

Chapter 9: Manipulating Data

9.1 Creating Variables

9.2 Creating Variables Conditionally

9.3 Subsetting Observations

Objectives

- Create SAS variables with the assignment statement in the DATA step.
- Create data values by using operators including SAS functions.
- Subset variables by using the DROP and KEEP statements.
- Examine the compilation and execution phases of the DATA step when you read a SAS data set.
- Subset variables by using the DROP= and KEEP= options. (Self-Study)

Business Scenario

A new SAS data set named **Work.comp** needs to be created by reading the **orion.sales** data set.

Work.comp must include the following new variables:

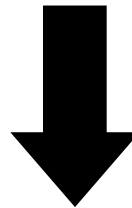
- **Bonus**, which is equal to a constant 500
- **Compensation**, which is the combination of the employee's salary and bonus
- **BonusMonth**, which is equal to the month that the employee was hired

Work.comp must not include the **Gender**, **Salary**, **Job_Title**, **Country**, **Birth_Date**, and **Hire_Date** variables from **orion.sales**.

Business Scenario

Partial `orion.sales`

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	Country	Birth_Date	Hire_Date
120102	Tom	Zhou	M	108255	Sales Manager	AU	3510	10744
120103	Wilson	Dawes	M	87975	Sales Manager	AU	-3996	5114
120121	Irenie	Elvish	F	26600	Sales Rep. II	AU	-5630	5114



Partial `Work.comp`

Employee_ID	First_Name	Last_Name	Bonus	Compensation	Bonus Month
120102	Tom	Zhou	500	108755	6
120103	Wilson	Dawes	500	88475	1
120121	Irenie	Elvish	500	27100	1

Assignment Statements (Review)

Assignment statements are used in the DATA step to update existing variables or create new variables.

```
DATA output-SAS-data-set;  
    SET input-SAS-data-set;  
    variable = expression;  
RUN;
```

```
DATA output-SAS-data-set;  
    INFILE 'raw-data-file-name';  
    INPUT specifications;  
    variable = expression;  
RUN;
```

Assignment Statements (Review)

The *assignment statement* evaluates an expression and assigns the resulting value to a variable.

General form of the assignment statement:

variable = expression;

- *variable* names an existing or new variable.
- *expression* is a sequence of operands and operators that form a set of instructions that produce a value.

Operands (Review)

Operands are constants (character, numeric, or date) and variables (character or numeric).

Examples:

`Bonus = 500;`

← numeric constant

`Gender = 'M';`

← character constant

`Hire_Date = '01APR2008'd;`

← date constant

`NewSalary = 1.1 * Salary;`

↑
variable

Operators (Review)

Operators are symbols that represent an arithmetic calculation and SAS functions.

Examples:

```
Revenue = Quantity * Price;
```

```
NewCountry = upcase(Country);
```


Arithmetic Operators

Arithmetic operators indicate that an arithmetic calculation is performed.

Symbol	Definition	Priority
**	exponentiation	I
-	negative prefix	I
*	multiplication	II
/	division	II
+	addition	III
-	subtraction	III



If a missing value is an operand for an arithmetic operator, the result is a missing value.

9.01 Quiz

What is the result of the assignment statement?

- a. . (missing)
- b. 0
- c. 7
- d. 9

```
num = 4 + 10 / 2;
```

9.02 Quiz

What is the result of the assignment statement given the values of **var1** and **var2**?

- a. . (missing)
- b. 0
- c. 5
- d. 10

```
num = var1 + var2 / 2;
```

var1	var2
.	10

SAS Functions (Review)

A SAS *function* is a routine that returns a value that is determined from specified arguments.

Some SAS functions manipulate character values, compute descriptive statistics, or manipulate SAS date values.

General form of a SAS function:

function-name(argument1, argument2, ...)

- Depending on the function, zero, one, or many arguments are used.
- Arguments are separated with commas.

Descriptive Statistics Function

The *SUM function* returns the sum of the arguments.

General form of the SUM function:

SUM(*argument1, argument2, ...*)

- The arguments must be numeric values.
- Missing values are ignored by some of the descriptive statistics functions.

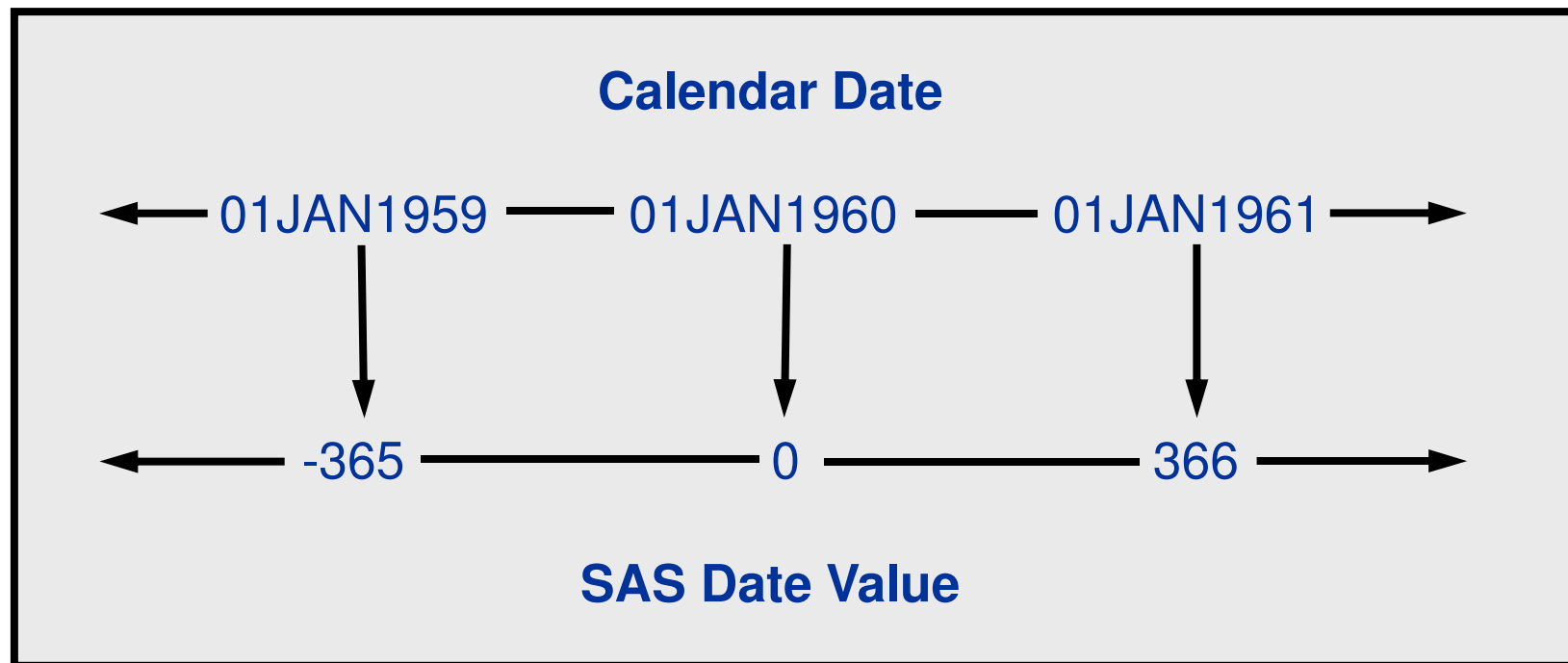
Example:

Compensation=sum (Salary, Bonus) ;

Date Functions

SAS date functions can be used to

- extract information from SAS date values
- create SAS date values.



Date Functions – Extracting Information

YEAR(<i>SAS-date</i>)	extracts the year from a SAS date and returns a four-digit value for year.
QTR(<i>SAS-date</i>)	extracts the quarter from a SAS date and returns a number from 1 to 4.
MONTH(<i>SAS-date</i>)	extracts the month from a SAS date and returns a number from 1 to 12.
DAY(<i>SAS-date</i>)	extracts the day of the month from a SAS date and returns a number from 1 to 31.
WEEKDAY(<i>SAS-date</i>)	extracts the day of the week from a SAS date and returns a number from 1 to 7, where 1 represents Sunday, and so on.

Example:

```
BonusMonth=month (Hire_Date) ;
```

Date Functions – Creating SAS Dates

TODAY()	returns the current date as a SAS date value.
MDY(<i>month,day,year</i>)	returns a SAS date value from numeric month, day, and year values.

Example:

```
AnnivBonus=mdy (month (Hire_Date) , 15 , 2008) ;
```


Business Scenario

Create **Bonus**, **Compensation**, and **BonusMonth**.

```
data work.comp;  
  set orion.sales;  
  Bonus=500;  
  Compensation=sum(Salary,Bonus);  
  BonusMonth=month(Hire_Date);  
run;
```

```
1700 data work.comp;  
1701   set orion.sales;  
1702   Bonus=500;  
1703   Compensation=sum(Salary,Bonus);  
1704   BonusMonth=month(Hire_Date);  
1705 run;
```

orion.sales
has 9 variables.

NOTE: There were 165 observations read from the data set ORION.SALES.
NOTE: The data set WORK.COMP has 165 observations and 12 variables.

9.03 Quiz

What statement needs to be added to the DATA step to eliminate six of the 12 variables?

The DROP and KEEP Statements (Review)

The *DROP statement* specifies the names of the variables to omit from the output data set(s).

DROP *variable-list* ;

The *KEEP statement* specifies the names of the variable to write to the output data set(s).

KEEP *variable-list* ;

The *variable-list* specifies the variables to drop or keep, respectively, in the output data set.

Business Scenario

Drop Gender, Salary, Job_Title, Country, Birth_Date, and Hire_Date.

```
data work.comp;  
    set orion.sales;  
    Bonus=500;  
    Compensation=sum(Salary, Bonus);  
    BonusMonth=month(Hire_Date);  
    drop Gender Salary Job_Title  
        Country Birth_Date Hire_Date;  
run;
```

Partial SAS Log

```
NOTE: There were 165 observations read from the data set ORION.SALES.  
NOTE: The data set WORK.COMP has 165 observations and 6 variables.
```

Business Scenario

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

Partial PROC PRINT Output

Obs	Employee_ID	First_ Name	Last_Name	Bonus	Compensation	Bonus Month
1	120102	Tom	Zhou	500	108755	6
2	120103	Wilson	Dawes	500	88475	1
3	120121	Irenie	Elvish	500	27100	1
4	120122	Christina	Ngan	500	27975	7
5	120123	Kimiko	Hotstone	500	26690	10
6	120124	Lucian	Daymond	500	26980	3
7	120125	Fong	Hofmeister	500	32540	3
8	120126	Satyakam	Denny	500	27280	8
9	120127	Sharryn	Clarkson	500	28600	11
10	120128	Monica	Kletschkus	500	31390	11

Setup for the Poll

- Submit program **p109a01**.
- Verify the results.

```
data work.comp;  
  set orion.sales;  
  drop Gender Salary Job_Title  
        Country Birth_Date Hire_Date;  
  Bonus=500;  
  Compensation=sum(Salary, Bonus);  
  BonusMonth=month(Hire_Date);  
run;
```

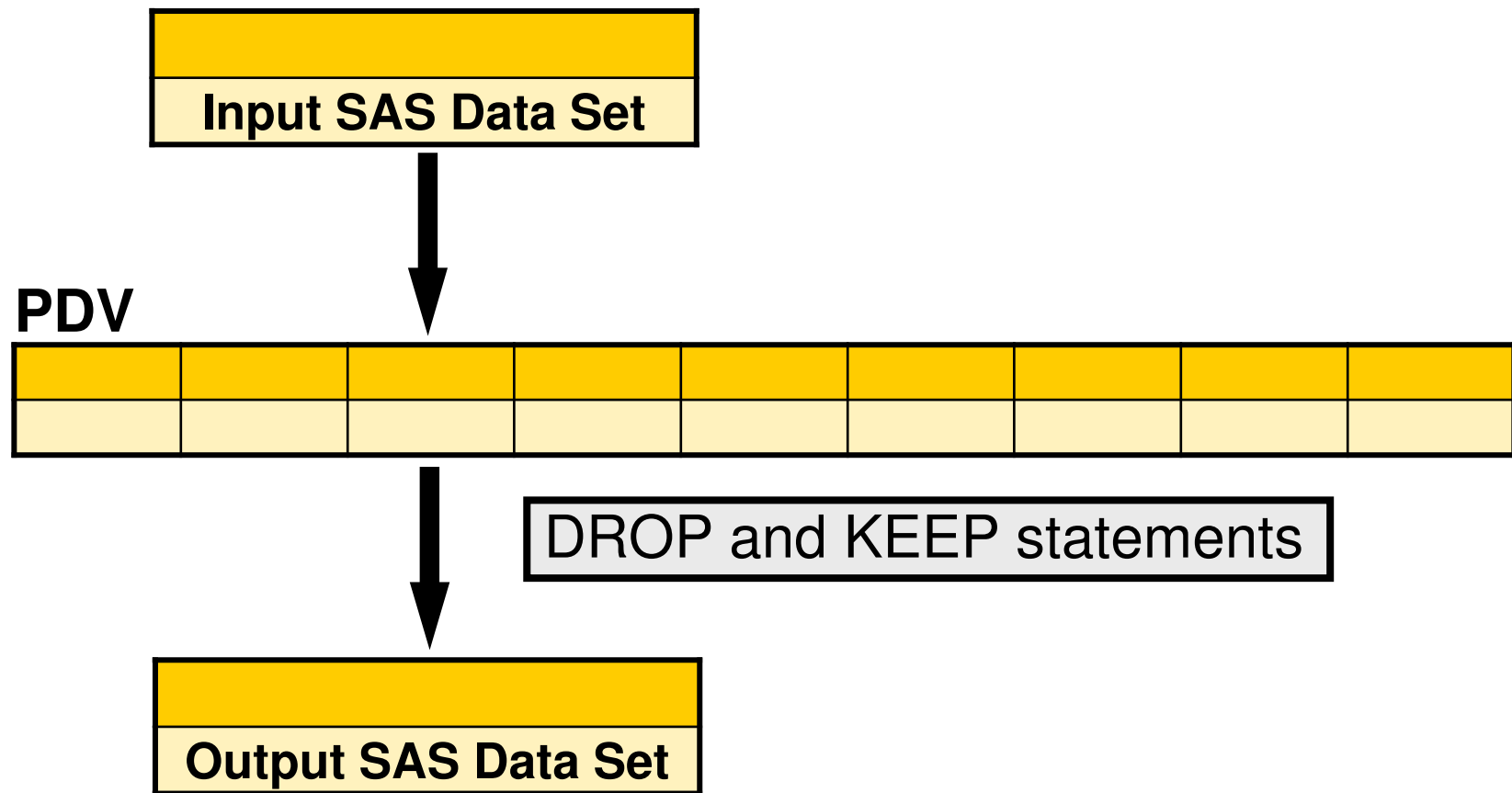
9.04 Poll

Are the correct results produced when the DROP statement is placed after the SET statement?

- ☐ Yes
- ☐ No

Processing the DROP and KEEP Statements

The DROP and KEEP statements select variables **after** they are brought into the program data vector.



Compilation

```
data work.comp;  
  set orion.sales;  
  drop Gender Salary Job_Title  
        Country Birth_Date Hire_Date;  
  Bonus=500;  
  Compensation=sum(Salary,Bonus);  
  BonusMonth=month(Hire_Date);  
run;
```

Compilation

```
data work.comp;  
  set orion.sales;  
  drop Gender Salary Job_Title  
        Country Birth_Date Hire_Date;  
  Bonus=500;  
  Compensation=sum(Salary,Bonus);  
  BonusMonth=month(Hire_Date);  
run;
```

PDV

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title
N 8	\$ 12	\$ 18	\$ 1	N 8	\$ 25

Country	Birth_Date	Hire_Date
\$ 2	N 8	N 8

Compilation

```
data work.comp;  
  set orion.sales;  
  drop Gender Salary Job_Title  
        Country Birth_Date Hire_Date;  
  Bonus=500;  
  Compensation=sum(Salary,Bonus);  
  BonusMonth=month(Hire_Date);  
run;
```

PDV

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title
N 8	\$ 12	\$ 18	\$ 1	N 8	\$ 25

Country	Birth_Date	Hire_Date	Bonus
\$ 2	N 8	N 8	N 8

Compilation

```
data work.comp;  
  set orion.sales;  
  drop Gender Salary Job_Title  
        Country Birth_Date Hire_Date;  
  Bonus=500;  
  Compensation=sum(Salary,Bonus);  
  BonusMonth=month(Hire_Date);  
run;
```

PDV

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title
N 8	\$ 12	\$ 18	\$ 1	N 8	\$ 25

Country	Birth_Date	Hire_Date	Bonus	Compensation
\$ 2	N 8	N 8	N 8	N 8

Compilation

```
data work.comp;  
  set orion.sales;  
  drop Gender Salary Job_Title  
        Country Birth_Date Hire_Date;  
  Bonus=500;  
  Compensation=sum(Salary,Bonus);  
  BonusMonth=month(Hire_Date);  
run;
```

PDV

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title
N 8	\$ 12	\$ 18	\$ 1	N 8	\$ 25

Country	Birth_Date	Hire_Date	Bonus	Compensation	BonusMonth
\$ 2	N 8	N 8	N 8	N 8	N 8

Compilation

```
data work.comp;  
  set orion.sales;  
  drop Gender Salary Job_Title  
      Country Birth_Date Hire_Date;  
  Bonus=500;  
  Compensation=sum(Salary, Bonus);  
  BonusMonth=month(Hire_Date);  
run;
```

PDV

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title
N 8	\$ 12	\$ 18	\$ 1	N 8	\$ 25

Country	Birth_Date	Hire_Date	Bonus	Compensation	BonusMonth
\$ 2	N 8	N 8	N 8	N 8	N 8

Compilation

```
data work.comp;  
  set orion.sales;  
  drop Gender Salary Job_Title  
        Country Birth_Date Hire_Date;  
  Bonus=500;  
  Compensation=sum(Salary,Bonus);  
  BonusMonth=month(Hire_Date);  
run;
```

PDV

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title
N 8	\$ 12	\$ 18	\$ 1	N 8	\$ 25

Country	Birth_Date	Hire_Date	Bonus	Compensation	BonusMonth
\$ 2	N 8	N 8	N 8	N 8	N 8

Descriptor Portion Work . comp

Employee_ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
N 8	\$ 12	\$ 18	N 8	N 8	N 8

Execution

Partial orion.sales

Employee_ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

```
data work.comp;
```

```
set orion.sales;
drop GenComp;
Comp =
    Hire_Date;
Bonus=500;
Compensation=sum(Salary,Bonus);
BonusMonth=month(Hire_Date);
run;
```

Initialize PDV

PDV

Employee_ID	Sender	Hire_Date	Bonus	Compensation	BonusMonth
.

Work.comp

Employee_ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
-------------	------------	-----------	-------	--------------	------------

Execution

Partial orion.sales

Employee _ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

...

```
data work.comp;
  set orion.sales;
  drop Gender Salary Job_Title
        Country Birth_Date
        Hire_Date;
  Bonus=500;
  Compensation=sum(Salary, Bonus);
  BonusMonth=month(Hire_Date);
run;
```

PDV

Employee _ID	Gender	Hire_Date	Bonus	Compensation	BonusMonth
120102	M	10744	.	.	.

Work.comp

Employee _ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
-----------------	------------	-----------	-------	--------------	------------

Execution

Partial orion.sales

Employee_ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

...

```
data work.comp;
  set orion.sales;
  drop Gender Salary Job_Title
        Country Birth_Date
        Hire_Date;
  Bonus=500;
  Compensation=sum(Salary, Bonus);
  BonusMonth=month(Hire_Date);
run;
```

PDV

Employee_ID	Gender	Hire_Date	Bonus	Compensation	BonusMonth
120102	M	10744	500	.	.

Work.comp

Employee_ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
-------------	------------	-----------	-------	--------------	------------

Execution

Partial orion.sales

Employee _ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

...

```
data work.comp;
  set orion.sales;
  drop Gender Salary Job_Title
        Country Birth_Date
        Hire_Date;
  Bonus=500;
  Compensation=sum(Salary,Bonus);
  BonusMonth=month(Hire_Date);
run;
```

PDV

Employee _ID	Gender	Hire_Date	Bonus	Compensation	BonusMonth
120102	M	10744	500	108755	.

Work.comp

Employee _ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
-----------------	------------	-----------	-------	--------------	------------

Execution

Partial orion.sales

Employee _ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

...

```
data work.comp;
  set orion.sales;
  drop Gender Salary Job_Title
        Country Birth_Date
        Hire_Date;
  Bonus=500;
  Compensation=sum(Salary, Bonus);
  BonusMonth=month(Hire_Date);
run;
```

PDV

Employee _ID	Gender	Hire_Date	Bonus	Compensation	BonusMonth
120102	M	10744	500	108755	6

Work.comp

Employee _ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
-----------------	------------	-----------	-------	--------------	------------

Execution

Partial orion.sales

Employee_ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

...

```
data work.comp;
  set orion.sales;
  drop Gender Salary Job_Title
        Country Birth_Date
        Hire_Date;
  Bonus=500;
  Compensation=sum(Salary, Bonus);
  BonusMonth=6;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

Employee_ID	Gender	Hire_Date	Bonus	Compensation	BonusMonth
120102	M	10744	500	108755	6

Work.comp

Employee_ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
120102	Tom	Zhou	500	108755	6

Execution

Partial orion.sales

Employee_ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

...

```
data work.comp;
```

```
set orion.sales;
```

```
drop GenComp; title
```

```
Comp =  
      Hire_Date;  
Bonus=500;
```

```
Bonus=500;
```

```
Compensation=sum(Salary, Bonus);
```

```
BonusMonth=month(Hire_Date);
```

```
run;
```

Reinitialize PDV

PDV

Employee_ID	Sender	Hire_Date	Bonus	Compensation	BonusMonth
120102	M	10744	.	.	.

Work.comp

Employee_ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
120102	Tom	Zhou	500	108755	6

Execution

Partial orion.sales

Employee_ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

...

```
data work.comp;
  set orion.sales;
  drop Gender Salary Job_Title
        Country Birth_Date
        Hire_Date;
  Bonus=500;
  Compensation=sum(Salary, Bonus);
  BonusMonth=month(Hire_Date);
run;
```

PDV

Employee_ID	Gender	Hire_Date	Bonus	Compensation	BonusMonth
120103	M	5114	.	.	.

Work.comp

Employee_ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
120102	Tom	Zhou	500	108755	6

Execution

Partial orion.sales

Employee_ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

...

```
data work.comp;
  set orion.sales;
  drop Gender Salary Job_Title
        Country Birth_Date
        Hire_Date;
  Bonus=500;
  Compensation=sum(Salary, Bonus);
  BonusMonth=month(Hire_Date);
run;
```

PDV

Employee_ID	Gender	Hire_Date	Bonus	Compensation	BonusMonth
120103	M	5114	500	.	.

Work.comp

Employee_ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
120102	Tom	Zhou	500	108755	6

Execution

Partial orion.sales

Employee_ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

...

```
data work.comp;
  set orion.sales;
  drop Gender Salary Job_Title
        Country Birth_Date
        Hire_Date;
  Bonus=500;
  Compensation=sum(Salary,Bonus);
  BonusMonth=month(Hire_Date);
run;
```

PDV

Employee_ID	Gender	Hire_Date	Bonus	Compensation	BonusMonth
120103	M	5114	500	88475	.

Work.comp

Employee_ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
120102	Tom	Zhou	500	108755	6

Execution

Partial orion.sales

Employee_ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

...

```
data work.comp;
  set orion.sales;
  drop Gender Salary Job_Title
        Country Birth_Date
        Hire_Date;
  Bonus=500;
  Compensation=sum(Salary, Bonus);
  BonusMonth=month(Hire_Date);
run;
```

PDV

Employee_ID	Gender	Hire_Date	Bonus	Compensation	BonusMonth
120103	M	5114	500	88475	1

Work.comp

Employee_ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
120102	Tom	Zhou	500	108755	6

Execution

Partial orion.sales

Employee_ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

...

```
data work.comp;
  set orion.sales;
  drop Gender Salary Job_Title
        Country Birth_Date
        Hire_Date;
  Bonus=500;
  Compensation=sum(Salary, Bonus);
  BonusMonth=1;
run;
```

Implicit OUTPUT;
Implicit RETURN;

PDV

Employee_ID	Gender	Hire_Date	Bonus	Compensation	BonusMonth
120103	M	5114	500	88475	1

Work.comp

Employee_ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
120102	Tom	Zhou	500	108755	6
120103	Wilson	Dawes	500	88475	1

Execution

Partial orion.sales

Employee_ID	Hire_Date
120102	10744
120103	5114
120121	5114
120122	6756

...

data work_comp:

Continue until EOF

```

Job_Title
Country Birth_Date
Hire_Date;
Bonus=500;
Compensation=sum(Salary, Bonus);
BonusMonth=month(Hire_Date);
run;

```

PDV

Employee_ID	Sender	Hire_Date	Bonus	Compensation	BonusMonth
120103	M	5114	500	88475	1

Work.comp

Employee_ID	First_Name	Last_Name	Bonus	Compensation	BonusMonth
120102	Tom	Zhou	500	108755	6
120103	Wilson	Dawes	500	88475	1

DROP= and KEEP= Options (Self-Study)

Alternatives to the DROP and KEEP statements are the DROP= and KEEP= data set options placed in the DATA statement.

- The *DROP= data set option* in the DATA statement excludes the variables for writing to the output data set.

```
DATA output-SAS-data-set (DROP = variable-list) ;
```

- The *KEEP= data set option* in the DATA statement specifies the variables for writing to the output data set.

```
DATA output-SAS-data-set (KEEP = variable-list) ;
```

DROP= and KEEP= Options (Self-Study)

The DROP= and KEEP= data set options can also be placed in the SET statement to control which variables are read from the input data set.

- The *DROP= data set option* in the SET statement excludes the variables for processing in the PDV.

```
SET input-SAS-data-set (DROP = variable-list) ;
```

- The *KEEP= data set option* in the SET statement specifies the variables for processing in the PDV.

```
SET input-SAS-data-set (KEEP = variable-list) ;
```

DROP= and KEEP= Options (Self-Study)

```
data work.comp(drop=Salary Hire_Date);  
  set orion.sales(keep=Employee_ID First_Name  
                  Last_Name Salary Hire_Date);  
  
  Bonus=500;  
  Compensation=sum(Salary, Bonus);  
  BonusMonth=month(Hire_Date);  
run;
```

orion.sales

Employee_ ID	First_ Name	Last_ Name	Gender	Salary	Job_ Title	Country	Birth_ Date	Hire_ Date
-----------------	----------------	---------------	--------	--------	------------	---------	----------------	---------------



PDV

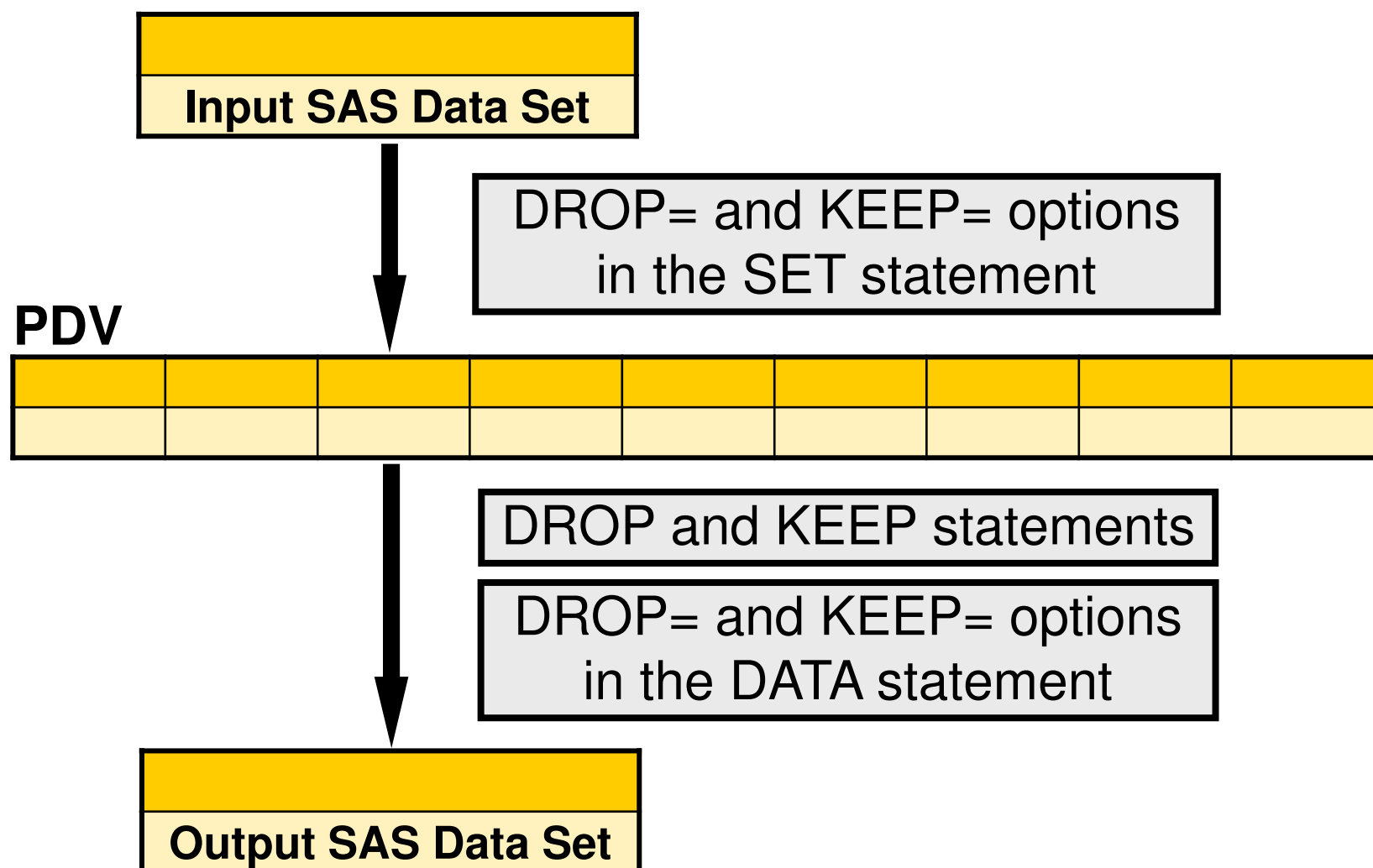
Employee_ ID	First_ Name	Last_ Name	Salary	Hire_ Date	Bonus	Compensation	BonusMonth
-----------------	----------------	---------------	--------	---------------	-------	--------------	------------



Work.comp

Employee_ ID	First_ Name	Last_ Name	Bonus	Compensation	BonusMonth
-----------------	----------------	---------------	-------	--------------	------------

DROP= and KEEP= Options (Self-Study)




Chapter 9: Manipulating Data



9.1 Creating Variables

9.2 Creating Variables Conditionally



9.3 Subsetting Observations

Objectives

- Execute statements conditionally by using IF-THEN and IF-THEN DO statements.
- Give alternate actions if the previous THEN clause is not executed by using the ELSE statement.
- Control the length of character variables by using the LENGTH statement.

Business Scenario

A new SAS data set named **Work.bonus** needs to be created by reading the **orion.sales** data set.

Work.bonus must include a new variable named **Bonus** that is equal to

- 500 for United States employees
- 300 for Australian employees.

IF-THEN Statements (Review)

The *IF-THEN statement* executes a SAS statement for observations that meet specific conditions.

General form of the IF-THEN statement:

IF *expression* **THEN** *statement*;

- *expression* is a sequence of operands and operators that form a set of instructions that define a condition for selecting observations.
- *statement* is any executable statement such as the assignment statement.

IF-THEN/ELSE Statements (Review)

The optional *ELSE statement* gives an alternate action if the previous THEN clause is not executed.

General form of the IF-THEN/ELSE statements:

IF *expression* **THEN** *statement*;
ELSE IF *expression* **THEN** *statement*;

- Using IF-THEN statements **without** the ELSE statement causes SAS to evaluate all IF-THEN statements.
- Using IF-THEN statements **with** the ELSE statement causes SAS to execute IF-THEN statements until it encounters the first true statement.

Business Scenario

Create the new variable **Bonus**.

```
data work.bonus;  
    set orion.sales;  
    if Country='US' then Bonus=500;  
    else if Country='AU' then Bonus=300;  
run;
```

```
1819 data work.bonus;  
1820     set orion.sales;  
1821     if Country='US' then Bonus=500;  
1822     else if Country='AU' then Bonus=300;  
1823 run;
```

NOTE: There were 165 observations read from the data set ORION.SALES.

NOTE: The data set WORK.BONUS has 165 observations and 10 variables

Business Scenario

```
proc print data=work.bonus;  
    var First_Name Last_Name Country Bonus;  
run;
```

Partial PROC PRINT Output

Obs	First_Name	Last_Name	Country	Bonus
60	Billy	Plested	AU	300
61	Matsuoka	Wills	AU	300
62	Vino	George	AU	300
63	Meera	Body	AU	300
64	Harry	Highpoint	US	500
65	Julienne	Magolan	US	500
66	Scott	Desanctis	US	500
67	Cherda	Ridley	US	500
68	Priscilla	Farren	US	500
69	Robert	Stevens	US	500

9.05 Quiz

Why are some of the **Bonus** values missing in the PROC PRINT output for **orion.nonsales**?

- Submit program **p109a02**.
- Review the results.

ELSE Statements

The conditional clause does not have to be in an ELSE statement.

For example:

```
data work.bonus;  
  set orion.sales;  
  if Country='US' then Bonus=500;  
  else Bonus=300;  
run;
```



All observations not equal to US get a bonus of 300.

Business Scenario

A new SAS data set named **Work.bonus** needs to be created by reading the **orion.sales** data set.

Work.bonus must include a new variable named **Bonus** that is equal to

- 500 for United States employees
- 300 for Australian employees.

Work.bonus must include another new variable named **Freq** that is equal to

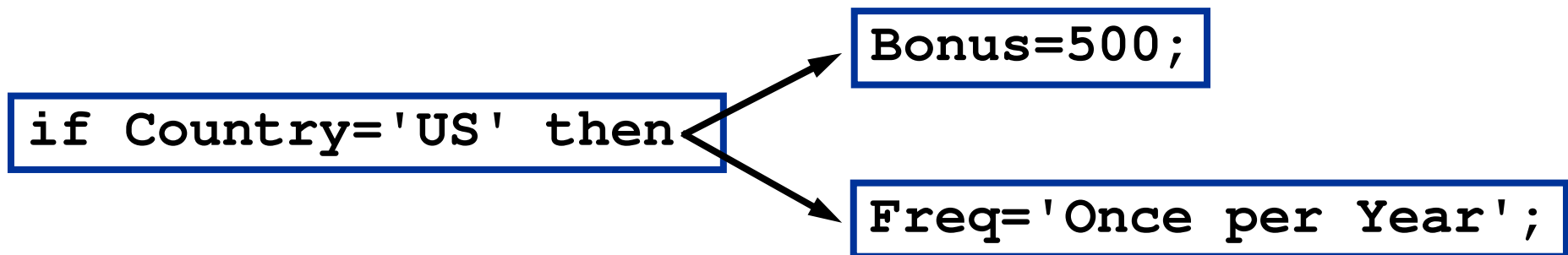
- **Once a Year** for United States employees
- **Twice a Year** for Australian employees.

IF-THEN/ELSE Statements

Only **one** executable statement is allowed in IF-THEN/ELSE statements.

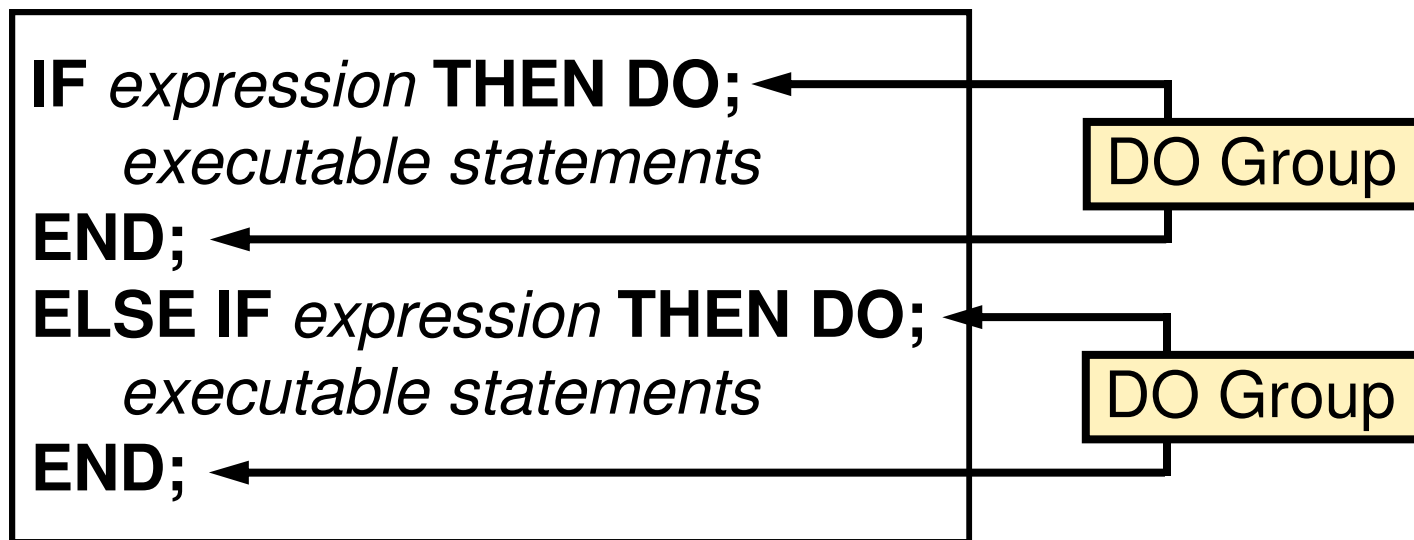
```
IF expression THEN statement;  
ELSE IF expression THEN statement;  
ELSE statement;
```

For the given business scenario, two statements need to be executed per each true expression.



IF-THEN DO/ELSE DO Statements

Multiple executable statements are allowed in IF-THEN DO/ELSE DO statements.



- Each DO group can contain multiple statements that apply to the expression.
- Each DO group ends with an END statement.

Business Scenario

Create another new variable named **Freq**.

```
data work.bonus;  
  set orion.sales;  
  if Country='US' then do;  
    Bonus=500;  
    Freq='Once a Year';  
  end;  
  else if Country='AU' then do;  
    Bonus=300;  
    Freq='Twice a Year';  
  end;  
run;
```

Business Scenario

```
proc print data=work.bonus;  
    var First_Name Last_Name  
        Country Bonus Freq;  
run;
```

Partial PROC PRINT Output

Obs	First_Name	Last_Name	Country	Bonus	Freq
60	Billy	Plested	AU	300	Twice a Yea
61	Matsuoka	Wills	AU	300	Twice a Yea
62	Vino	George	AU	300	Twice a Yea
63	Meera	Body	AU	300	Twice a Yea
64	Harry	Highpoint	US	500	Once a Year
65	Julienne	Magolan	US	500	Once a Year
66	Scott	Desanctis	US	500	Once a Year
67	Cherda	Ridley	US	500	Once a Year
68	Priscilla	Farren	US	500	Once a Year
69	Robert	Stevens	US	500	Once a Year

Compilation

```
data work.bonus;  
  set orion.sales;  
  if Country='US' then do;  
    Bonus=500;  
    Freq='Once a Year';  
  end;  
  else if Country='AU' then do;  
    Bonus=300;  
    Freq='Twice a Year';  
  end;  
run;
```

PDV

Employee_ID	First_Name	...	Hire_Date
N 8	\$ 12		N 8

Compilation

```
data work.bonus;  
  set orion.sales;  
  if Country='US' then do;  
    Bonus=500;  
    Freq='Once a Year';  
  end;  
  else if Country='AU' then do;  
    Bonus=300;  
    Freq='Twice a Year';  
  end;  
run;
```

PDV

Employee_ID	First_Name
N 8	\$ 12

...

Hire_Date	Bonus
N 8	N 8

Compilation

```
data work.bonus;  
  set orion.sales;  
  if Country='US' then do;  
    Bonus=500;  
    Freq='Once a Year';  
  end;  
  else if { 11 characters } then do;  
    Bonus=500;  
    Freq='Twice a Year';  
  end;  
run;
```

PDV

Employee_ID	First_Name
N 8	\$ 12

...

Hire_Date	Bonus	Freq
N 8	N 8	\$ 11

9.06 Quiz

How would you prevent **Freq** from being truncated?

The LENGTH Statement (Review)

The *LENGTH* statement defines the length of a variable explicitly.

General form of the LENGTH statement:

```
LENGTH variable(s) $ length;
```

Example:

```
length First_Name Last_Name $ 12  
      Gender $ 1;
```

Business Scenario

Set the length of the variable **Freq** to avoid truncation.

```
data work.bonus;  
  set orion.sales;  
  length Freq $ 12;  
  if Country='US' then do;  
    Bonus=500;  
    Freq='Once a Year';  
  end;  
  else if Country='AU' then do;  
    Bonus=300;  
    Freq='Twice a Year';  
  end;  
run;
```

Business Scenario

```
proc print data=work.bonus;  
  var First_Name Last_Name  
      Country Bonus Freq;  
run;
```

Partial PROC PRINT Output

Obs	First_Name	Last_Name	Country	Bonus	Freq
60	Billy	Plested	AU	300	Twice a Year
61	Matsuoka	Wills	AU	300	Twice a Year
62	Vino	George	AU	300	Twice a Year
63	Meera	Body	AU	300	Twice a Year
64	Harry	Highpoint	US	500	Once a Year
65	Julienne	Magolan	US	500	Once a Year
66	Scott	Desanctis	US	500	Once a Year
67	Cherda	Ridley	US	500	Once a Year
68	Priscilla	Farren	US	500	Once a Year
69	Robert	Stevens	US	500	Once a Year

ELSE Statements

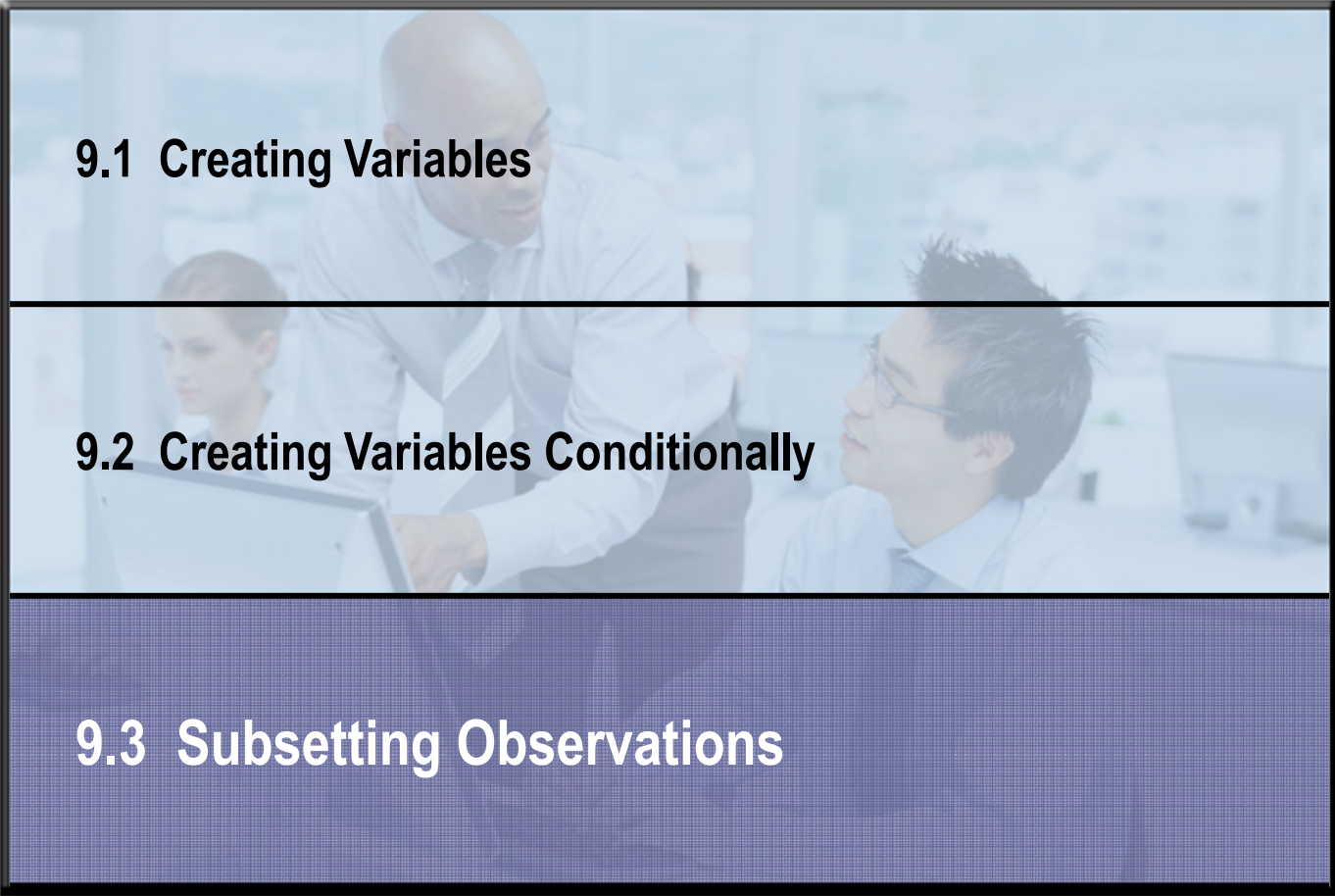
The conditional clause does not have to be in an ELSE statement.

```
data work.bonus;  
  set orion.sales;  
  length Freq $ 12;  
  if Country='US' then do;  
    Bonus=500;  
    Freq='Once a Year';  
  end;  
  else do;  
    Bonus=300;  
    Freq='Twice a Year';  
  end;  
run;
```



All observations not equal to US execute the statements in the second DO group.

Chapter 9: Manipulating Data



9.1 Creating Variables

9.2 Creating Variables Conditionally

9.3 Subsetting Observations

Objectives

- Subset observations by using the WHERE statement.
- Subset observations by using the subsetting IF statement.
- Subset observations by using the IF-THEN DELETE statement. (Self-Study)

Business Scenario

A new SAS data set named **Work.december** needs to be created by reading the **orion.sales** data set.

Work.december must include the following new variables:

- **Bonus**, which is equal to a constant 500.
- **Compensation**, which is the combination of the employee's salary and bonus.
- **BonusMonth**, which is equal to the month the employee was hired.

Work.december must include only the employees from Australia who have a bonus month in December.

The WHERE Statement (Review)

The *WHERE* statement subsets observations that meet a particular condition.

General form of the WHERE statement:

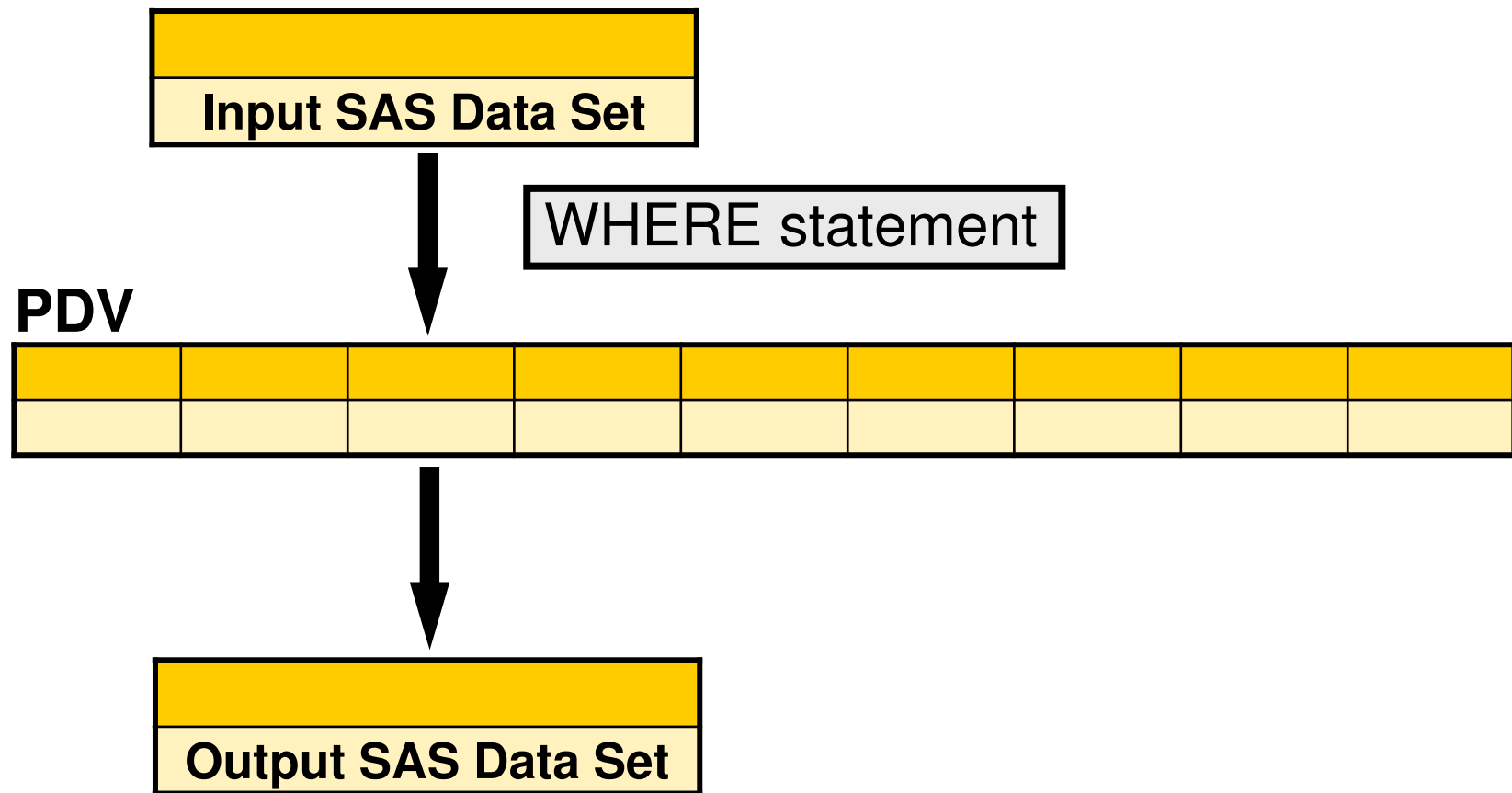
WHERE *where-expression*;

The *where-expression* is a sequence of operands and operators that form a set of instructions that define a condition for selecting observations.

- Operands include constants and variables.
- Operators are symbols that request a comparison, arithmetic calculation, or logical operation.

Processing the WHERE Statement

The WHERE statement selects observations **before** they are brought into the program data vector.



9.07 Quiz

Why does the WHERE statement not work in this DATA step?

```
data work.december;  
    set orion.sales;  
    BonusMonth=month(Hire_Date);  
    Bonus=500;  
    Compensation=sum(Salary, Bonus);  
    where Country='AU' and BonusMonth=12;  
run;
```

The Subsetting IF Statement

The *subsetting IF statement* continues processing only those observations that meet the condition.

General form of the subsetting IF statement:

IF *expression*;

The *expression* is a sequence of operands and operators that form a set of instructions that define a condition for selecting observations.

- Operands include constants and variables.
- Operators are symbols that request a comparison, arithmetic calculation, or logical operation.

The Subsetting IF Statement

Examples:

```
if Salary > 50000;
```

```
if Last_Name='Smith' and First_Name='Joe';
```

```
if Country not in ('GB', 'FR', 'NL');
```

```
if Hire_Date = '15APR2008'd;
```

```
if BirthMonth = 5 or BirthMonth = 6;
```

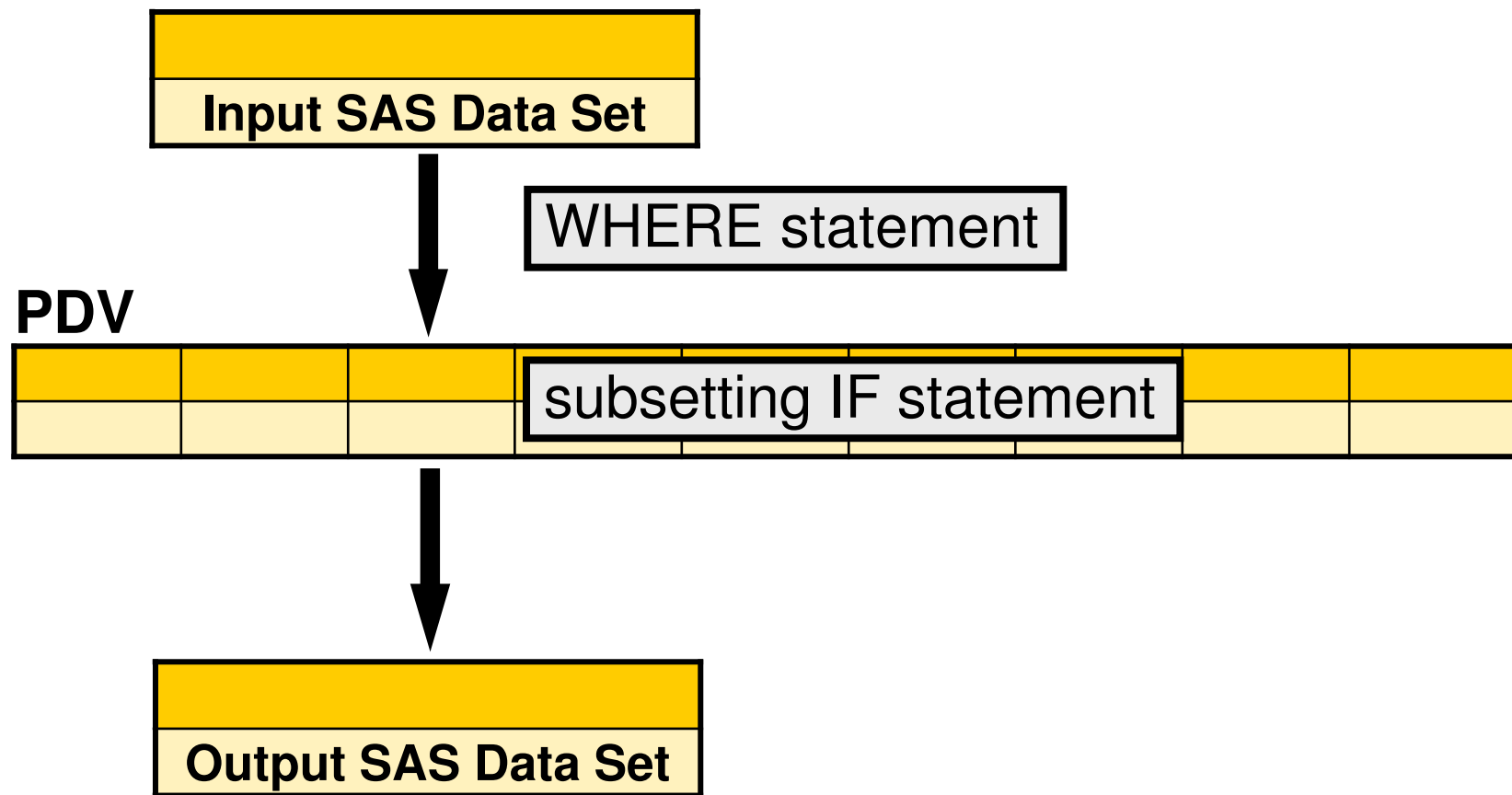
```
if upcase(Gender)='M';
```

```
if 40000 <= Compensation <= 80000;
```

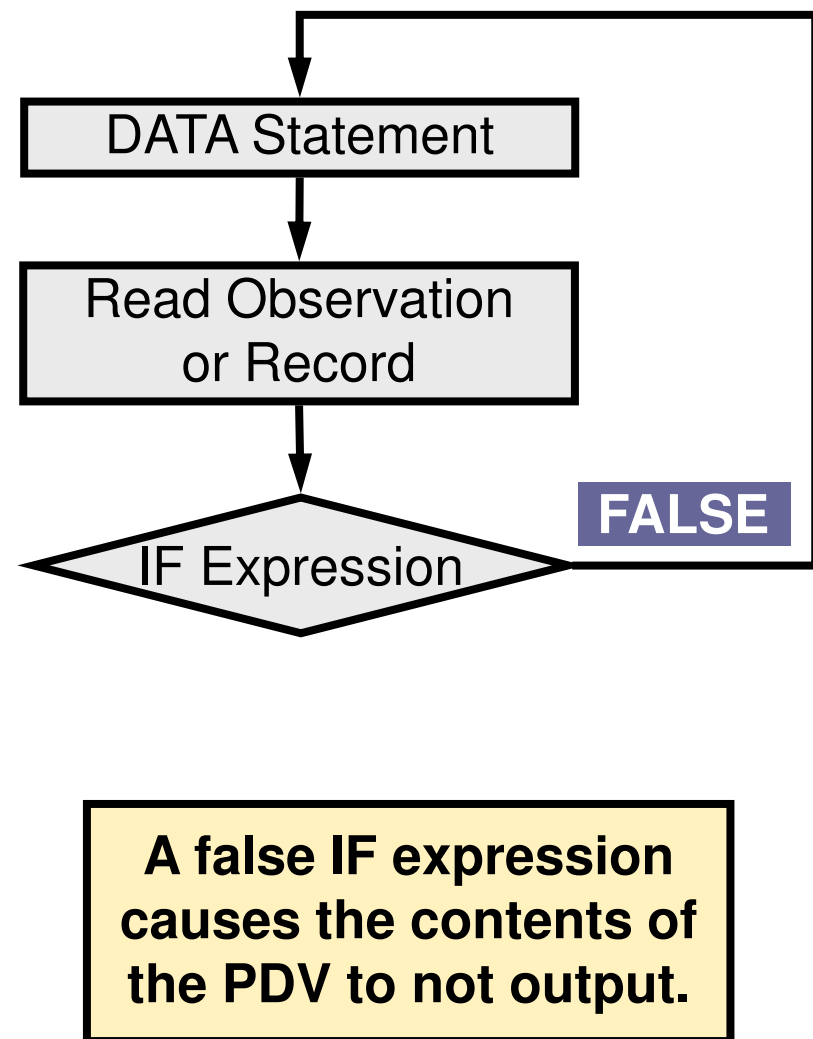
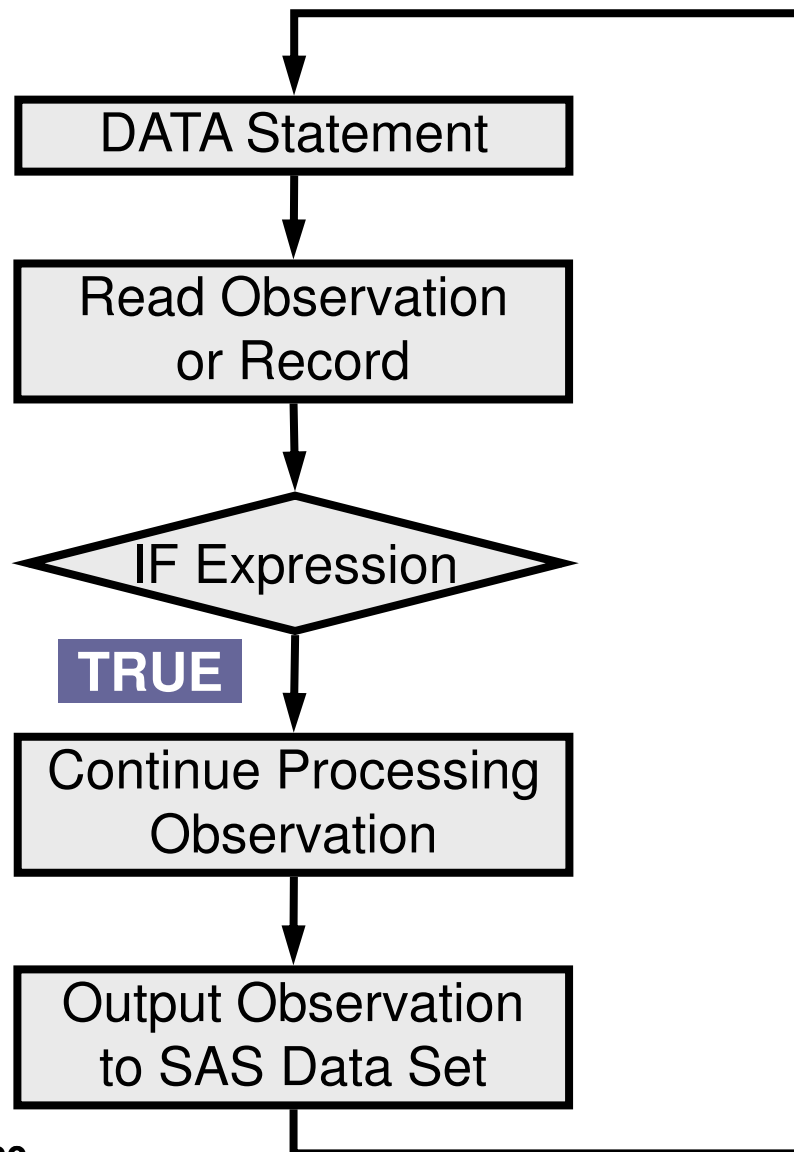
```
if sum(Salary,Bonus) < 43000;
```

Processing the Subsetting IF Statement

The subsetting IF statement determines if observations continue being processed in the program data vector.



Processing the Subsetting IF Statement



Business Scenario

Include only the employees from Australia who have a bonus month in December.

```
data work.december;  
  set orion.sales;  
  where Country='AU';  
  BonusMonth=month(Hire_Date);  
  if BonusMonth=12;  
  Bonus=500;  
  Compensation=sum(Salary, Bonus);  
run;
```

Partial SAS Log

```
NOTE: There were 63 observations read from the data set ORION.SALES.  
      WHERE Country='AU';  
NOTE: The data set WORK.DECEMBER has 3 observations and 12 variables.
```

9.08 Quiz

Could you write only an IF statement?

- ☐ Yes
- ☐ No

```
data work.december;  
  set orion.sales;  
  where Country='AU';  
  BonusMonth=month(Hire_Date);  
  if BonusMonth=12;  
    Bonus=500;  
  Compensation  
run;
```

```
data work.december;  
  set orion.sales;  
  BonusMonth=month(Hire_Date);  
  if BonusMonth=12 and Country='AU';  
    Bonus=500;  
  Compensation=sum(Salary, Bonus);  
run;
```

WHERE Statement versus Subsetting IF Statement

Step and Usage	WHERE	IF
PROC step	Yes	No
DATA step (source of variable)		
INPUT statement	No	Yes
assignment statement	No	Yes
SET statement (single data set)	Yes	Yes
SET/MERGE statement (multiple data sets)		
Variable in ALL data sets	Yes	Yes
Variable not in ALL data sets	No	Yes

The IF-THEN DELETE Statement (Self-Study)

An alternative to the subsetting IF statement is the DELETE statement in an IF-THEN statement.

General form of the IF-THEN DELETE statement:

IF *expression* THEN DELETE;

The *DELETE statement* stops processing the current observation.

The IF-THEN DELETE Statement (Self-Study)

```
data work.december;  
  set orion.sales;  
  where Country='AU';  
  BonusMonth=month(Hire_Date);  
  if BonusMonth ne 12 then delete;  
  Bonus=500;  
  Compensation=sum(Salary, Bonus);  
run;
```

```
data work.december;  
  set orion.sales;  
  where Country='AU';  
  BonusMonth=month(Hire_Date);  
  if BonusMonth=12;  
  Bonus=500;  
  Compensation=sum(Salary, Bonus);  
run;
```

equivalent

Chapter Review

1. What is an advantage in using the SUM function instead of an arithmetic operator?
2. Do the DROP/KEEP statements eliminate variables from the input or output data set in the DATA step?
3. When would you use DO group statements in the DATA step?
4. What is the default length of a numeric variable created in an assignment statement?