Automatic Numeric-to-Character Conversion

```
data hrdata;
   keep Phone Code Mobile;
   set orion.convert;
   Phone='(' !! Code !! ') ' !! Mobile;
run;
```

Partial PDV

Phone $ 23
(303) 393-0956

9 leading blanks

To fix this, use the PUT function to explicitly control the numeric-to-character conversion

The PUT Function

The PUT function writes values with a specific format.

CharVar=PUT(*source*,*format*);

The PUT function returns the value produced when *source* is written with *format*.

EXAMPLE:

major=‘STAT’;

MajorName=PUT(major, $majfmt.);

Resulting value of **MajorName** - Statistics

The PUT Function – Example

This DATA step shows examples of the PUT function.

```
data conversion;
   NVar1=614;
   NVar2=55000;
   NVar3=366;
   CVar1=put(NVar1,3.);
   CVar2=put(NVar2,dollar7.);
   CVar3=put(NVar3,date9.);
run;
```

p205d21

The PUT Function – Example

```
proc contents data=conversion varnum;
run;
```

The VARNUM option in the PROC CONTENTS statement prints a list of the variables by their logical position in the data set.

Partial PROC CONTENTS Output

```
 Variables in Creation Order

#    Variable    Type    Len

1    NVar1       Num       8
2    NVar2       Num       8
3    NVar3       Num       8
4    CVar1       Char      3
5    CVar2       Char      7
6    CVar3       Char      9
```

The PUT Function – Example

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

PROC PRINT Output

```
NVar1     NVar2     NVar3     CVar1     CVar2       CVar3

 614      55000      366       614     $55,000   01JAN1961
```

Explicit Numeric-to-Character Conversion

```
data hrdata;
   keep Phone Code Mobile;
   set orion.convert;
   Phone='(' !! put(Code,3.) !! ') '
               !! Mobile;
run;
```

Partial Log

```
42   data hrdata;
43     keep Phone Code Mobile;
44     set orion.convert;
45     Phone='(' !! put(Code,3.) !! ') ' !! Mobile;
46   run;

NOTE: The data set WORK.HRDATA has 3 observations
      and 3 variables.
```

p205d22

Explicit Numeric-to-Character Conversion

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

PROC PRINT Output

```
Code    Mobile          Phone

 303    393-0956    (303) 393-0956
 919    770-8292    (919) 770-8292
 301    449-5239    (301) 449-5239
```

The CAT Functions and Numeric Conversion

The CAT family of functions converts any numeric argument to a character string by using the BEST12. format and then **removing any leading blanks**. No note is written to the log.

This assignment statement using CAT:

```
Phone=cat('(',Code,') ',Mobile);
```

gives equivalent results to this statement:

```
Phone='(' !! put(Code,3.) !! ') ' !! Mobile;
```

demo

Now you can write the complete SAS program to convert the personnel data.

Convert HR Data – Complete Program

```
data hrdata;
   keep EmpID GrossPay Bonus Phone HireDate;
   set orion.convert(rename=(GrossPay=
                             CharGross));
   EmpID = input(ID,5.)+11000;
   GrossPay = input(CharGross,comma6.);
   Bonus = GrossPay*.10;
   HireDate = input(Hired,mmddyy10.);
   Phone=cat('(',Code,') ',Mobile);
run;

proc print data=hrdata noobs;
   var EmpID GrossPay Bonus Phone HireDate;
   format HireDate mmddyy10.;
run;
```

p205d23

Convert HR Data – Complete Program

PROC PRINT Output

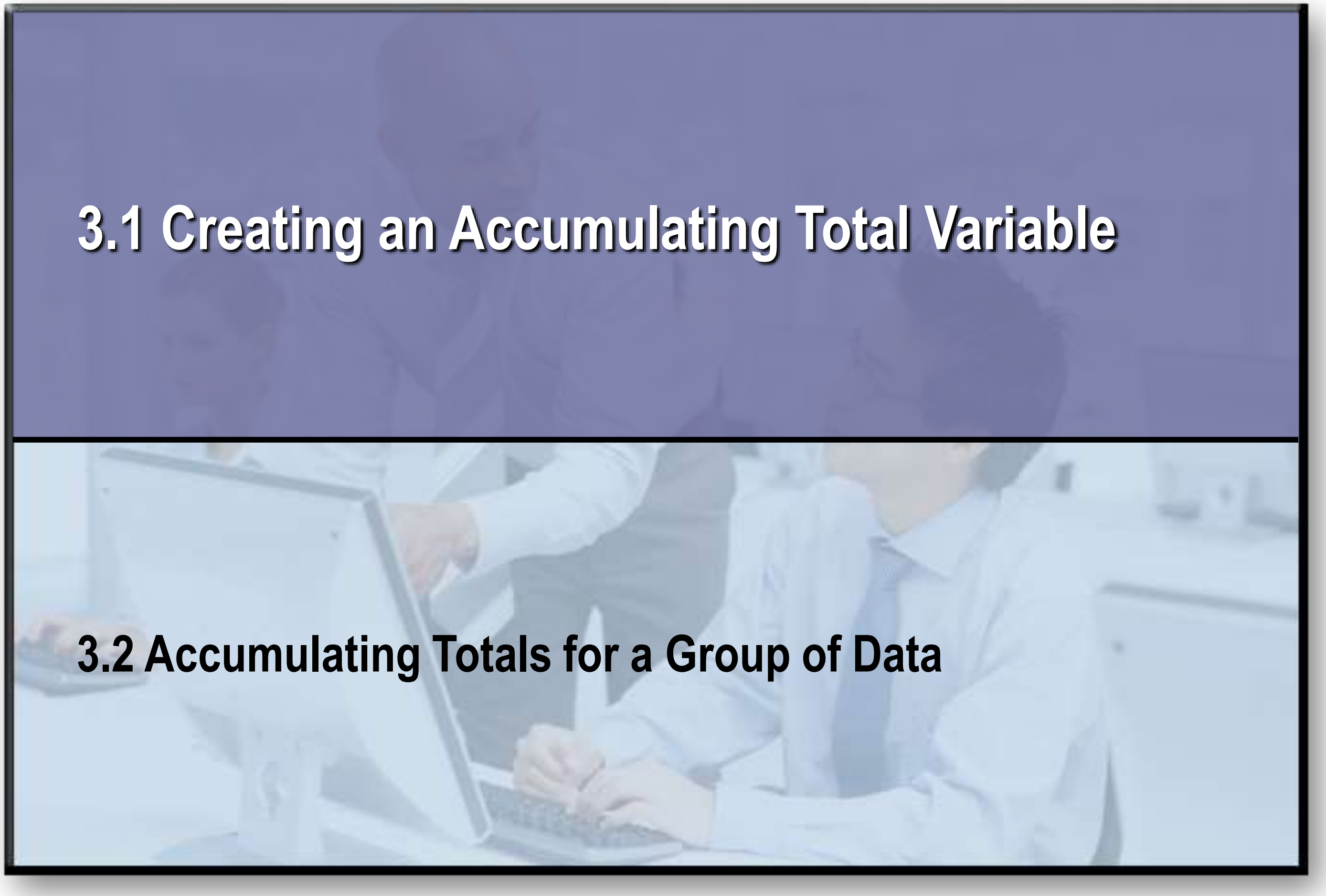
```
             Gross
    EmpID     Pay      Bonus         Phone             HireDate

    11036    52000      5200    (303) 393-0956       04/13/2002
    11048    32000      3200    (919) 770-8292       08/25/1998
    11052    49000      4900    (301) 449-5239       06/08/2001
```

Chapter 3: Summarizing Data

3.1 Creating an Accumulating Total Variable

3.2 Accumulating Totals for a Group of Data

Chapter 3: Summarizing Data



3.1 Creating an Accumulating Total Variable

3.2 Accumulating Totals for a Group of Data

Objectives

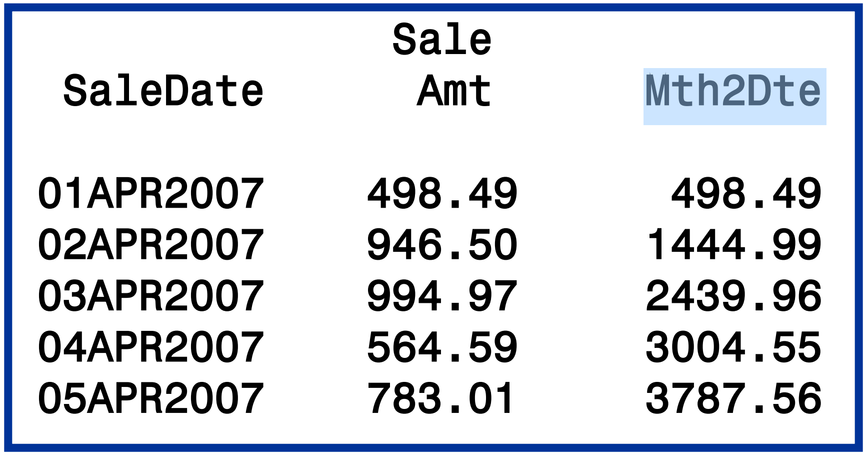
- Explain how SAS initializes the value of a variable in the PDV.
- Prevent reinitialization of a variable in the PDV.
- Create an accumulating variable.

Business Scenario

A retail manager for Orion Star Sportswear asked to see her department’s daily sales for April, as well as a month-to-date total for each day.

Create a new data set, **`mnthtot`**, that includes the month-to-date total (**`Mth2Dte`**) for each day.

Partial Listing of **`mnthtot`**



```
                Sale
 SaleDate        Amt         Mth2Dte

01APR2007     498.49          498.49
02APR2007     946.50         1444.99
03APR2007     994.97         2439.96
04APR2007     564.59         3004.55
05APR2007     783.01         3787.56
```

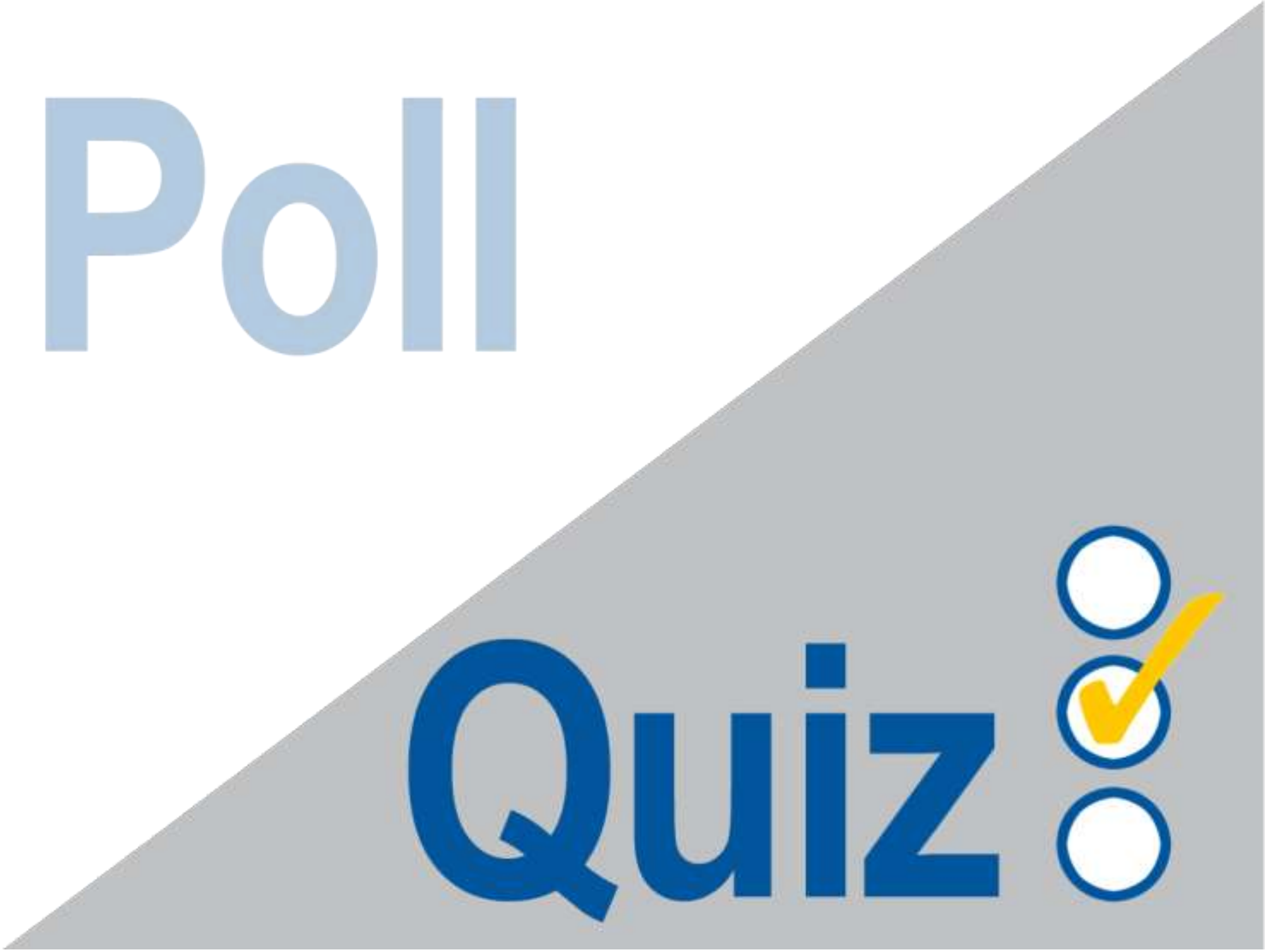
Input Data

The SAS data set **orion.aprsales** contains daily sales data from the Orion Star Sportswear department.

Partial Listing of **orion.aprsales**

```
  SaleDate     SaleAmt

01APR2007       498.49
02APR2007       946.50
03APR2007       994.97
04APR2007       564.59
05APR2007       783.01
06APR2007       228.82
07APR2007       930.57
```

One observation for each day in April shows the date (**SaleDate**) and the total sales for that day (**SaleAmt**).


Poll
Quiz

3.01 Quiz

Open and submit the program in **p203a01**. Does this program create the correct values for **`Mth2Dte`**?

```
data mnthtot;
   set orion.aprsales;
   Mth2Dte=Mth2Dte+SaleAmt;
run;
```

p203a01

3.01 Quiz – Correct Answer

Open and submit the program in p203a01. Does this program create the correct values for **Mth2Dte**?

```
data mnthtot;
   set orion.aprsales;
   Mth2Dte=Mth2Dte+SaleAmt;
run;
```

Partial Listing of **mnthtot**

```
                 Sale
  SaleDate        Amt       Mth2Dte

 01APR2007     498.49          .
 02APR2007     946.50          .
 03APR2007     994.97          .
 04APR2007     564.59          .
```

No, the program creates Mth2Dte with all missing values.

p203a01

Creating an Accumulating Variable

By default, variables created with an assignment statement are initialized to missing at the top of each iteration of the DATA step.

```
Mth2Dte=Mth2Dte+SaleAmt;
```

`Mth2Dte` is an example of an *accumulating variable* that needs to keep its value from one observation to the next.

The RETAIN Statement

The RETAIN statement prevents SAS from reinitializing the values of new variables at the top of the DATA step.

General form of the RETAIN statement:

RETAIN *variable-name <initial-value*> …;

Previous values of retained variables are available for processing across iterations of the DATA step.

The RETAIN Statement – Details

The RETAIN statement

- retains the value of the variable in the PDV across iterations of the DATA step
- initializes the retained variable to missing before the first iteration of the DATA step if an initial value is not specified
- is a compile-time-only statement.

✎ The RETAIN statement has no effect on variables that are read with SET, MERGE, or UPDATE statements; variables read from SAS data sets are automatically retained.

Create an Accumulating Variable

Retain the values of **`Mth2Dte`** and set an initial value.

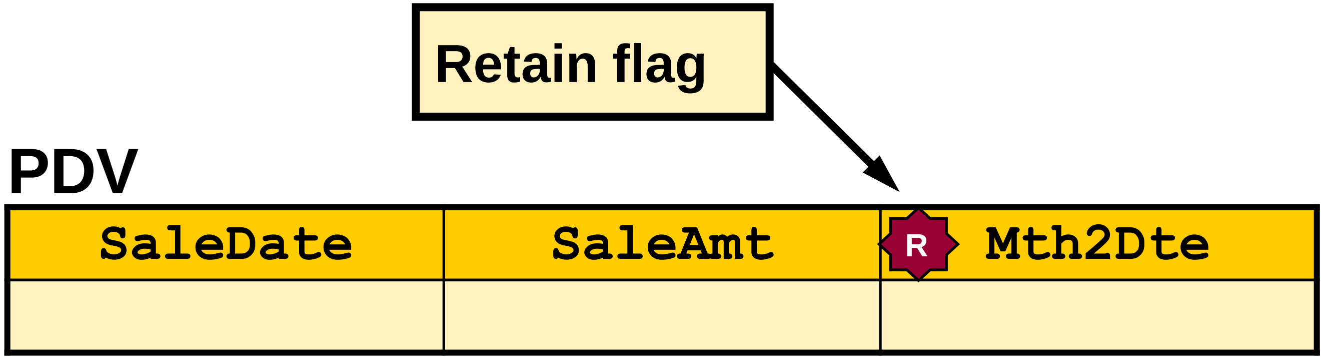
```
data mnthtot;
   set orion.aprsales;
   retain Mth2Dte 0;
   Mth2Dte=Mth2Dte+SaleAmt;
run;
```

⚠ If you do not supply an initial value, all the values of **`Mth2Dte`** will be missing.

p203d02

Compilation: Create an Accumulating Variable

```
data mnthtot;
   set orion.aprsales;
   retain Mth2Dte 0;
   Mth2Dte=Mth2Dte+SaleAmt;
run;
```



Retain flag

PDV

SaleDate	SaleAmt	R Mth2Dte

...

Execution: Create an Accumulating Variable

```
SaleDate     SaleAmt

  17257      498.49
  17258      946.50
  17259      994.97
  17260      564.59
  17261      783.01
```

```
data mnthtot;
   set orion.aprsales;
   retain Mth2Dte 0;
   Mth2Dte=Mth2Dte+SaleAmt;
run;
```

Initialize PDV.

PDV

SaleDate	SaleAmt	R Mth2Dte
.	.	0

...

Execution: Create an Accumulating Variable

SaleDate	SaleAmt
17257	498.49
17258	946.50
17259	994.97
17260	564.59
17261	783.01

```
data mnthtot;
   set orion.aprsales;
   retain Mth2Dte 0;
   Mth2Dte=Mth2Dte+SaleAmt;
run;
```

PDV

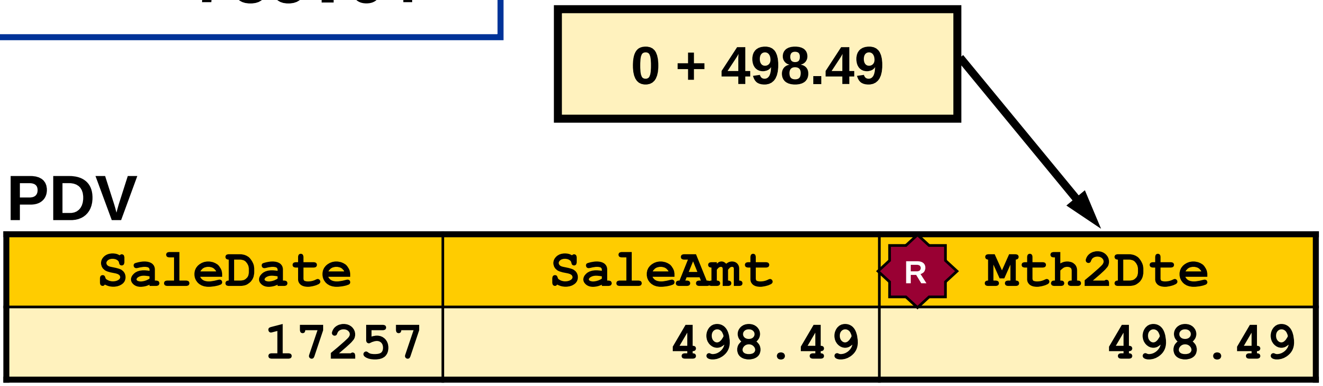
SaleDate	SaleAmt	R Mth2Dte
17257	498.49	0

...

Execution: Create an Accumulating Variable

SaleDate	SaleAmt
17257	498.49
17258	946.50
17259	994.97
17260	564.59
17261	783.01

```
data mnthtot;
    set orion.aprsales;
    retain Mth2Dte 0;
    Mth2Dte=Mth2Dte+SaleAmt;
run;
```



0 + 498.49

PDV

SaleDate	SaleAmt	R Mth2Dte
17257	498.49	498.49

Execution: Create an Accumulating Variable

SaleDate	SaleAmt
17257	498.49
17258	946.50
17259	994.97
17260	564.59
17261	783.01

```
data mnthtot;
    set orion.aprsales;
    retain Mth2Dte 0;
    Mth2Dte=Mth2Dte+SaleAmt;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

SaleDate	SaleAmt	R Mth2Dte
17257	498.49	498.49

Write observation to **mnthtot**

...

Execution: Create an Accumulating Variable

SaleDate	SaleAmt
17257	498.49
17258	946.50
17259	994.97
17260	564.59
17261	783.01

```
data mnthtot;
    set orion.aprsales;
    retain Mth2Dte 0;
    Mth2Dte=Mth2Dte+SaleAmt;
run;
```

Mth2Dte is not reinitialized.

PDV

SaleDate	SaleAmt	R Mth2Dte
17258	498.49	498.49

...

Execution: Create an Accumulating Variable

SaleDate	SaleAmt
17257	498.49
17258	946.50
17259	994.97
17260	564.59
17261	783.01

```
data mnthtot;
   set orion.aprsales;
   retain Mth2Dte 0;
   Mth2Dte=Mth2Dte+SaleAmt;
run;
```

PDV

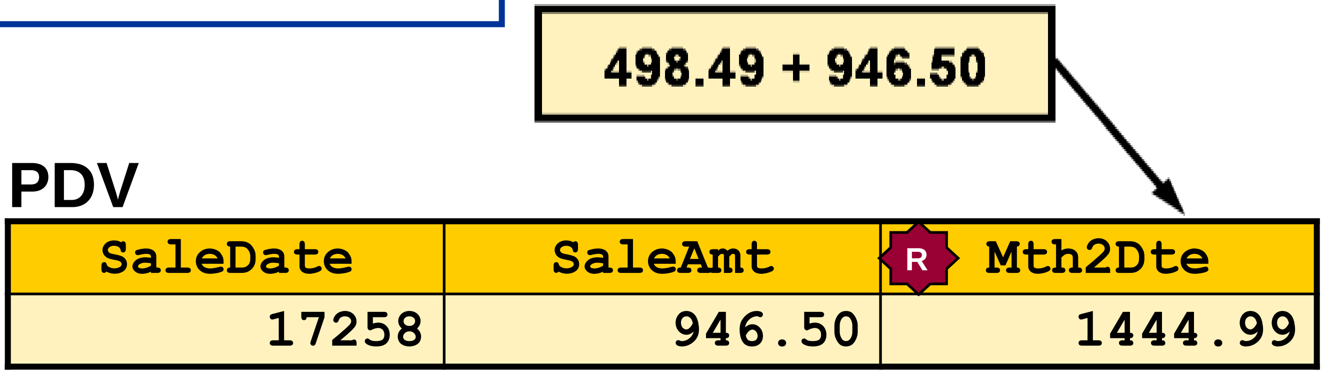
SaleDate	SaleAmt	R Mth2Dte
17258	946.50	498.49

...

Execution: Create an Accumulating Variable

SaleDate	SaleAmt
17257	498.49
17258	946.50
17259	994.97
17260	564.59
17261	783.01

```
data mnthtot;
   set orion.aprsales;
   retain Mth2Dte 0;
   Mth2Dte=Mth2Dte+SaleAmt;
run;
```



498.49 + 946.50

PDV

SaleDate	SaleAmt	R Mth2Dte
17258	946.50	1444.99

...

Execution: Create an Accumulating Variable

SaleDate	SaleAmt
17257	498.49
17258	946.50
17259	994.97
17260	564.59
17261	783.01

```
data mnthtot;
   set orion.aprsales;
   retain Mth2Dte 0;
   Mth2Dte=Mth2Dte+SaleAmt;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

SaleDate	SaleAmt	R Mth2Dte
17258	946.50	1444.99

Write observation to **mnthtot**

...

Execution: Create an Accumulating Variable

SaleDate	SaleAmt
17257	498.49
17258	946.50
17259	994.97
17260	564.59
17261	783.01

```
data mnthtot;
    set orion.aprsales;
    retain Mth2Dte 0;
    Mth2Dte=Mth2Dte+SaleAmt;
run;
```

Mth2Dte is not reinitialized.

PDV

SaleDate	SaleAmt	R Mth2Dte
17258	946.50	1444.99

...

Execution: Create an Accumulating Variable

SaleDate	SaleAmt
17257	498.49
17258	946.50
17259	994.97
17260	564.59
17261	783.01

```
data mnthtot;
   set orion.aprsales;
   retain Mth2Dte 0;
   Mth2Dte=Mth2Dte+SaleAmt;
run;
```

Continue until EOF.

PDV

SaleDate	SaleAmt	R Mth2Dte
17258	946.50	1444.99

Create an Accumulating Variable

```
proc print data=mnthtot noobs;
   format SaleDate date9.;
run;
```

Partial PROC PRINT Output

```
                Sale
 SaleDate        Amt       Mth2Dte

01APR2007     498.49       498.49
02APR2007     946.50      1444.99
03APR2007     994.97      2439.96
04APR2007     564.59      3004.55
05APR2007     783.01      3787.56
```

p203d02

Poll

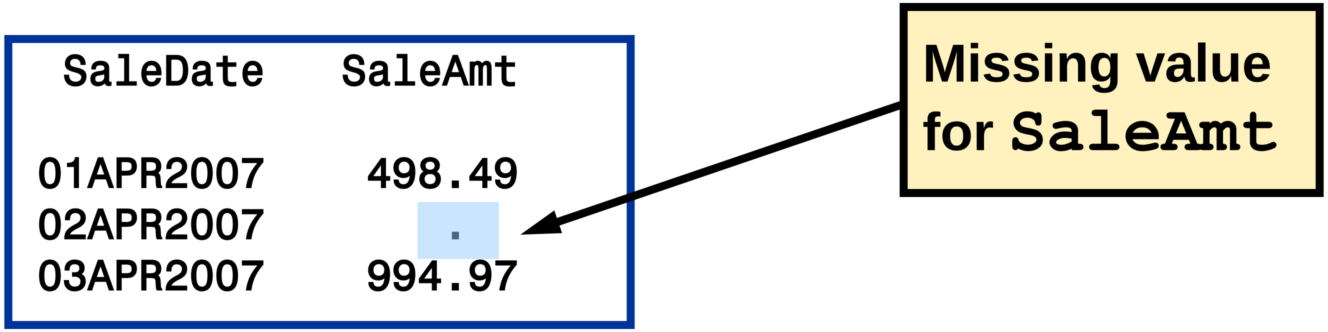
Quiz

Setup for the Poll

What happens if there are missing values for **`SaleAmt`**?

Open and submit **p203a02** and examine the output.

Partial listing of input data



```
  SaleDate    SaleAmt

01APR2007      498.49
02APR2007           .
03APR2007      994.97
```

Missing value for `SaleAmt`

3.02 Multiple Choice Poll

What effect did the missing value for **`SaleAmt`** have on **`Mth2Dte`**?

a. The missing value was ignored; **`Mth2Dte`** values were not affected.

b. The missing value will cause the DATA step to stop processing.

c. The missing value will cause the subsequent values for **`Mth2Dte`** to be set to missing.

3.02 Multiple Choice Poll – Correct Answer

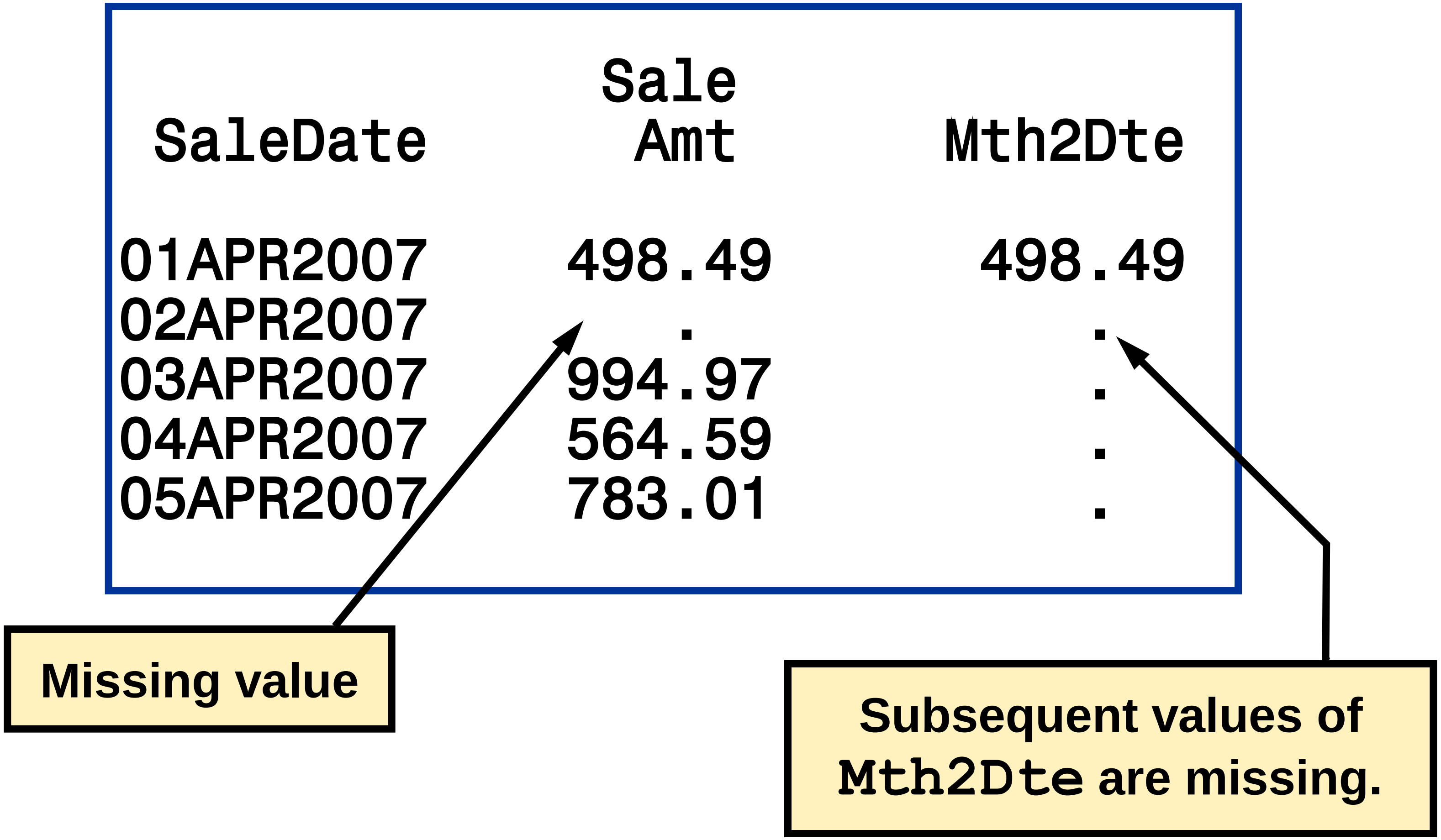
What effect did the missing value for **SaleAmt** have on **Mth2Dte**?

a. The missing value was ignored; **Mth2Dte** values were not affected.

b. The missing value will cause the DATA step to stop processing.

[✓ c.] The missing value will cause the subsequent values for **Mth2Dte** to be set to missing.

Undesirable Output: Missing Values



```
              Sale
 SaleDate      Amt      Mth2Dte

01APR2007    498.49      498.49
02APR2007       .           .
03APR2007    994.97         .
04APR2007    564.59         .
05APR2007    783.01         .
```

Missing value

Subsequent values of `Mth2Dte` are missing.

The SUM Function

A RETAIN statement along with a SUM function in an assignment statement can be used to create **Mth2Dte**.

```
retain Mth2Dte 0;
Mth2Dte=sum(Mth2Dte,SaleAmt);
```

✎ The SUM function ignores missing values.

The Sum Statement

When you create an accumulating variable, a better alternative is to use the sum statement.

General form of the sum statement:

variable **+** *expression***;**

Example:

```
Mth2Dte+SaleAmt;
```

Functionally equivalent to:

```
retain Mth2Dte 0;
Mth2Dte=sum(Mth2Dte,SaleAmt);
```

The Sum Statement – Details

The sum statement

- creates the variable on the left side of the plus sign if it does not already exist
- initializes the variable to zero before the first iteration of the DATA step
- automatically retains the variable
- adds the value of *expression* to the variable at execution
- ignores missing values.

The Sum Statement – Example

Use the sum statement to create **Mth2Dte**.

```
data mnthtot2;
   set work.aprsales2;
   Mth2Dte+SaleAmt;
run;
```

Specifics about **Mth2Dte**:

- Initialized to zero
- Automatically retained
- Increased by the value of **SaleAmt** for each observation
- Ignored missing values of **SaleAmt**

p203d03

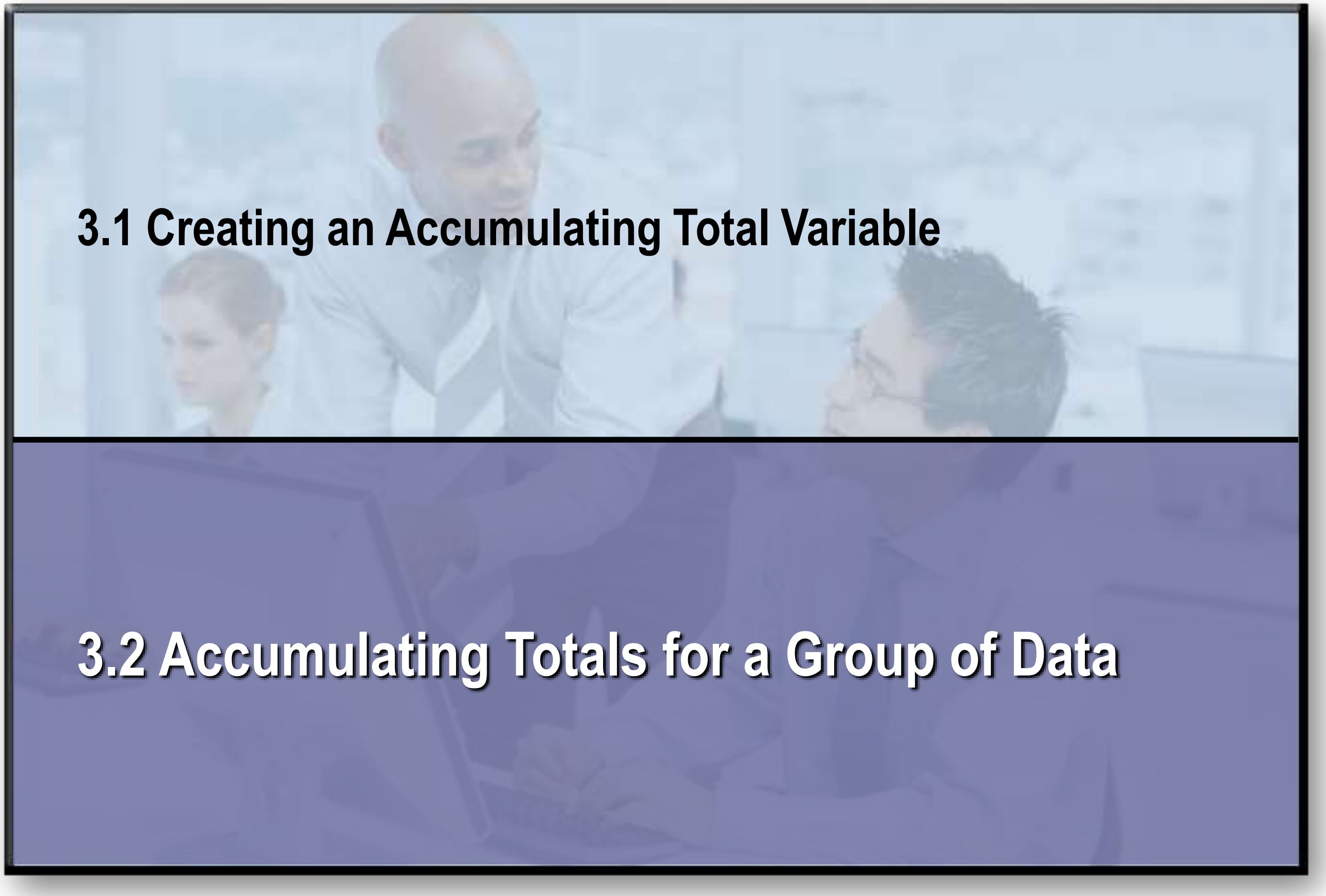
The Sum Statement – Example

```
proc print data=mnthtot2 noobs;
   format SaleDate date9.;
run;
```

Partial PROC PRINT Output

```
 SaleDate      SaleAmt      Mth2Dte

01APR2007       498.49       498.49
02APR2007          .         498.49
03APR2007       994.97      1493.46
04APR2007       564.59      2058.05
05APR2007       783.01      2841.06
```

Chapter 3: Summarizing Data



3.1 Creating an Accumulating Total Variable

3.2 Accumulating Totals for a Group of Data

Objectives

- Define First. and Last. processing.
- Calculate an accumulating total for groups of data.
- Use a subsetting IF statement to output selected observations.

Business Scenario

The SAS data set **orion.specialsals** contains information about employees working on special projects.

Partial Listing of **orion.specialsals**

```
Employee_
    ID      Salary    Dept

  110004     42000    HUMRES
  110009     34000    ENGINR
  110011     27000    FINANC
  110036     20000    ENGINR
  110037     19000    ENGINR
```

The **Salary** variable represents the portion of the employee's salary allocated to the project. An analyst would like to see these salary totals by department.

Desired Output

Create a new data set, **deptsals**, that has the total salaries for each department.

Listing of **deptsals**

```
  Dept        DeptSal

ADMIN          410000
ENGINR         163000
FINANC         318000
HUMRES         181000
SALES          373000
```

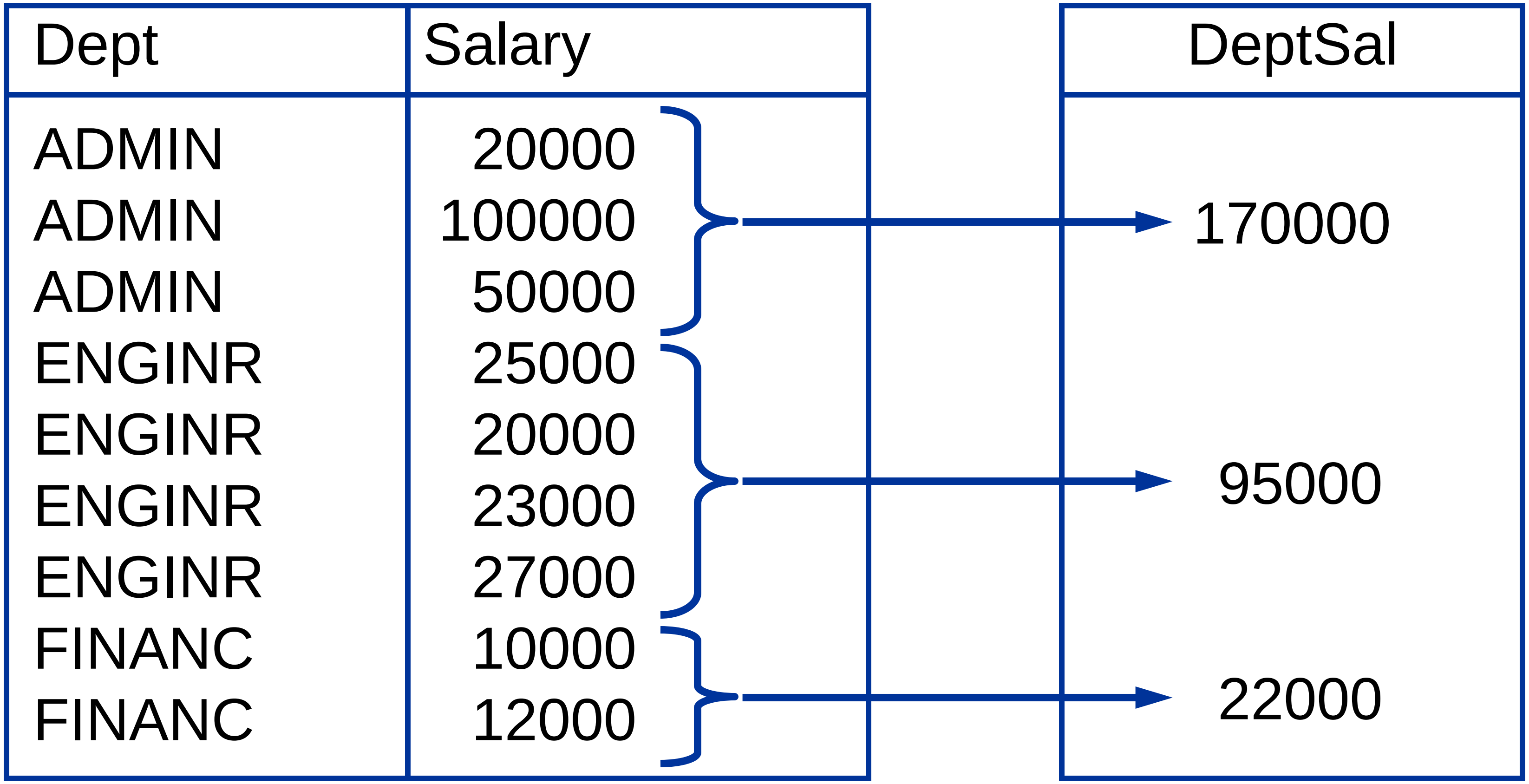
Processing Needed

Dept	Salary
ADMIN	20000
ADMIN	100000
ADMIN	50000
ENGINR	25000
ENGINR	20000
ENGINR	23000
ENGINR	27000
FINANC	10000
FINANC	12000

DeptSal

Step 1: Sort the data by `Dept`.

Processing Needed



Dept	Salary
ADMIN	20000
ADMIN	100000
ADMIN	50000
ENGINR	25000
ENGINR	20000
ENGINR	23000
ENGINR	27000
FINANC	10000
FINANC	12000

DeptSal
170000
95000
22000

Step 2: Summarize the observations by department groups.

The SORT Procedure (Overview)

You can rearrange the observations into groups using the SORT procedure.

General form of a PROC SORT step:

```
PROC SORT DATA=input-SAS-data-set
                      <OUT=output-SAS-data-set>;
         BY <DESCENDING> BY-variable ...;
RUN;
```

The SORT Procedure

The SORT procedure

- rearranges the observations in a SAS data set
- replaces the input data set by default
- can create a new data set that is a sorted copy
- can sort on multiple variables
- can sort in ascending (default) or descending order
- does not generate printed output.

BY-Group Processing

The BY statement in the DATA step enables SAS to process data in groups.

General form of a BY statement in a DATA step:

```
DATA output-SAS-data-set;
      SET input-SAS-data-set;
      BY BY-variable …;
      <additional SAS statements>
RUN;
```

BY-Group Processing

This is a good start for the SAS program …

```
proc sort data=orion.specialsals
          out=salsort;
   by Dept;
run;

data deptsals(keep=Dept DeptSal);
   set salsort;
   by Dept;
   <additional SAS statements>
run;
```

Step 1: Sort by Dept

Step 2: Process by Dept groups

…but you need some way to identify the beginning and end of each department's group of observations.

First. and Last. Values

A BY statement in a DATA step creates two temporary variables for each variable listed in the BY statement.

General form of the First. and Last. variables:

First.*BY-variable*
Last.*BY-variable*

- The First. variable has a value of 1 for the *first* observation in a BY group; otherwise, it equals 0.
- The Last. variable has a value of 1 for the *last* observation in a BY group; otherwise, it equals 0.

First. / Last. Values – 1st DATA Step Iteration

Dept	Salary
ADMIN	20000
ADMIN	100000
ADMIN	50000
ENGINR	25000
ENGINR	20000
ENGINR	23000
ENGINR	27000
FINANC	10000
FINANC	12000

First.Dept
1

Last.Dept
?

How can SAS determine the value for `Last.Dept`?

...

First. / Last. Values – 1st DATA Step Iteration

Dept	Salary
ADMIN	20000
ADMIN	100000
ADMIN	50000
ENGINR	25000
ENGINR	20000
ENGINR	23000
ENGINR	27000
FINANC	10000
FINANC	12000

First.Dept
1

Last.Dept
0

SAS looks ahead at the next observation to determine `Last.Dept` value.

...

First. / Last. Values – 2nd DATA Step Iteration

Dept	Salary
ADMIN	20000
ADMIN	100000
ADMIN	50000
ENGINR	25000
ENGINR	20000
ENGINR	23000
ENGINR	27000
FINANC	10000
FINANC	12000

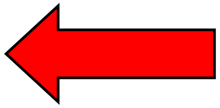
First.Dept
0

Last.Dept
0

...

First. / Last. Values – 3rd DATA Step Iteration

Dept	Salary
ADMIN	20000
ADMIN	100000
ADMIN	50000
ENGINR	25000
ENGINR	20000
ENGINR	23000
ENGINR	27000
FINANC	10000
FINANC	12000



First.Dept
0

Last.Dept
1

...

First. / Last. Values – 4th DATA Step Iteration

Dept	Salary
ADMIN	20000
ADMIN	100000
ADMIN	50000
ENGINR	25000
ENGINR	20000
ENGINR	23000
ENGINR	27000
FINANC	10000
FINANC	12000

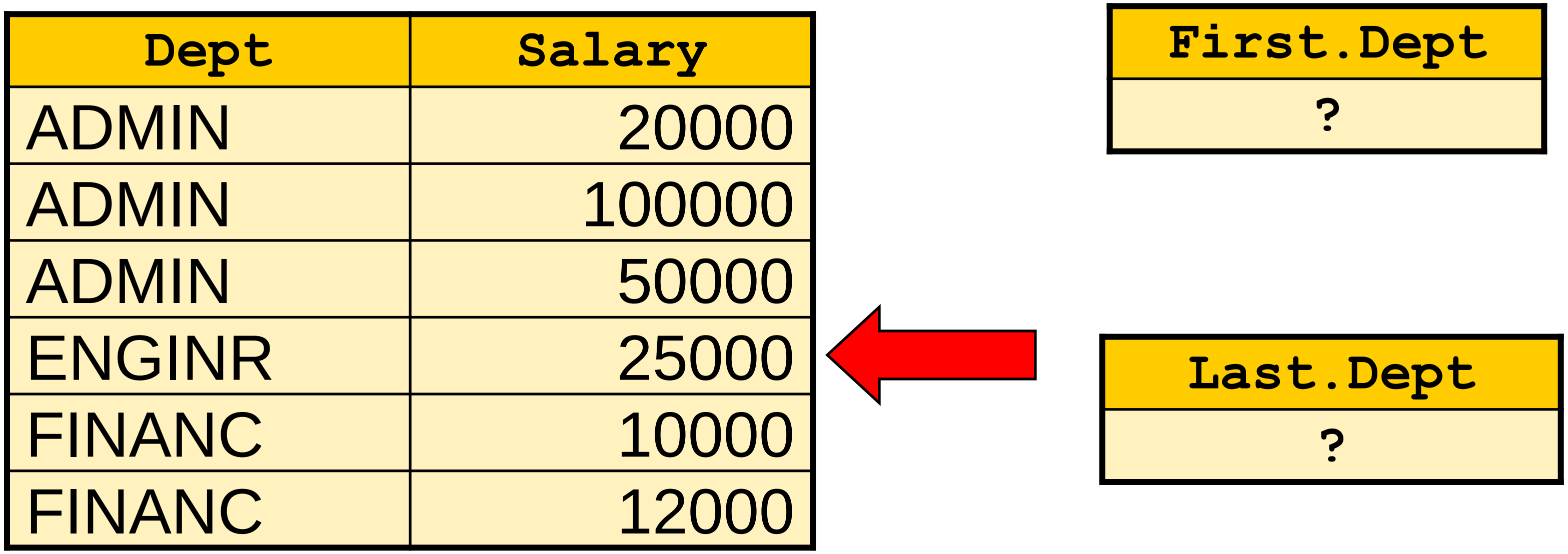
First.Dept
1

Last.Dept
0

Poll
Quiz

3.04 Quiz

What are the values for **`First.Dept`** and **`Last.Dept`** when the DATA step is processing the observation indicated by the red arrow?



Dept	Salary
ADMIN	20000
ADMIN	100000
ADMIN	50000
ENGINR	25000
FINANC	10000
FINANC	12000

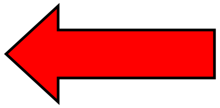
First.Dept
?

Last.Dept
?

3.04 Quiz – Correct Answer

What are the values for **`First.Dept`** and **`Last.Dept`** when the DATA step is processing the observation indicated by the red arrow?

Dept	Salary
ADMIN	20000
ADMIN	100000
ADMIN	50000
ENGINR	25000
FINANC	10000
FINANC	12000



First.Dept
1

Last.Dept
1

`First.Dept` and `Last.Dept` are both 1. This will happen when a group is composed of a single observation.

What Must Happen When?

There is a three-part process for using the DATA step to summarize grouped data.

Task 1: Set the accumulating variable to zero at the start of each BY group.

Task 2: Increment the accumulating variable with a sum statement (automatically retains).

Task 3: Output only the last observation of each BY group.

Summarizing Data by Groups

Task 1: Set the accumulating variable to zero
at the start of each BY group.

```
data deptsals(keep=Dept DeptSal);
   set SalSort;
   by Dept;
   if First.Dept then DeptSal=0;
   <additional SAS statements>
run;
```

✎ The condition is considered true when **`First.Dept`** has a value of 1.

Summarizing Data by Groups

Task 2: Increment the accumulating variable with
a sum statement (automatically retains).

```
data deptsals(keep=Dept DeptSal);
    set SalSort;
    by Dept;
    if First.Dept then DeptSal=0;
    DeptSal+Salary;
    <additional SAS statements>
run;
```

Summarizing Data by Groups

Task 3: Output only the last observation of each BY group.

Dept	Salary	DeptSal
ADMIN	20000	20000
ADMIN	100000	120000
ADMIN	50000	170000
ENGINR	25000	25000
ENGINR	20000	45000
ENGINR	23000	68000
ENGINR	27000	95000
FINANC	10000	10000
FINANC	12000	22000

Subsetting IF Statement (Review)

The subsetting IF defines a condition that the observation must meet to be further processed by the DATA step.

General form of the subsetting IF statement:

IF *expression***;**

- If the expression is true, the DATA step continues processing the current observation.
- If the expression is false, SAS returns to the top of the DATA step.

Summarizing Data by Groups

Task 3: Output only the last observation of each
BY group.

```
data deptsals(keep=Dept DeptSal);
   set SalSort;
   by Dept;
   if First.Dept then DeptSal=0;
   DeptSal+Salary;
   if Last.Dept;
run;
```

p203d04

Summarizing Data by Groups

Partial SAS Log

```
NOTE: There were 39 observations read
      from the data set WORK.SALSORT.
NOTE: The data set WORK.DEPTSALS has 5
      observations and 2 variables.
```

Summarizing Data by Groups

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

PROC PRINT Output

```
  Dept        DeptSal

ADMIN          410000
ENGINR         163000
FINANC         318000
HUMRES         181000
SALES          373000
```

Poll
Quiz

3.05 Multiple Answer Poll

What must happen in the DATA step to summarize data by groups? (Circle all that apply.)

a. Sort the input data.

b. Set the accumulating variable to zero at the start of each BY group.

c. Increment the accumulating variable.

d. Output only the last observation of each BY group.

3.05 Multiple Answer Poll – Correct Answer

What must happen in the DATA step to summarize data by groups? (Circle all that apply.)

a. Sort the input data.

[✓ b.] Set the accumulating variable to zero at the start of each BY group.

[✓ c.] Increment the accumulating variable.

[✓ d.] Output only the last observation of each BY group.

Choice a. does not apply because sorting is done with PROC SORT, not in the DATA step.

Business Scenario

Each employee listed in **orion.projsals** is assigned to a special project. A business analyst would like to see the salary totals from each department for each special project.

Partial Listing of **orion.projsals**

```
Employee_
    ID      Salary    Proj    Dept

 110004     42000     EZ      HUMRES
 110009     34000     WIN     ENGINR
 110011     27000     WIN     FINANC
 110036     20000     WIN     ENGINR
 110037     19000     EZ      ENGINR
 110048     19000     EZ      FINANC
 110077     27000     CAP1    ADMIN
 110097     20000     EZ      ADMIN
 110107     31000     EZ      ENGINR
```

Business Scenario – Desired Output

Create a new data set, **pdsals**, that shows the number of employees and salary totals from each department for each special project.

Partial Listing of **pdsals**

```
                     Dept      Num
Proj      Dept        Sal     Emps

CAP1     ADMIN      70000       2
EZ       ADMIN      83000       3
EZ       ENGINR    109000       4
EZ       FINANC    122000       3
EZ       HUMRES    178000       5
NGEN     ADMIN      37000       2
```

Sorting by Project and Department

This is similar to the previous business scenario except that now the data must be sorted by multiple BY variables.

Sort the data by **Proj** and **Dept**:

- **Proj** is the primary sort variable.
- **Dept** is the secondary sort variable.

```
proc sort data=orion.projsals
          out=projsort;
   by Proj Dept;
run;
```

p203d05

Sorting by Project and Department

```
proc print data=projsort noobs;
   var Proj Dept Salary;
run;
```

Partial PROC PRINT Output

```
Proj        Dept        Salary

CAP1       ADMIN         27000
CAP1       ADMIN         43000
EZ         ADMIN         20000
EZ         ADMIN         31000
EZ         ADMIN         32000
EZ         ENGINR        19000
```

p203d05

Multiple BY Variables

The DATA step must include both **`Proj`** and **`Dept`** in the BY statement.

```
data pdsals;
   set projsort;
   by Proj Dept;
   <additional SAS statements>
run;
```

How does the DATA step set First. and Last. values for multiple BY variables?

First. / Last. Values – 1st DATA Step Iteration

Proj	Dept
CAP1	ADMIN
CAP1	ADMIN
CAP1	ADMIN
EZ	ADMIN
EZ	ENGINR
EZ	ENGINR
NGEN	ENGINR
NGEN	SALES
NGEN	SALES
NGEN	SALES
NGEN	SALES

First.Proj
1

First.Dept
1

Last.Proj
?

Last.Dept
?

...

First. / Last. Values – 1st DATA Step Iteration

Proj	Dept
CAP1	ADMIN
CAP1	ADMIN
CAP1	ADMIN
EZ	ADMIN
EZ	ENGINR
EZ	ENGINR
NGEN	ENGINR
NGEN	SALES
NGEN	
NGEN	
NGEN	

SAS looks ahead at the next observation to determine the values for `Last.Proj` and `Last.Dept`.

First.Proj
1

First.Dept
1

Last.Proj
0

Last.Dept
0

...

First. / Last. Values – 2nd DATA Step Iteration

Proj	Dept
CAP1	ADMIN
CAP1	ADMIN
CAP1	ADMIN
EZ	ADMIN
EZ	ENGINR
EZ	ENGINR
NGEN	ENGINR
NGEN	SALES
NGEN	SALES
NGEN	SALES
NGEN	SALES

First.Proj
0

First.Dept
0

Last.Proj
0

Last.Dept
0

...

First. / Last. Values – 3rd DATA Step Iteration

Proj	Dept
CAP1	ADMIN
CAP1	ADMIN
CAP1	ADMIN
EZ	ADMIN
EZ	ENGINR
EZ	ENGINR
NGEN	ENGINR
NGEN	SALES
NGEN	SALES
NGEN	SALES
NGEN	SALES

First.Proj
0

First.Dept
0

Last.Proj
1

Last.Dept
1

...

First. / Last. Values – 4th DATA Step Iteration

Proj	Dept
CAP1	ADMIN
CAP1	ADMIN
CAP1	ADMIN
EZ	ADMIN
EZ	ENGINR
EZ	ENGINR
NGEN	ENGINR
NGEN	SALES
NGEN	SALES
NGEN	SALES
NGEN	SALES

First.Proj
1

First.Dept
1

Last.Proj
0

Last.Dept
1

...

First. / Last. Values – 5th DATA Step Iteration

Proj	Dept
CAP1	ADMIN
CAP1	ADMIN
CAP1	ADMIN
EZ	ADMIN
EZ	ENGINR
EZ	ENGINR
NGEN	ENGINR
NGEN	SALES
NGEN	SALES
NGEN	SALES
NGEN	SALES

First.Proj
0

First.Dept
1

Last.Proj
0

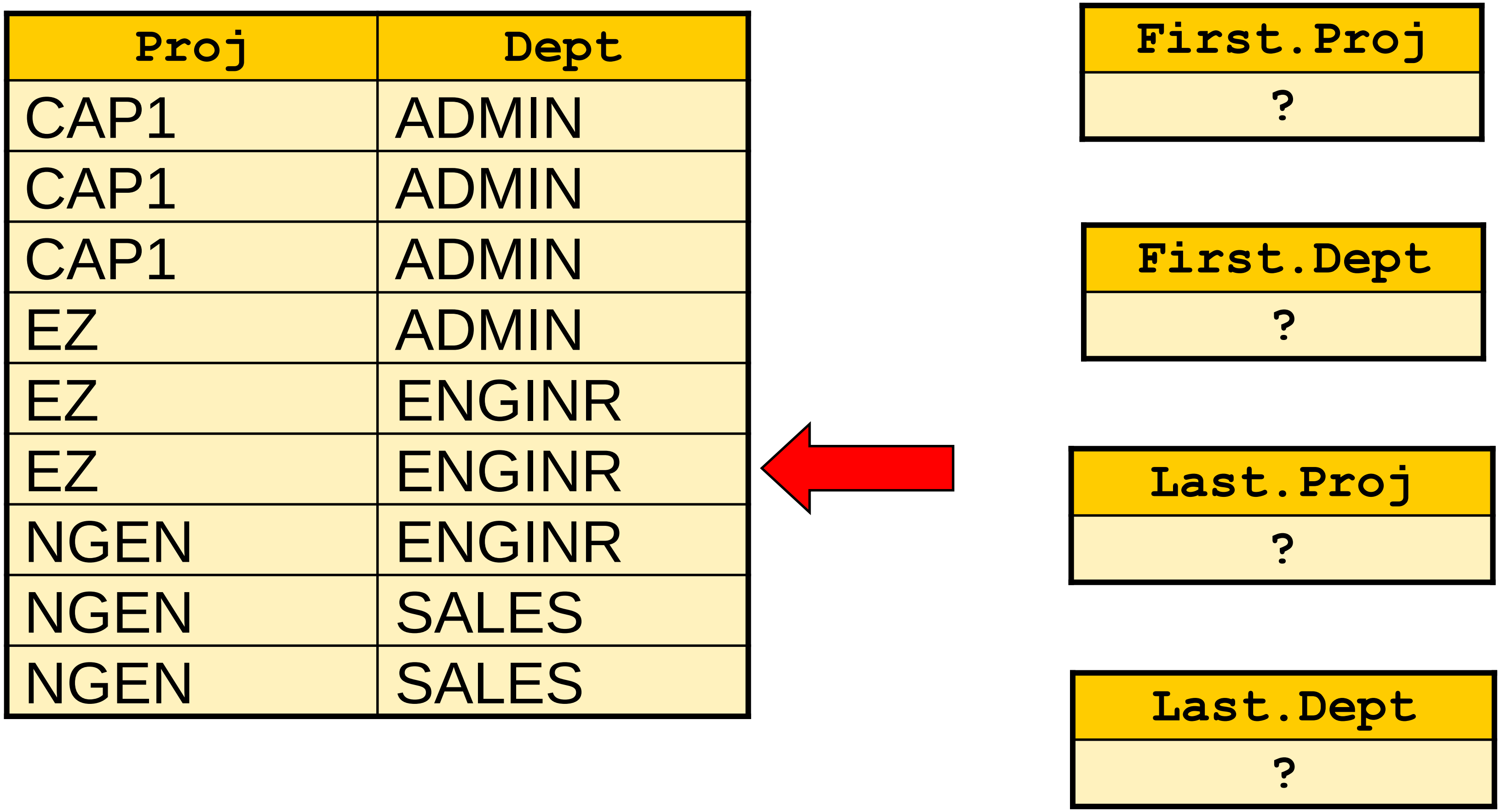
Last.Dept
0

...

Poll
Quiz

3.06 Quiz

What are the values for First. and Last. variables when the DATA step is processing the observation indicated by the red arrow?



Proj	Dept
CAP1	ADMIN
CAP1	ADMIN
CAP1	ADMIN
EZ	ADMIN
EZ	ENGINR
EZ	ENGINR
NGEN	ENGINR
NGEN	SALES
NGEN	SALES

First.Proj
?

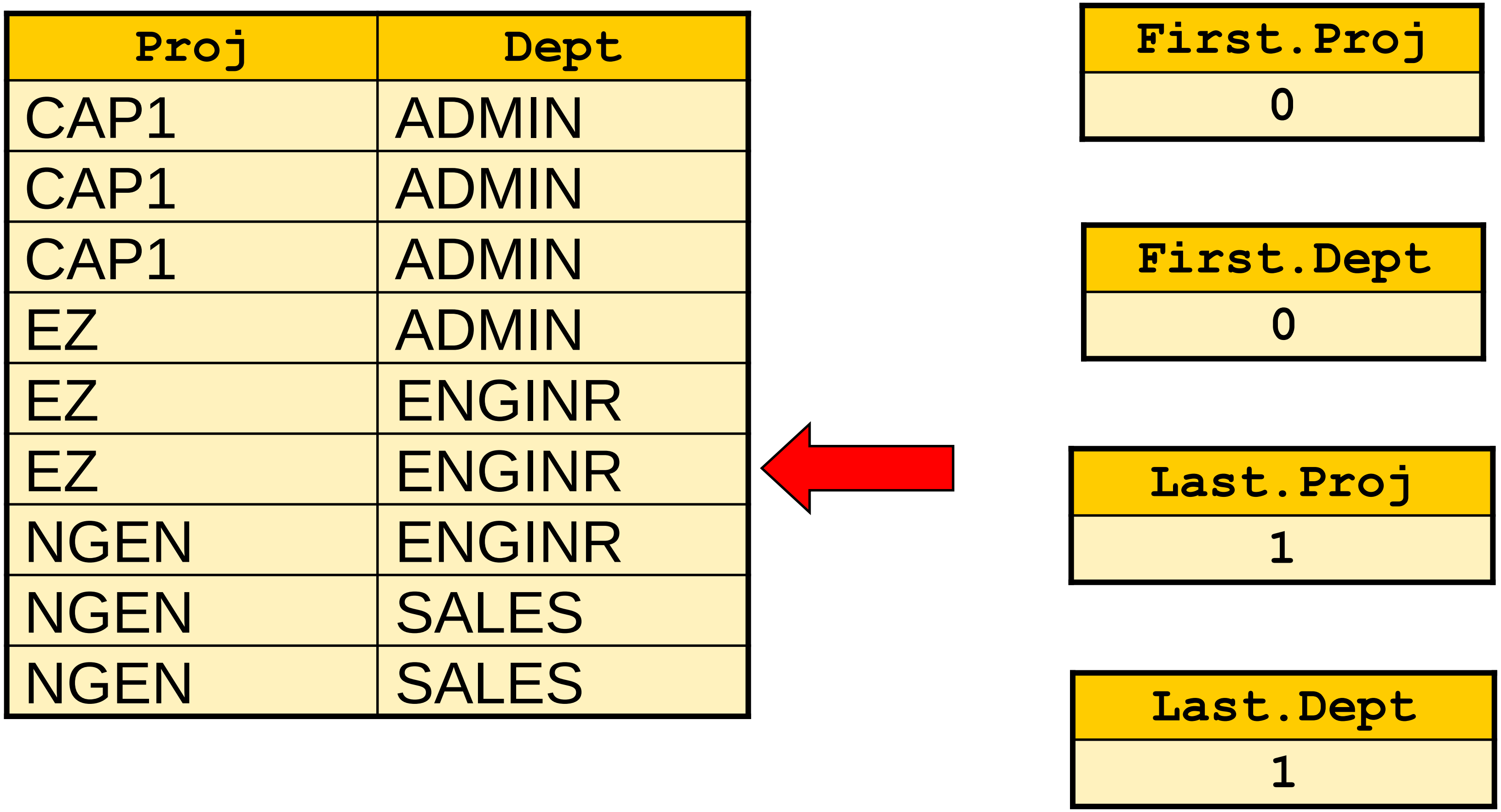
First.Dept
?

Last.Proj
?

Last.Dept
?

3.06 Quiz – Correct Answer

What are the values for First. and Last. variables when the DATA step is processing the observation indicated by the red arrow?



Proj	Dept
CAP1	ADMIN
CAP1	ADMIN
CAP1	ADMIN
EZ	ADMIN
EZ	ENGINR
EZ	ENGINR
NGEN	ENGINR
NGEN	SALES
NGEN	SALES

First.Proj
0

First.Dept
0

Last.Proj
1

Last.Dept
1

First. and Last. for Multiple BY Variables

When you use more than one variable in the BY statement, `Last.`*BY-variable*=1 for the **primary variable** forces `Last.`*BY-variable*=1 for the **secondary variable(s)**.

		First.	Last.	First.	
Proj	Dept	Proj	Proj	Dept	Last.Dept
CAP1	ADMIN	1	0	1	0
CAP1	ADMIN	0	0	0	0
CAP1	ADMIN	0	1	0	1
EZ	ADMIN	1	0	1	1
EZ	ENGINR	0	0	1	0

Change in Primary

Change in Secondary

Multiple BY Variables

Here is the complete DATA step:

```
data pdsals(keep=Proj Dept
                 DeptSal NumEmps);
   set projsort;
   by Proj Dept;
   if First.Dept then do;
      DeptSal=0;
      NumEmps=0;
   end;
   DeptSal+Salary;
   NumEmps+1;
   if Last.Dept;
run;
```

p203d05

Multiple BY Variables

Partial SAS Log

```
NOTE: There were 39 observations read
      from the data set WORK.PROJSORT.
NOTE: The data set WORK.PDSALS has 14
      observations and 4 variables.
```

Multiple BY Variables

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

Partial PROC PRINT Output

```
                        Dept       Num
Proj      Dept           Sal      Emps

CAP1     ADMIN         70000        2
EZ       ADMIN         83000        3
EZ       ENGINR       109000        4
EZ       FINANC       122000        3
```

p203d05