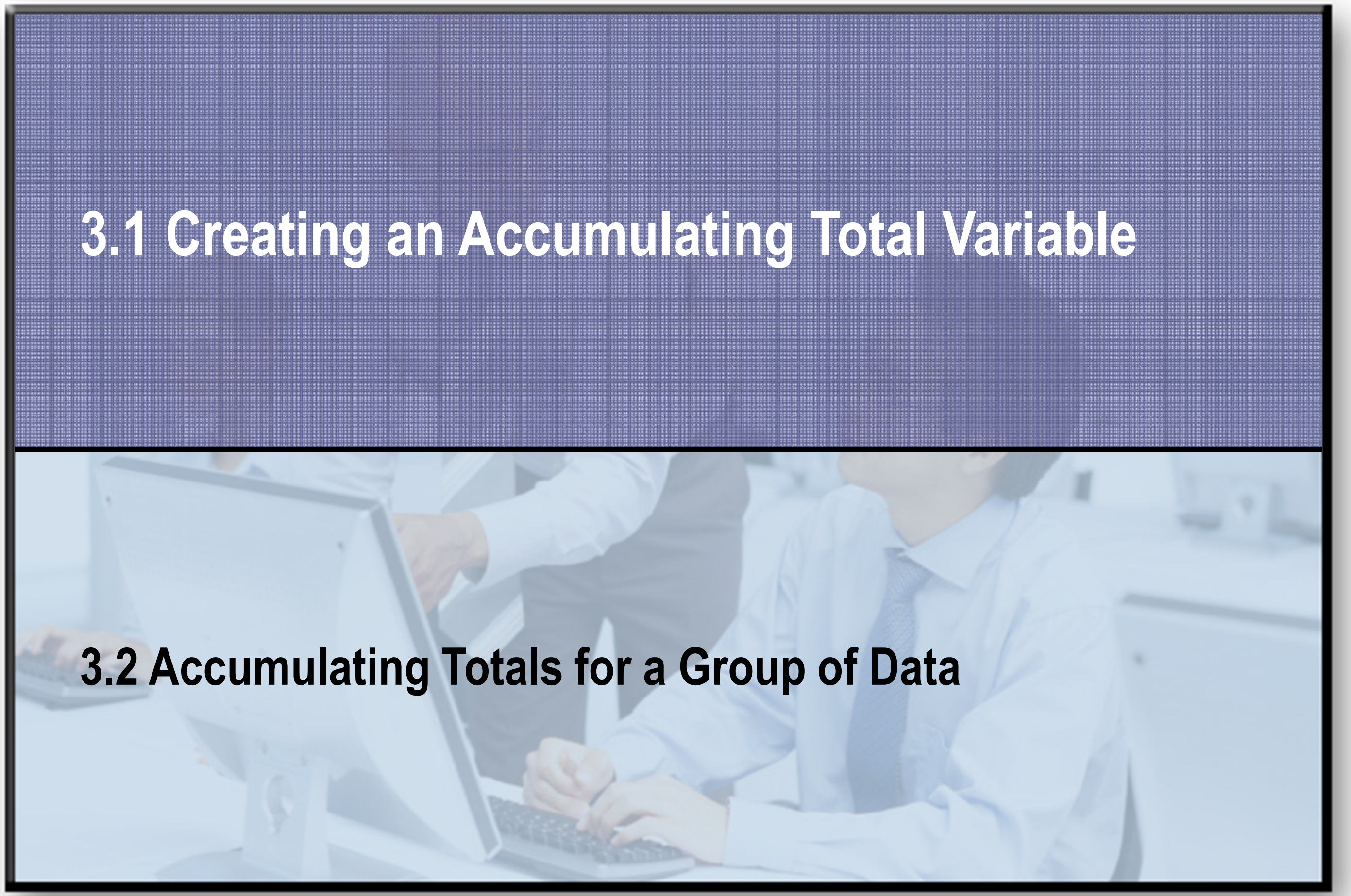
Chapter 3: Summarizing Data



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`**).

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

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.

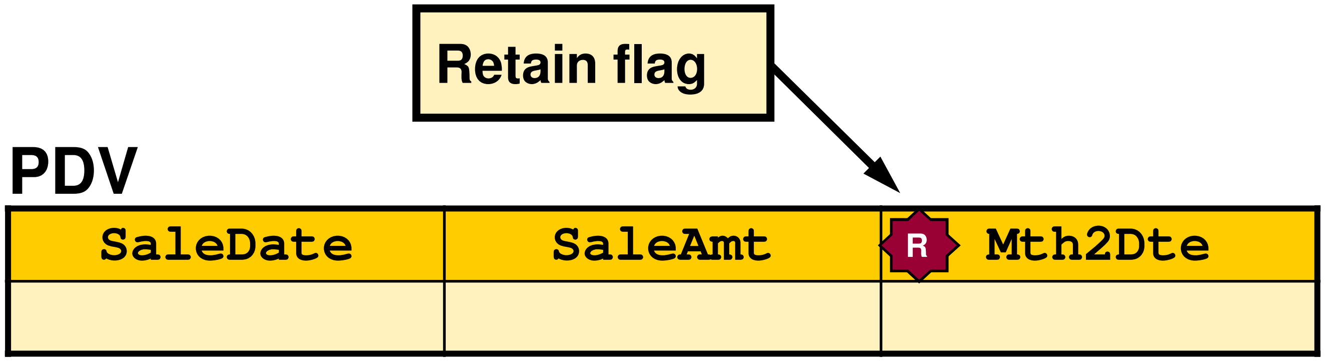
```
data mnthtot;
   set orion.aprsales;
   retain Mth2Dte 0;
   Mth2Dte=Mth2Dte+SaleAmt;
run;
```

⚠ 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;
```



...

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

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.

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;
```

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

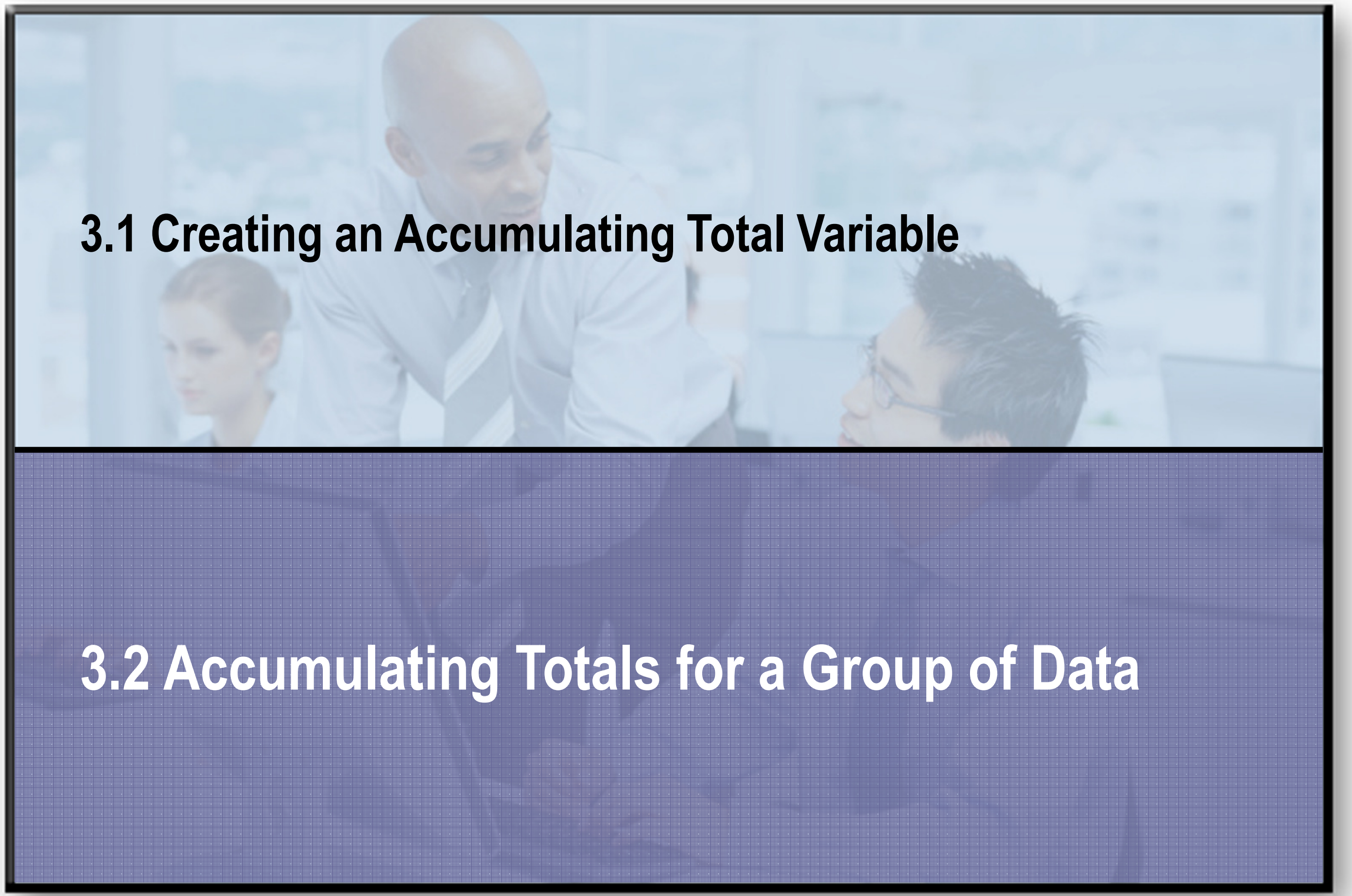
```
proc print data=mnthtot2 noobs;
   format SaleDate date9.;
run;
```

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
```

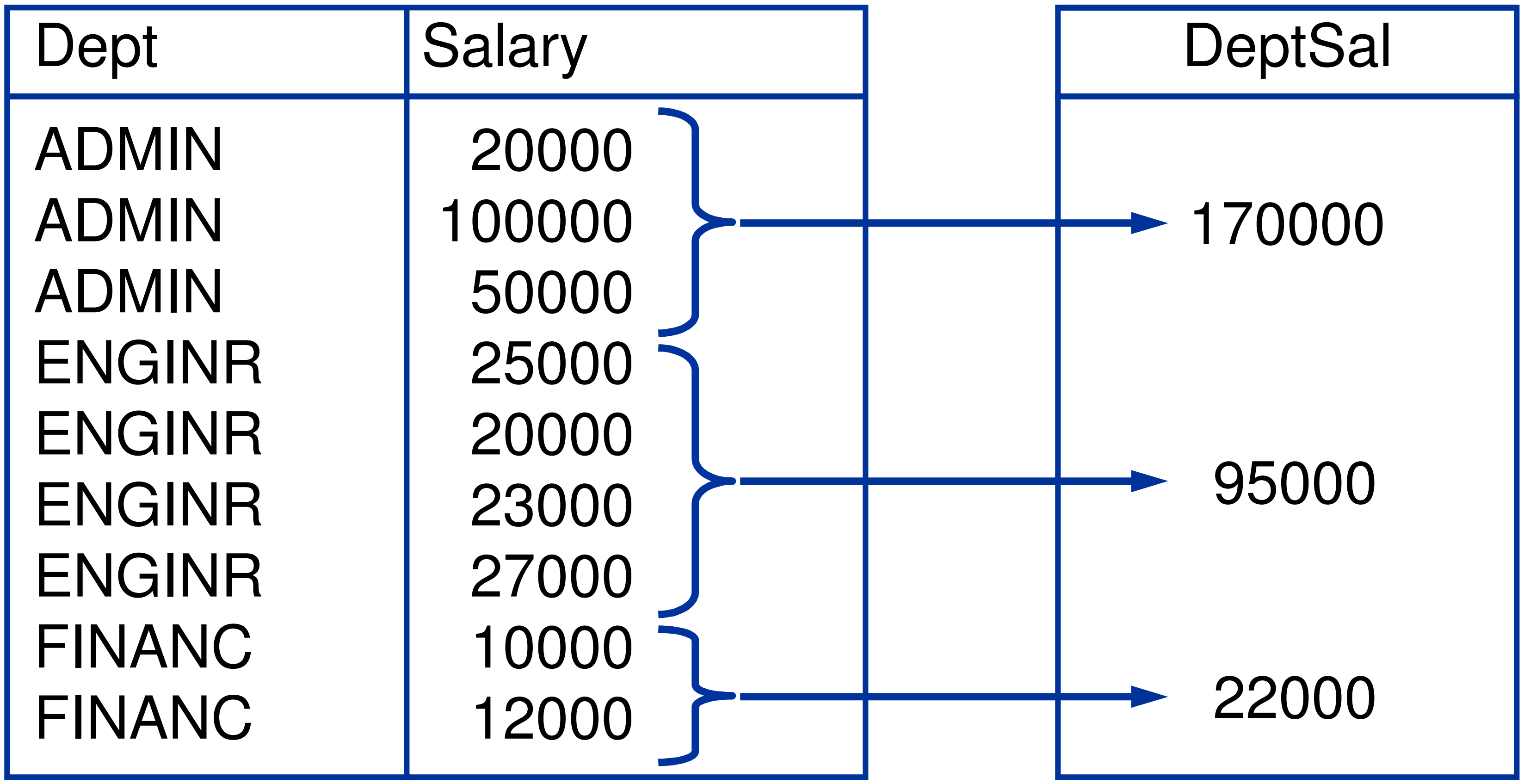
Processing Needed

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.

3.03 Multiple Choice Poll

How often do you work with data that needs to be processed by groups?

a. All the time
b. Regularly
c. Not often

The SORT Procedure (Review)

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;
```

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

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
?

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:

$$\boxed{\textbf{IF}\ \textit{expression}\textbf{;}}$$

- 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
```

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.

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	

First.Proj
1

First.Dept
1

Last.Proj
0

Last.Dept
0

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

...

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

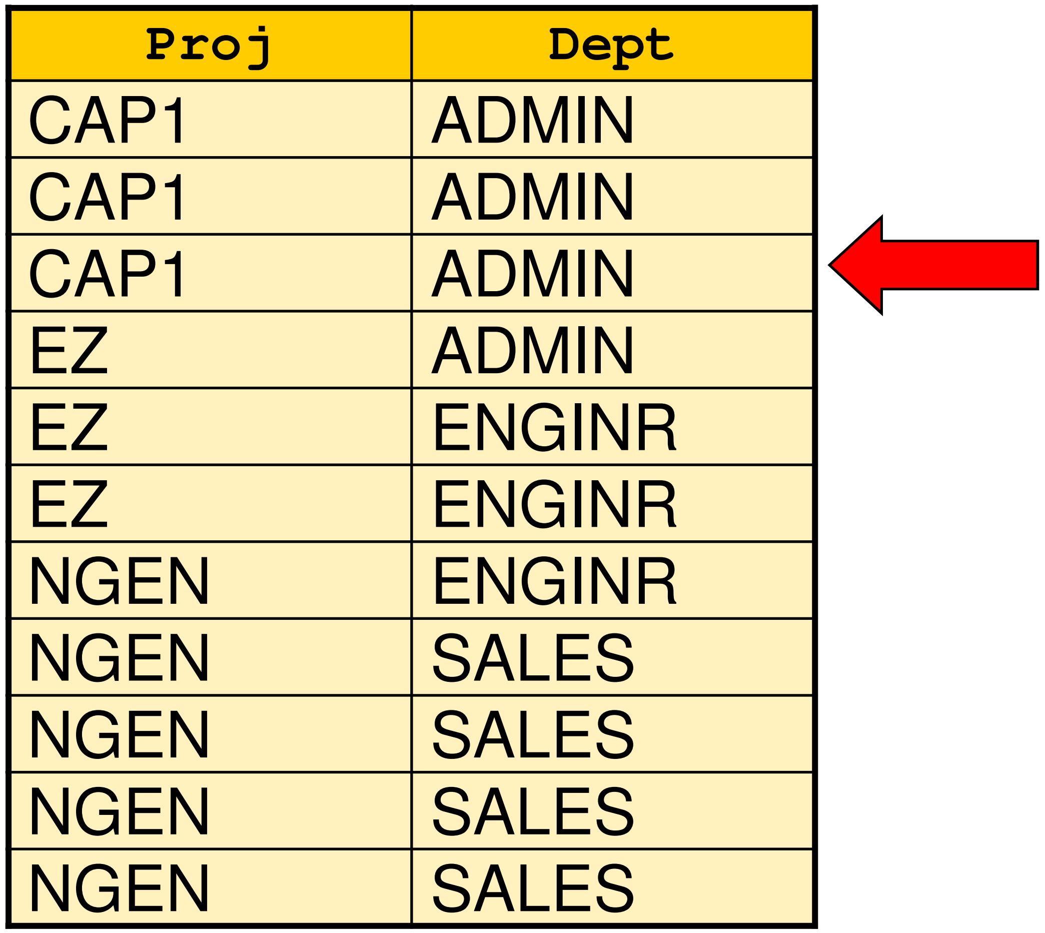
First.Dept
0

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

...

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
?

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)**.

Proj	Dept	First. Proj	Last. Proj	First. 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

Chapter Review

1. What statement prevents SAS from reinitializing the values in a variable?

2. Describe the effect on `Total_Salary` if it is used in this statement: `Total_Salary + Salary`;

3. If a data set is not sorted, can it be used as the input data set for BY-group processing?

continued...

Chapter Review

4. What statement in the DATA step enables BY-group processing?

5. In BY-group processing, two temporary variables are created for each variable listed in the BY statement. What are the names of the temporary variables?

6. What are the possible values for those temporary variables?

7. Is there ever a case where the values for the First.BY-variable and the Last.BY-variable can both be 1?

Chapter Review Answers

4. What statement in the DATA step enables BY-Group processing?

   The BY statement enables BY-group processing.

5. In BY-group processing, two temporary variables are created for each variable listed in the BY statement. What are the names of the temporary variables?

   First.BY-variable and Last.BY-variable where the BY-variable is the name of the variable used in the BY statement

continued...

Chapter Review Answers

6. What are the possible values for those temporary variables?

   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.

7. Is there ever a case where the values for the First.BY-variable and the Last.BY-variable can both be 1?

   Yes, this happens when a BY group is composed of a single observation.