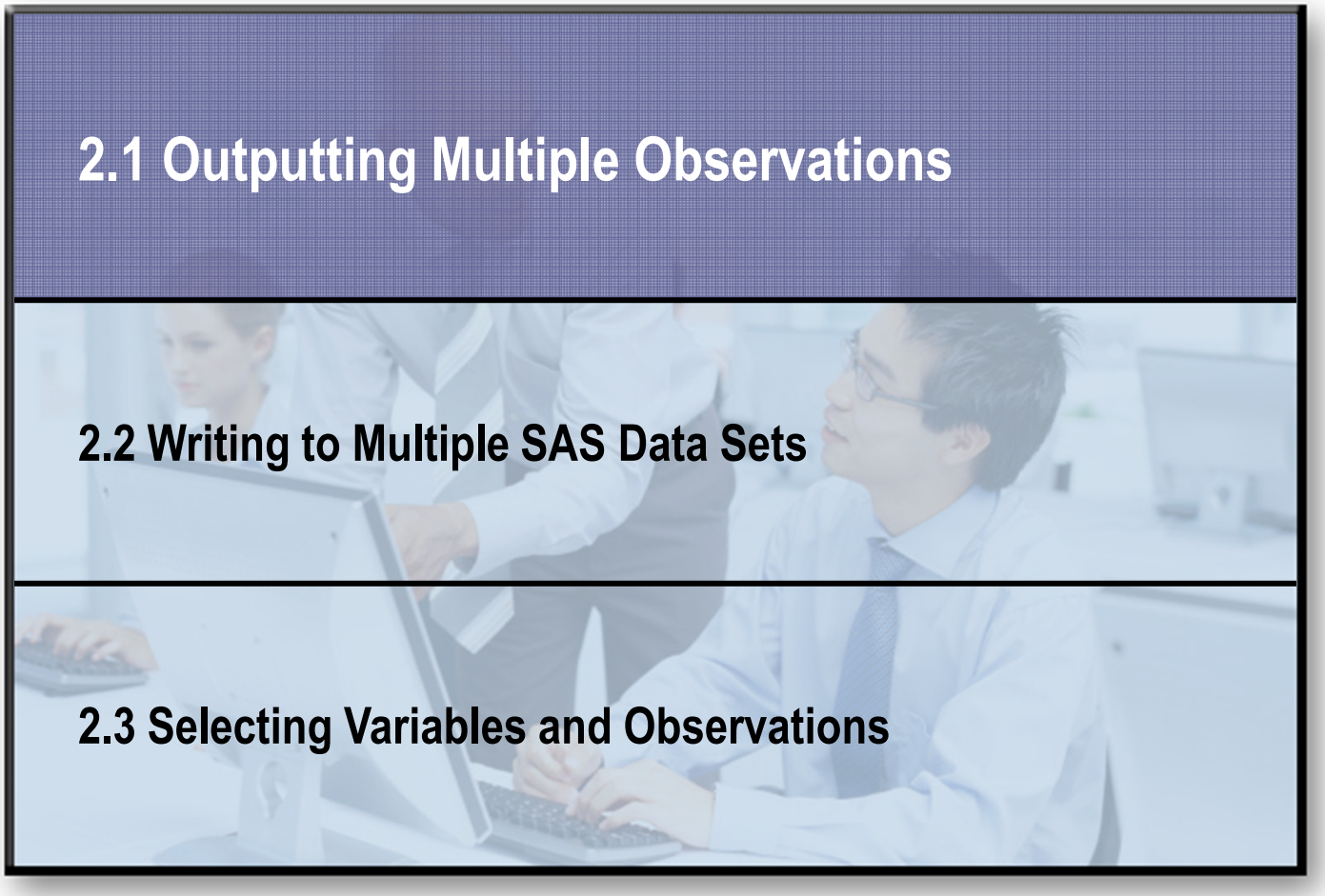
Chapter 2: Controlling Input and Output



Objectives

- Explicitly control the output of multiple observations to a SAS data set.

Business Scenario – A Forecasting Application

The growth rate of six departments at Orion Star is stored in the **Increase** variable in the data set **orion.growth**.

If each department grows at its predicted rate for the next two years, how many employees will be in each department at the end of each year?

Listing of **orion.growth**

Department	Total_ Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS	25	0.10
Marketing	20	0.20
Sales	201	0.30
Sales Management	11	0.10

A Forecasting Application

The output SAS data set, **forecast**, should contain 12 observations. Two observations are written for each observation read.

Partial Listing of **forecast**

Department	Total_ Employees	Increase	Year
Administration	42.500	0.25	1
Administration	53.125	0.25	2
Engineering	11.700	0.30	1
Engineering	15.210	0.30	2
IS	27.500	0.10	1
IS	30.250	0.10	2

2.01 Quiz

Which of the following occur at the end of a DATA step iteration?

```
data forecast;
   set orion.growth;
   total_employees=
      total_employees * (1+increase);
run;
```

?

a. Reinitialize the PDV.
b. Implicit OUTPUT; implicit RETURN.
c. Read the next observation.

Explicit Output

The explicit OUTPUT statement writes the contents of the program data vector (PDV) to the data set or data sets being created. The presence of an explicit OUTPUT statement overrides implicit output.

```
data forecast;
   set orion.growth;
   Year=1;
   Total_Employees=Total_Employees*(1+Increase);
   output;
   Year=2;
   Total_Employees=Total_Employees*(1+Increase);
   output;
run;
```

No Implicit OUTPUT;

Compilation

```
data forecast;
    set orion.growth;
    Year=1;
    Total_Employees=Total_Employees*(1+Increase);
    output;
    Year=2;
    Total_Employees=Total_Employees*(1+Increase);
    output;
run;
```

PDV – Program Data Vector

Department $ 20	Total_ Employees N 8	Increase N 8

...

Compilation

```
data forecast;
   set orion.growth;
   Year=1;
   Total_Employees=Total_Employees*(1+Increase);
   output;
   Year=2;
   Total_Employees=Total_Employees*(1+Increase);
   output;
run;
```

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8

...

Compilation

```
data forecast;
   set orion.growth;
   Year=1;
   Total_Employees=Total_Employees*(1+Increase);
   output;
   Year=2;
   Total_Employees=Total_Employees*(1+Increase);
   output;
run;
```

Write descriptor portion of output data set

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8

work.forecast

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8

Execution: Explicit Output

orion.growth

Department	Total_Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS		.0

```
data forecast;
    set orion.gi
    Year=1;
    Total_Employ                    ees*(1+Increase);
    output;
    Year=2;
    Total_Employees=Total_Employees*(1+Increase);
    output;
run;
```

Initialize PDV

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
	.	.	.

...

Execution: Explicit Output

orion.growth

Department	Total_Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS		.0

```
data forecast;
    set orion.growth;
    Year=1;
    Total_Employees=Total_Employees*(1+Increase);
    output;
    Year=2;
    Total_Employees=Total_Employees*(1+Increase);
    output;
run;
```

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
Administration	34	0.25	.

...

Execution: Explicit Output

orion.growth

Department	Total_Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS		.0

```
data forecast;
   set orion.growth;
   Year=1;
   Total_Employees=Total_Employees*(1+Increase);
   output;
   Year=2;
   Total_Employees=Total_Employees*(1+Increase);
   output;
run;
```

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
Administration	34	0.25	1

...

Execution: Explicit Output

orion.growth

Department	Total_Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS		0

```
data forecast;
   set orion.growth;
   Year=1;
   Total_Employees=Total_Employees*(1+Increase);
   output;
   Year=2;
   Total_Employees=Total_Employees*(1+Increase);
   output;
run;
```

34 * (1 + 0.25)

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
Administration	42.5	0.25	1

...

Execution: Explicit Output

orion.growth

Department	Total_Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS		0

```
data forecast;
   set orion.growth;
   Year=1;
   Total_Employees=Total_Employees*(1+Increase);
   output;
   Year=2
   Total_
   output
run;
```

Increase);

Output current observation

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
Administration	42.5	0.25	1

...

Execution: Explicit Output

orion.growth

Department	Total_Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS		.0

```
data forecast;
   set orion.growth;
   Year=1;
   Total_Employees=Total_Employees*(1+Increase);
   output;
   Year=2;
   Total_Employees=Total_Employees*(1+Increase);
   output;
run;
```

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
Administration	42.5	0.25	2

...

Execution: Explicit Output

orion.growth

Department	Total_Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS		.0

```
data forecast;
   set orion.growth;
   Year=1;
   Total_Employees=Total_Employees*(1+Increase);
   output;
   Year=2;
   Total_Employees=Total_Employees*(1+Increase);
   output;
run;
```

42.5 * (1 + 0.25)

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
Administration	53.125	0.25	2

...

Execution: Explicit Output

orion.growth

Department	Total_Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS		0

```
data forecast;
   set orion.growth;
   Year=1;
   Total_Employees=Total_Employees*(1+Increase);
   output;
   Year=2;
   Total_Employees=Total_Employees*(1+Increase);
   output;
run;
```

Output current observation

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
Administration	53.125	0.25	2

...

Execution: Explicit Output

orion.growth

Department	Total_Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS		0

```
data forecast;
   set orion.growth;
   Year=1;
   Total_Employees=Total_Employees*(1+Increase);
   output;
   Year=2;
   Total_Employees=Total_Employees*(1+Increase);
   output;
run;
```

No Implicit OUTPUT;

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
Administration	53.125	0.25	2

Output: A Forecasting Application

The **forecast** data set contains two observations after the first iteration of the DATA step.

work.forecast

Department	Total_ Employees	Increase	Year
Administration	42.500	0.25	1
Administration	53.125	0.25	2

Setup for the Poll

Prior to the second iteration of the DATA step, some variables in the program data vector will be reinitialized.

```
data forecast;
   set orion.growth;
   Year=1;
   Total_Employees=Total_Employees*(1+Increase);
   output;
   Year=2;
   Total_Employees=Total_Employees*(1+Increase);
   output;
run;
```

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
Administration	53.125	0.25	2

2.02 Multiple Answer Poll

Which variable(s) will be reinitialized?

a. **`Department`**

b. **`Total_Employees`**

c. **`Increase`**

d. **`Year`**

Execution: Explicit Output

orion.growth

Department	Total_Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS		.0

```
data forecast;
    set orion.g
    Year=1;
    Total_Employ                    es*(1+Increase);
    output;
    Year=2;
    Total_Employees=Total_Employees*(1+Increase);
    output;
run;
```

Reinitialize PDV

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
Administration	53.125	0.25	.

...

Execution: Explicit Output

orion.growth

Department	Total_Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS		.0

```
data forecast;
    set orion.growth;
    Year=1;
    Total_Employees=Total_Employees*(1+Increase);
    output;
    Year=2;
    Total_Employees=Total_Employees*(1+Increase);
    output;
run;
```

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
Engineering	9	0.30	.

...

Execution: Explicit Output

orion.growth

Department	Total_Employees	Increase
Administration	34	0.25
Engineering	9	0.30
IS		0

```
data forecast;
   set orion.growth;
   Year=1;
   Total_Emp                    s*(1+Increase);
   output;
   Year=2;
   Total_Employees=Total_Employees*(1+Increase);
   output;
run;
```

Continue until EOF

PDV

Department $ 20	Total_ Employees N 8	Increase N 8	Year N 8
Engineering	9	0.30	1

Check the Results

Partial SAS Log

```
NOTE: There were 6 observations read from the data set
      ORION.GROWTH.
NOTE: The data set WORK.FORECAST has 12 observations
      and 4 variables.
```

Partial PROC PRINT Output

Department	Total_ Employees	Year
Administration	42.500	1
Administration	53.125	2
Engineering	11.700	1
Engineering	15.210	2
IS	27.500	1
IS	30.250	2
Marketing	24.000	1
Marketing	28.800	2

2.03 Quiz

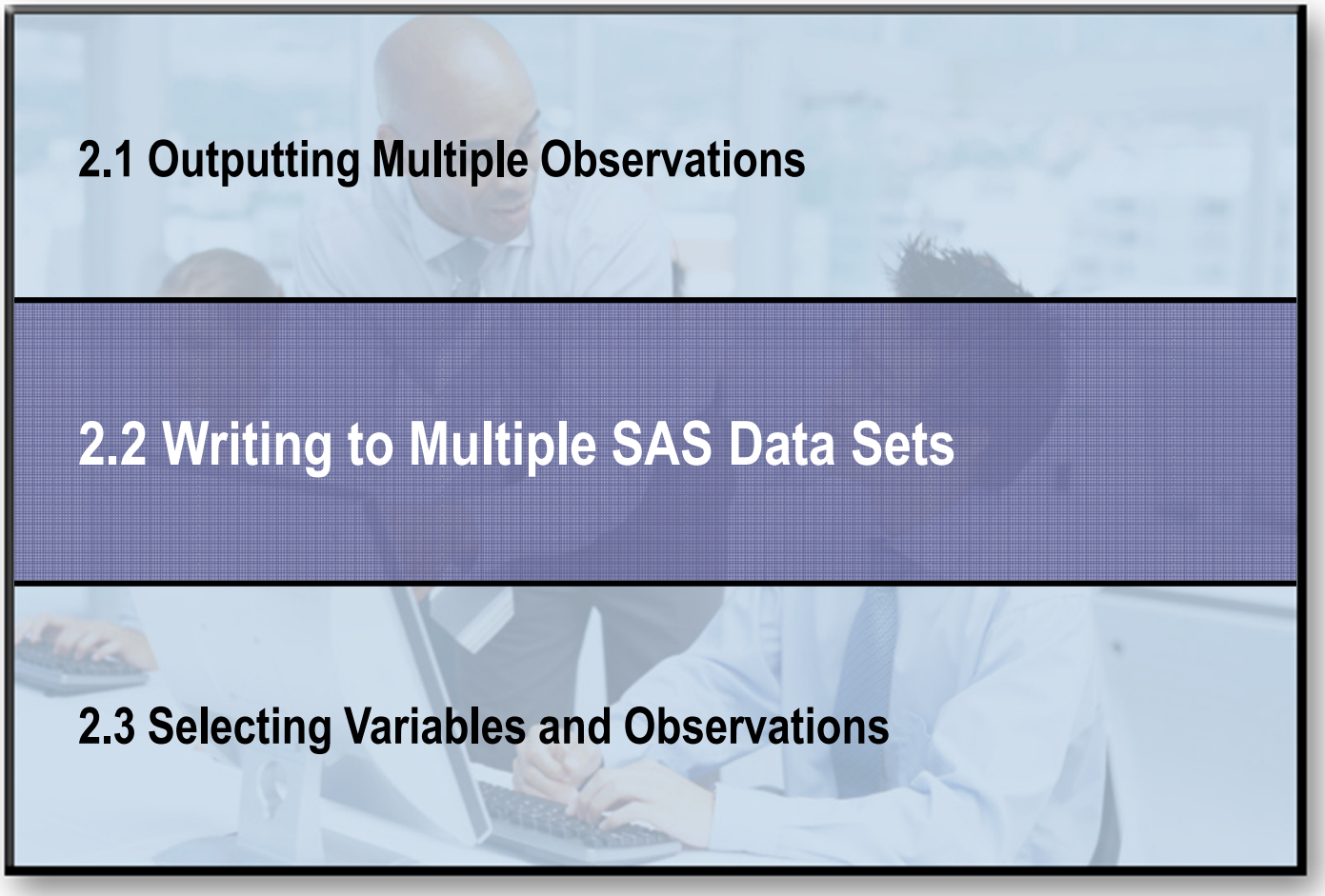
Open and submit **p202a01**. Modify the DATA step to write only one observation per department. Show the number of employees after two years.

Desired Results

Department	Total_ Employees	Year
Administration	53.125	2
Engineering	15.210	2
IS	30.250	2
Marketing	28.800	2
Sales	339.690	2
Sales Management	13.310	2

p202a01

Chapter 2: Controlling Input and Output



Objectives

- Create multiple SAS data sets in a single DATA step.
- Use conditional processing to control the data set(s) to which an observation is written.

Business Scenario

Use the **orion.employee_addresses** data set as input to create three new data sets: **usa**, **australia**, and **other**.

- The **usa** data set will contain observations with a **Country** value of `US`.
- The **australia** data set will contain observations with a **Country** value of `AU`.
- Observations with any other **Country** value will be written to the **other** data set.

Browse the Input Data Set

```
proc print data=orion.employee_addresses;
   var Employee_Name City Country;
run;
```

Partial PROC PRINT Output

Obs	Employee_Name	City	Country
1	Abbott, Ray	Miami-Dade	US
2	Aisbitt, Sandy	Melbourne	AU
3	Akinfolarin, Tameaka	Philadelphia	US
4	Amos, Salley	San Diego	US
5	Anger, Rose	Philadelphia	US
6	Anstey, David	Miami-Dade	US
7	Antonini, Doris	Miami-Dade	US
8	Apr, Nishan	San Diego	US
9	Ardskin, Elizabeth	Miami-Dade	US
10	Areu, Jeryl	Miami-Dade	US
11	Arizmendi, Gilbert	San Diego	US
12	Armant, Debra	San Diego	US

Creating Multiple DATA Sets

Multiple data sets can be created in a DATA step by listing the names of the output data sets in the DATA statement.

DATA <*SAS-data-set-1 … SAS-data-set-n*>**;**

You can direct output to a specific data set or data sets by listing the data set names in the OUTPUT statement.

OUTPUT <*SAS-data-set-1 … SAS-data-set-n*>**;**

An OUTPUT statement without arguments writes to every SAS data set listed in the DATA statement.

Creating Multiple SAS Data Sets

Create three new data sets: **usa**, **australia**, and **other**.

```
data usa australia other;
   set orion.employee_addresses;
   if Country='AU' then output australia;
   else if Country='US' then output usa;
   else output other;
run;
```

Check the SAS Log

Three data sets were created. The log shows that US was the most frequently occurring value.

Partial SAS Log

```
NOTE: There were 424 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.USA has 311 observations and 9
      variables.
NOTE: The data set WORK.AUSTRALIA has 105 observations and
      9  variables.
NOTE: The data set WORK.OTHER has 8 observations and 9
      variables.
```

Efficient Conditional Processing

It is more efficient to check values in order of decreasing frequency.

```
data usa australia other;
   set orion.employee_addresses;
   if Country='US' then output usa;
   else if Country='AU' then output australia;
   else output other;
run;
```

```
NOTE: There were 424 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.USA has 311 observations and 9
      variables.
NOTE: The data set WORK.AUSTRALIA has 105 observations and
      9 variables.
NOTE: The data set WORK.OTHER has 8 observations and 9
      variables.
```

2.04 Quiz

Consider the results of the previous DATA step.

Can all three data sets be printed with a single PRINT procedure?

Partial SAS Log

```
NOTE: There were 424 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.USA has 311 observations and 9
      variables.
NOTE: The data set WORK.AUSTRALIA has 105 observations and
      9 variables.
NOTE: The data set WORK.OTHER has 8 observations and 9
      variables.
```

Displaying Multiple SAS Data Sets

The PRINT procedure can only print one data set. A separate PROC PRINT step is required for each data set.

```
title 'Employees in the United States';
proc print data=usa;
run;

title 'Employees in Australia';
proc print data=australia;
run;

title 'Non US and AU Employees';
proc print data=other;
run;

title;
```

Using a SELECT Group

An alternate form of conditional execution uses a SELECT group.

```
SELECT <(select-expression)>;
     WHEN-1 (value-1 <…,value-n>)
          statement;
     <…WHEN-n (value-1 <…,value-n>)
          statement;>
     <OTHERWISE statement;>
END;
```

The *select-expression* specifies any SAS expression that evaluates to a single value.

- Often a variable name is used as the *select-expression*.

Using a SELECT Group

The previous task can be rewritten using a SELECT group:

```
data usa australia other;
   set orion.employee_addresses;
   select (Country);
      when ('US') output usa;
      when ('AU') output australia;
      otherwise output other;
   end;
run;
```

The SELECT statement processes the WHEN statements from top to bottom, so it is more efficient to check the values in order of decreasing frequency.

Check the SAS Log

Results using SELECT are the same as IF-THEN/ELSE results.

Partial SAS Log

```
NOTE: There were 424 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.USA has 311 observations and 9
      variables.
NOTE: The data set WORK.AUSTRALIA has 105 observations and 9
      variables.
NOTE: The data set WORK.OTHER has 8 observations and 9
      variables.
```

The OTHERWISE Statement

The OTHERWISE statement is optional, but omitting it results in an error when all WHEN conditions are false.

```
SELECT <(select-expression)>;
      WHEN-1 (value-1 <…,value-n>)
            statement;
      <…WHEN-n (value-1 <…,value-n>)
            statement;>
      <OTHERWISE statement;>
END;
```

Use the OTHERWISE statement followed by a null statement to prevent SAS from issuing an error message.

```
otherwise;
```

2.05 Quiz

Open the file **p202a03** and submit it. View the log, identify and correct the problem, and resubmit the program.

```
data usa australia;
   set orion.employee_addresses;
   select (Country);
      when ('US') output usa;
      when ('AU') output australia;
   end;
run;
```

Test for Multiple Values in a WHEN Statement

Multiple values can be listed in the WHEN expression.

```
data usa australia other;
   set orion.employee_addresses;
   select (Country);
      when ('US','us') output usa;
      when ('AU','au') output australia;
      otherwise output other;
   end;
run;
```

Partial SAS Log

```
NOTE: There were 424 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.USA has 316 observations and 9 variables.
NOTE: The data set WORK.AUSTRALIA has 108 observations and 9
      variables.
NOTE: The data set WORK.OTHER has 0 observations and 9 variables.
```

Using Functions in a Select Expression

An alternate solution uses the UPCASE function.

```
data usa australia other;
   set orion.employee_addresses;
   select (upcase(Country));
      when ('US') output usa;
      when ('AU') output australia;
      otherwise output other;
   end;
run;
```

Partial SAS Log

```
NOTE: There were 424 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.USA has 316 observations and 9 variables.
NOTE: The data set WORK.AUSTRALIA has 108 observations and 9
      variables.
NOTE: The data set WORK.OTHER has 0 observations and 9 variables.
```

Using DO-END in a SELECT Group

Use DO and END statements to execute multiple statements when an expression is true.

```
data usa australia other;
   set orion.employee_addresses;
   select (upcase(country));
      when ('US') do;
         Benefits=1;
         output usa;
      end;
      when ('AU') do;
         Benefits=2;
         output australia;
      end;
      otherwise do;
         Benefits=0;
         output other;
      end;
   end;
run;
```

Omitting the Select Expression

The *select-expression* can be omitted in a SELECT group:

```
SELECT;
      WHEN (expression-1)
           statement;
  <…WHEN (expression-n)
           statement;>
    <OTHERWISE statement;>
END;
```

Each WHEN expression evaluates to true or false.

- If true, the associated statement(s) is executed.
- If false, SAS proceeds to the next WHEN statement.
- If all WHEN expressions are false, then the statement(s) following the OTHERWISE statement executes.

Omitting the Select Expression

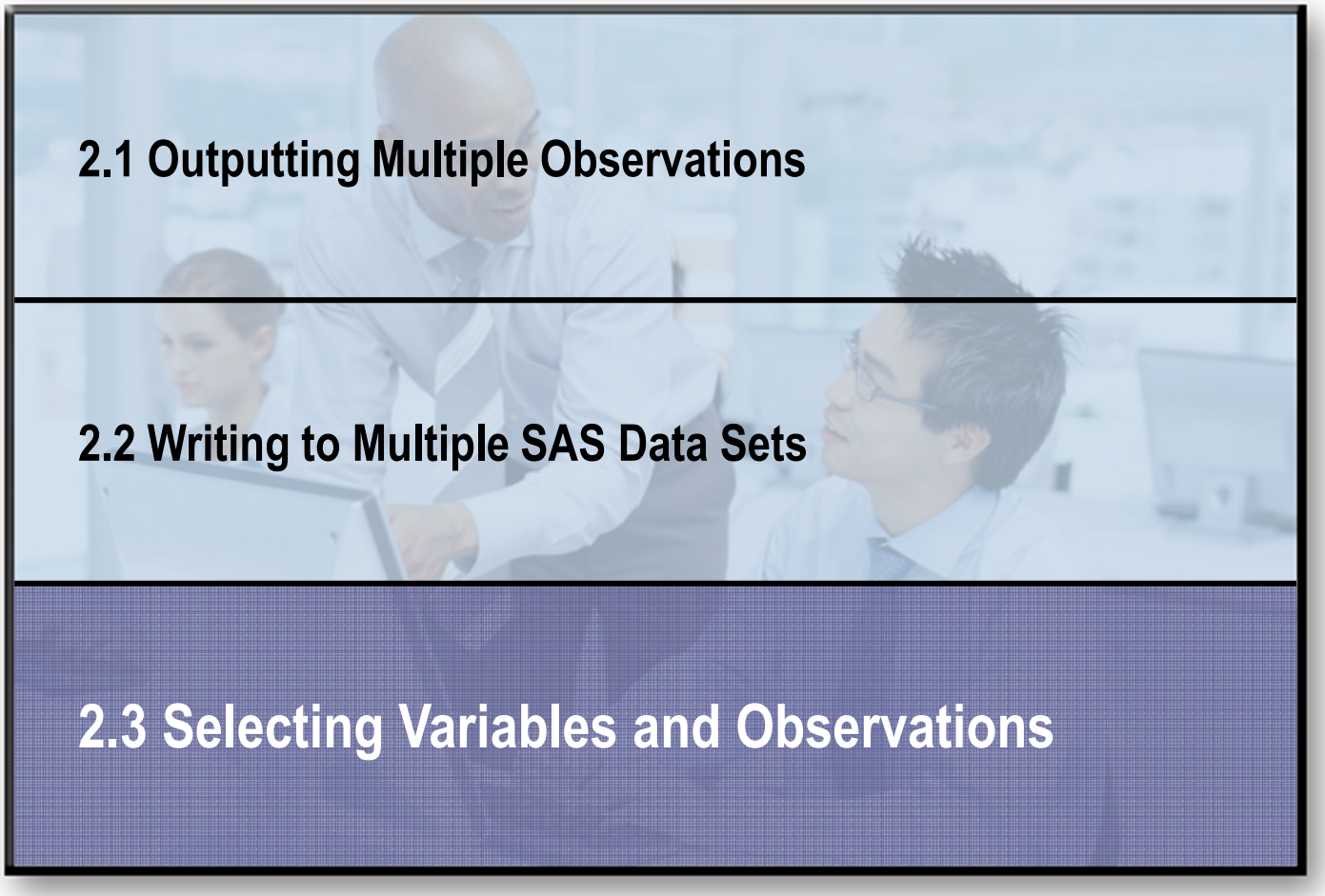
This version of the current example omits the SELECT expression:

```
data usa australia other;
   set orion.employee_addresses;
   select;
      when (country='US') output usa;
      when (country='AU') output australia;
      otherwise output other;
   end;
run;
```

Partial SAS Log

```
NOTE: There were 424 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.USA has 311 observations and 9 variables.
NOTE: The data set WORK.AUSTRALIA has 105 observations and 9
      variables.
NOTE: The data set WORK.OTHER has 8 observations and 9 variables.
```

Chapter 2: Controlling Input and Output



Objectives

- Control which variables are written to an output data set during a DATA step.
- Control which variables are read from an input data set during a DATA step.
- Control how many observations are processed from an input data set during a DATA or PROC step.

Business Scenario

Create three data sets that are subsets of **`orion.employee_addresses`** based on the value of the **`Country`** variable.

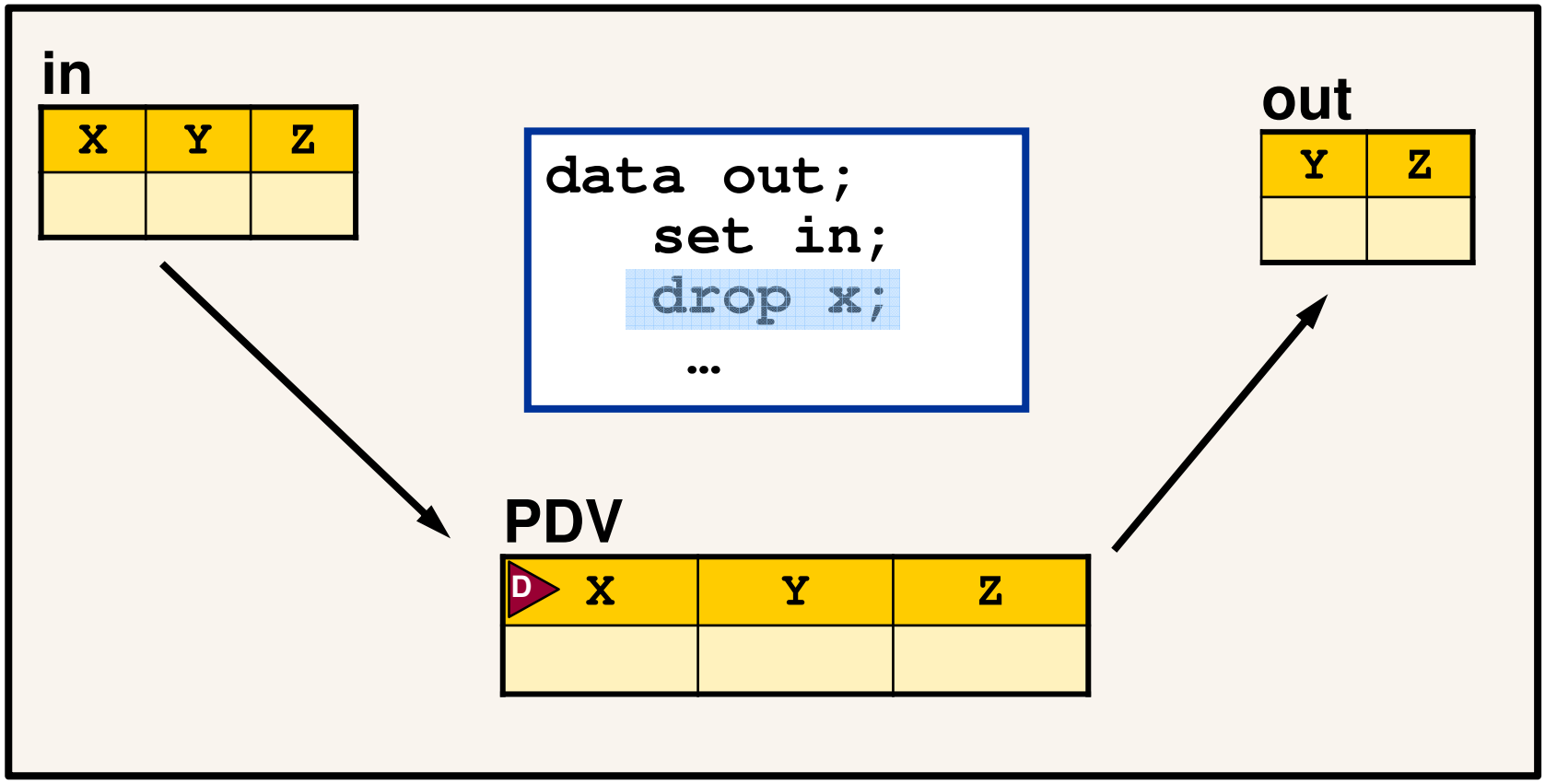
Name the data sets **`usa`**, **`australia`**, and **`other`**, and write different variables to each output data set.

Controlling Variable Output (Review)

By default, SAS writes all variables from the input data set to every output data set.

In the DATA step, the DROP and KEEP statements can be used to control which variables are written to output data sets.

Controlling Variable Output (Review)



The DROP and KEEP statements affect output data sets. The statements can be used when reading from a SAS data set or from a raw data file.

Display Information About the Variables

The **orion.employee_addresses** data set contains nine variables.

```
proc contents data=orion.employee_addresses;
run;
```

Partial PROC CONTENTS Output

```
      ---Alphabetic List of Variables and Attributes---

           #    Variable            Type    Len

           6    City                Char     30
           9    Country             Char      2
           1    Employee_ID         Num       8
           2    Employee_Name       Char     40
           8    Postal_Code         Char     10
           7    State               Char      2
           3    Street_ID           Num       8
           5    Street_Name         Char     40
           4    Street_Number       Num       8
```

The DROP Statement

The DROP statement drops variables from every output data set.

```
data usa australia other;
   drop Street_ID;
   set orion.employee_addresses;
   if Country='US' then output usa;
   else if Country='AU' then output australia;
   else output other;
run;
```

Partial SAS Log

```
NOTE: There were 424 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.USA has 311 observations and 8 variables.
NOTE: The data set WORK.AUSTRALIA has 105 observations and 8
      variables.
NOTE: The data set WORK.OTHER has 8 observations and 8 variables.
```

Controlling Variable Output

The task is to drop **Street_ID** and **Country** from **usa**, drop **Street_ID**, **Country**, and **State** from **australia**, and keep all variables in **other**.

USA

Employee_ ID	Employee_Name	Street_ Number	Street_Name	City	State	Postal_ Code
121044	Abbott, Ray	2267	Edwards Mill Rd	Miami-Dade	FL	33135
120761	Akinfolarin, Tameaka	5	Donnybrook Rd	Philadelphia	PA	19145
120656	Amos, Salley	3524	Calico Ct	San Diego	CA	92116

Australia

Employee_ ID	Employee_Name	Street_ Number	Street_Name	City	Postal_ Code
120145	Aisbitt, Sandy	30	Bingera Street	Melbourne	2001
120185	Bahlman, Sharon	24	LaTrobe Street	Sydney	2165
120109	Baker, Gabriele	166	Toorak Road	Sydney	2119

Other

Employee_ ID	Employee_Name	Street_ID	Street_ Number	Street_Name	City	State	Postal_ Code	Country
121019	Desanctis, Scott	9260121087	765	Greenhaven Ln	Philadelphia	PA	19102	us
120997	Donathan, Mary	9260121069	4923	Gateridge Dr	Philadelphia	PA	19152	us
120747	Farthing, Zashia	9260123756	763	Chatterson Dr	San Diego	CA	92116	us

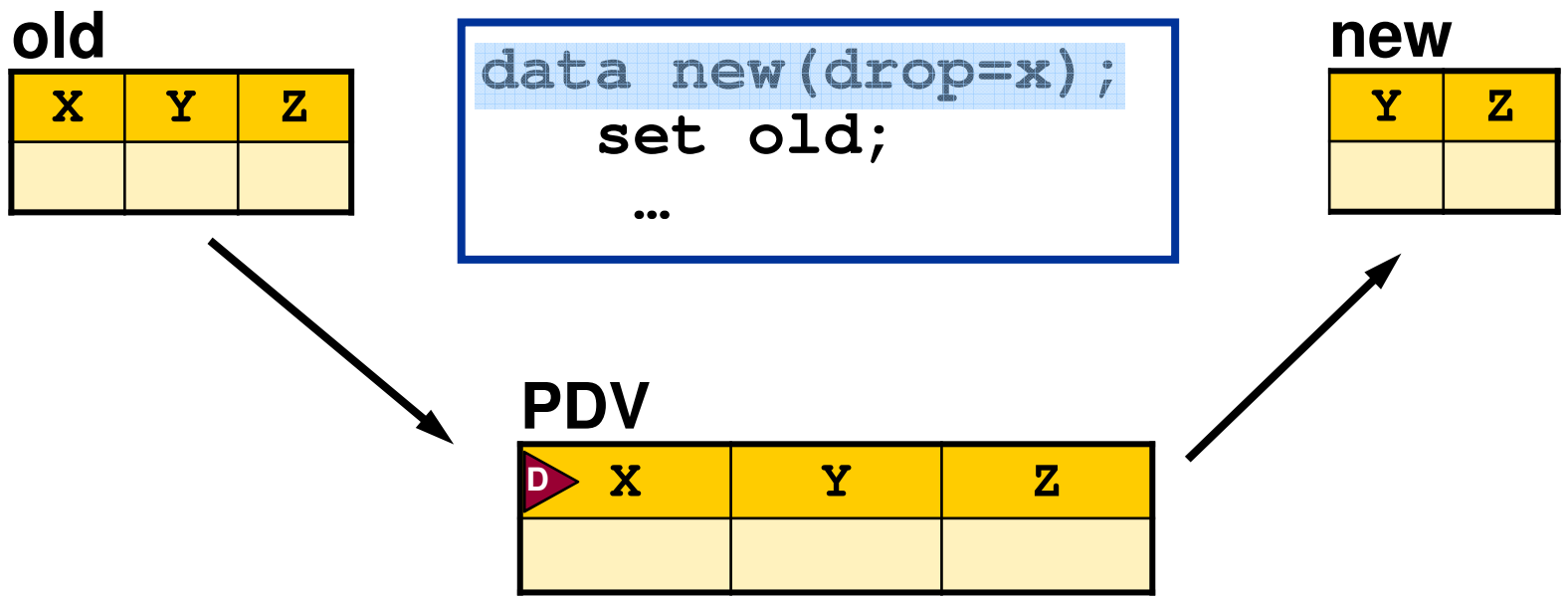
The DROP= Data Set Option

The DROP= data set option can be used to exclude variables from a specific output data set.

General form of the DROP= data set option:

SAS-data-set(DROP=*variable-1 <variable-2 …variable-n>*)

The DROP= Option on an Output Data Set



The specified variables are **not** written to the output data set; however, all variables are available for processing.

Using the DROP= Data Set Option

```
data usa(drop=Street_ID Country)
     australia(drop=Street_ID State Country)
     other;
   set orion.employee_addresses;
   if Country='US' then output usa;
   else if Country='AU' then
        output australia;
   else output other;
run;
```

```
NOTE: There were 424 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.USA has 311 observations
      and 7 variables.
NOTE: The data set WORK.AUSTRALIA has 105 observations
      and 6 variables.
NOTE: The data set WORK.OTHER has 8 observations
      and 9 variables.
```

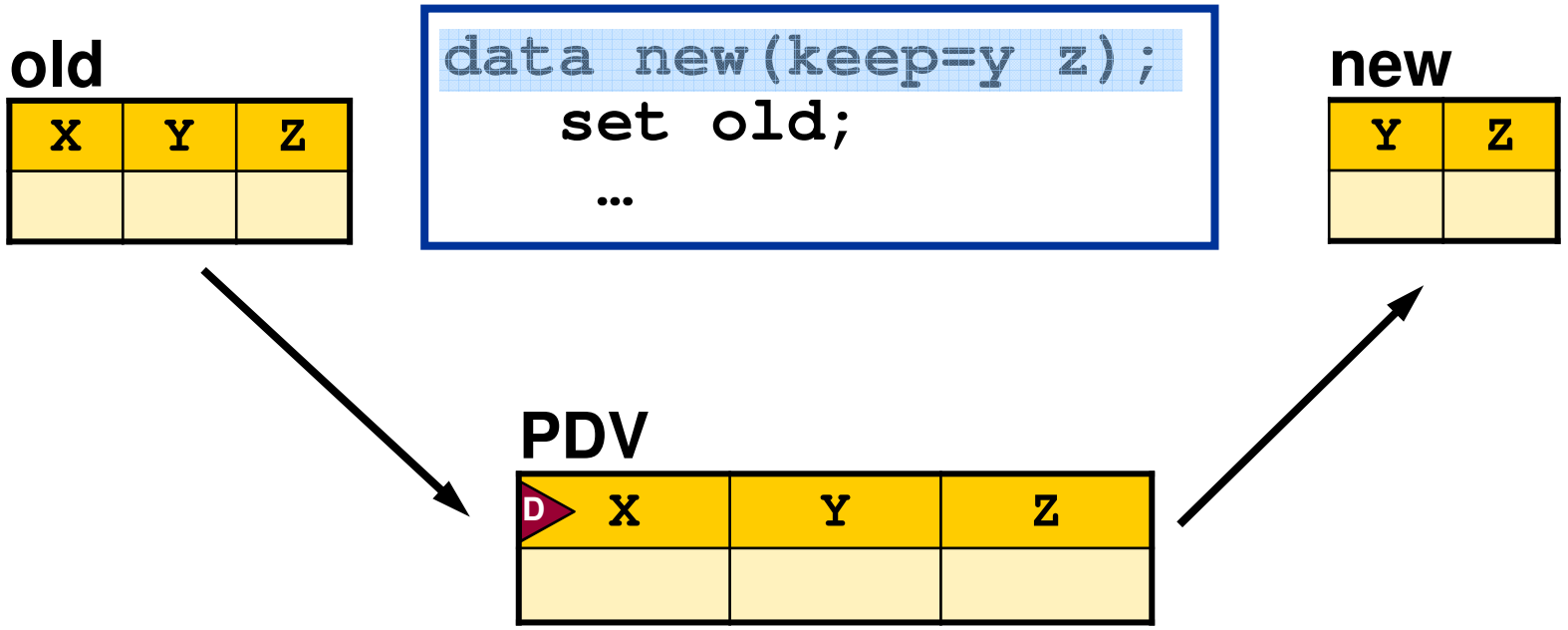
The KEEP= Data Set Option

The KEEP= data set option can be used to specify which variables to write to a specific output data set.

General form of the KEEP= data set option:

SAS-data-set(KEEP=*variable-1 <variable-2 …variable-n>*)

The KEEP= Option on an Output Data Set



Only the specified variables are written to the output data set; however, all variables are available for processing.

Using the DROP= and KEEP= Options

The DROP= and KEEP= options can both be used in a SAS program.

```
data usa(keep=Employee_Name City State)
     australia(drop=Street_ID State)
     other;
   set orion.employee_addresses;
   if Country='US' then output usa;
   else if Country='AU' then output australia;
   else output other;
run;
```

Attempting to drop and keep the same variable in a data set results in a warning.

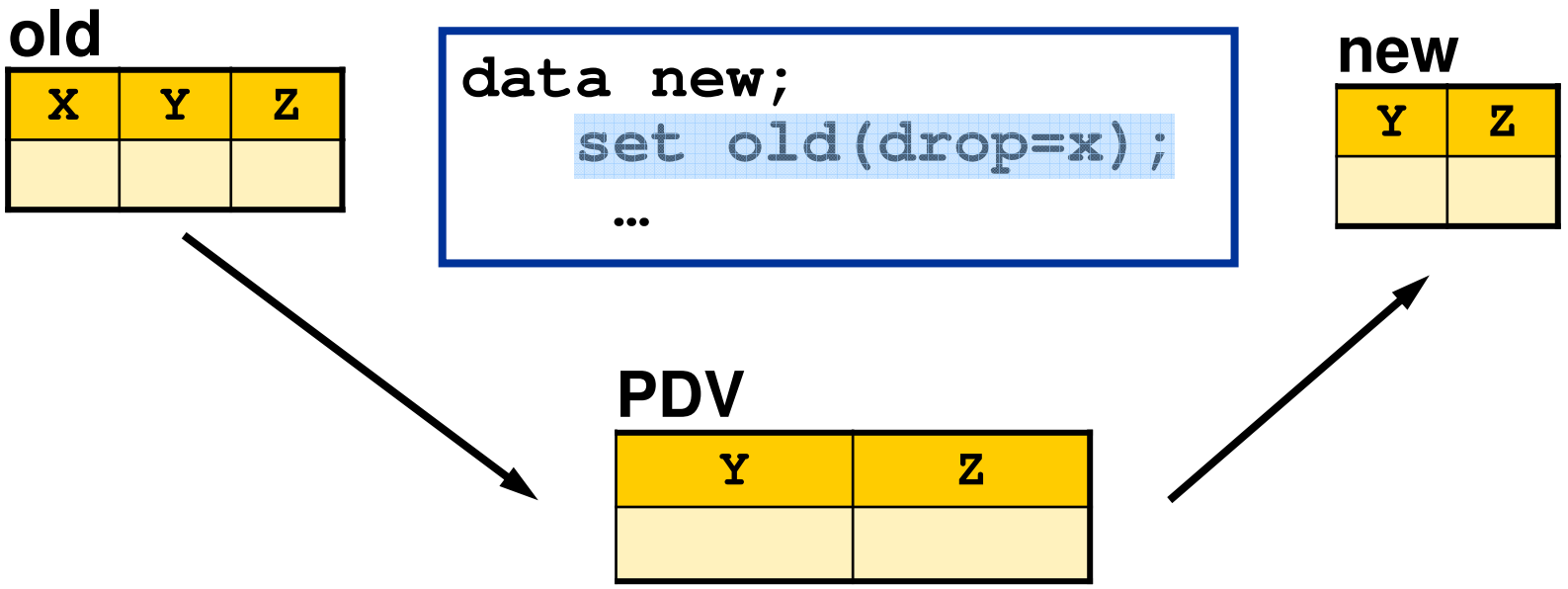
2.06 Quiz

The data set **orion.employee_addresses** contains nine variables. How many variables will be in the **usa**, **australia**, and **other** data sets?

```
data usa(keep=Employee_Name City State Country)
     australia(drop=Street_ID State Country)
     other;
   set orion.employee_addresses;
   if Country='US' then output usa;
   else if Country='AU' then output australia;
   else output other;
run;
```

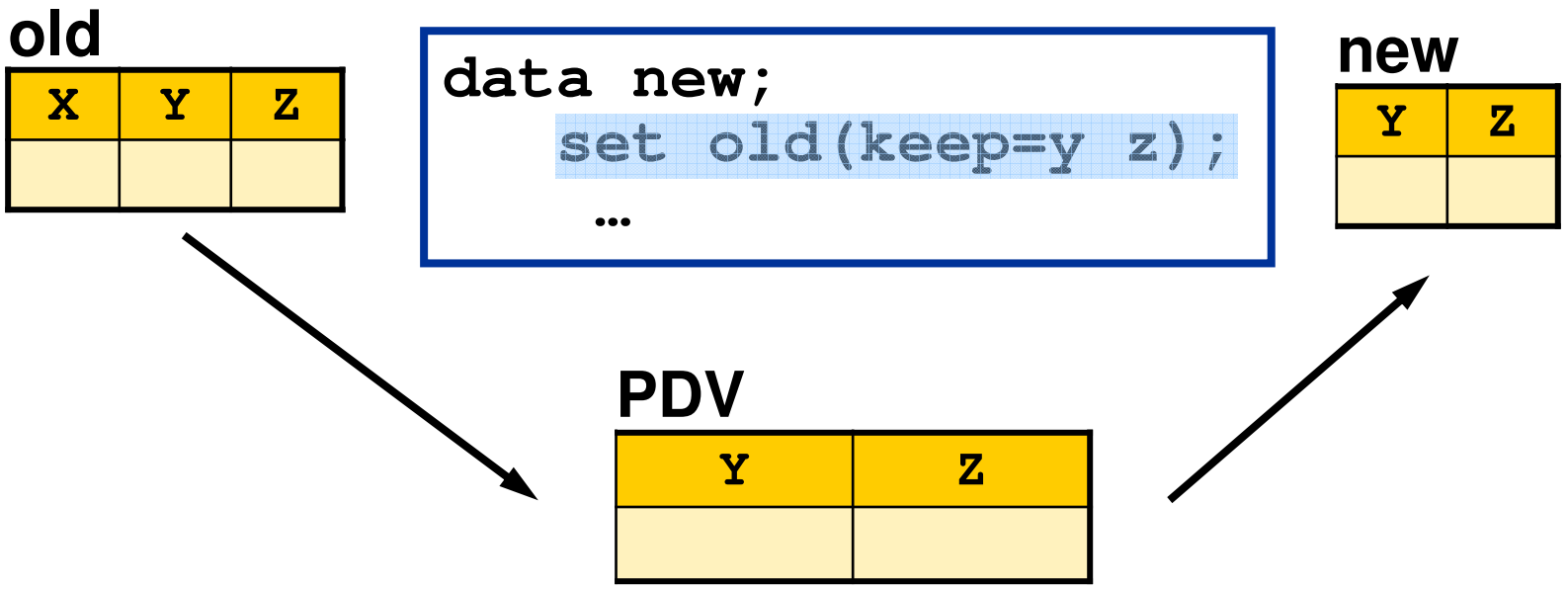
Using DROP= on an Input Data Set

When a **DROP=** data set option is used on an input data set, the specified variables are not read into the PDV, and therefore are **not** available for processing.



Using KEEP= on an Input Data Set

When a **KEEP=** data set option is used on an input data set, only the specified variables are read into the PDV, and therefore **are** available for processing.



2.07 Quiz

Open file **p202a05** and submit it. The intent is to drop **Country**, **Street_ID**, and **Employee_ID** from every data set, and to drop **State** from **australia**. What is wrong with the program?

```
data usa australia(drop=State) other;
   set orion.employee_addresses
       (drop=Country Street_ID Employee_ID);
   if Country='US' then output usa;
   else if Country='AU' then output australia;
   else output other;
run;
```

An Improved Solution

Use a combination of the DROP= option and the DROP statement to achieve the desired results.

```
data usa australia(drop=State) other;
  set orion.employee_addresses
      (drop=Street_ID Employee_ID);
  drop Country;
  if Country='US' then output usa;
  else if Country='AU' then output australia;
  else output other;
run;
```

State is only dropped from australia

PDV

City	D	Country	Employee _Name	Postal _Code	D	State	Street _Name	Street_ Number

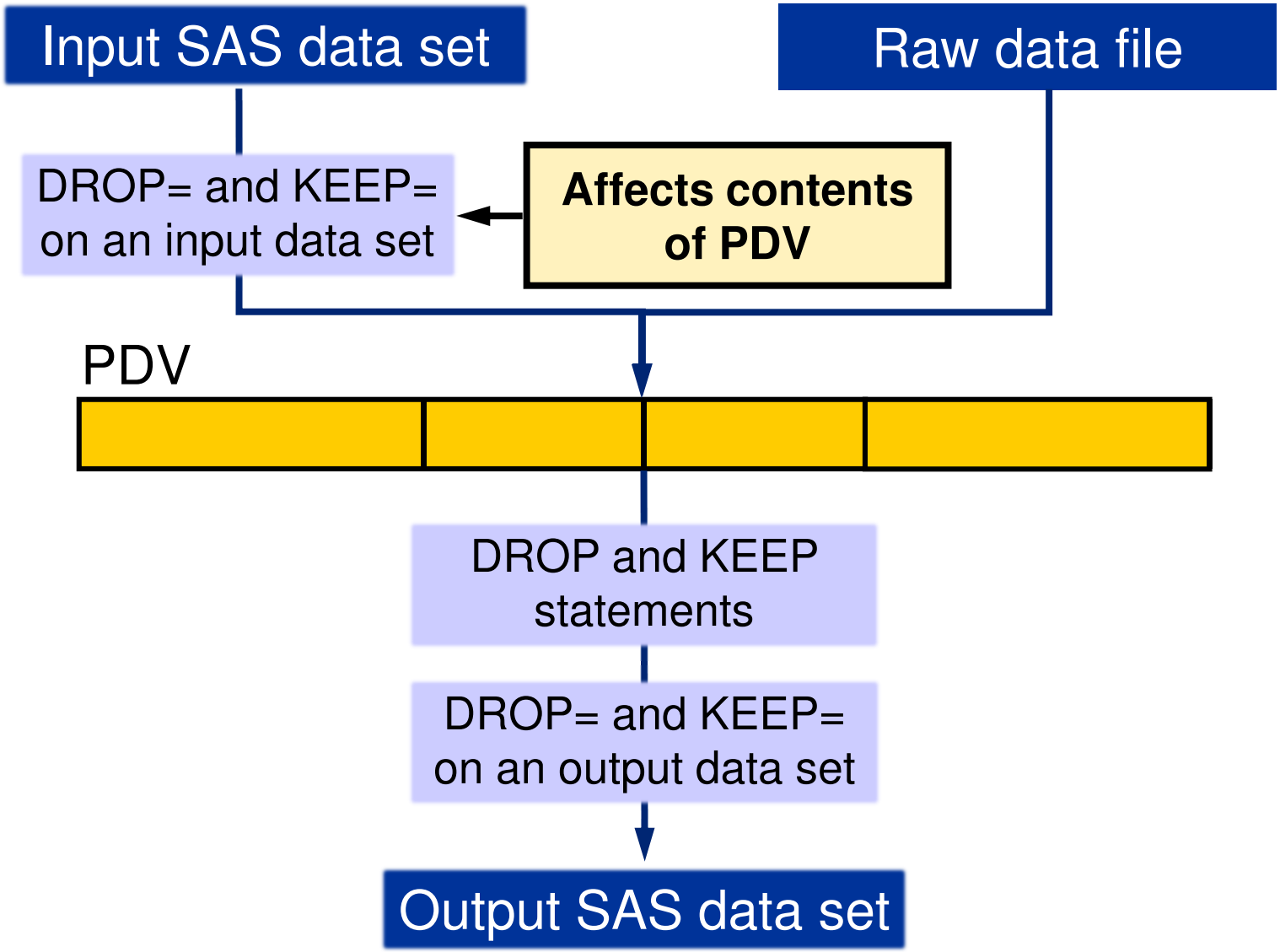
p202a05s

Check the SAS Log

Partial SAS Log

```
NOTE: There were 424 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.USA has 311 observations
      and 6 variables.
NOTE: The data set WORK.AUSTRALIA has 105 observations
      and 5 variables.
NOTE: The data set WORK.OTHER has 8 observations
      and 6 variables.
```

Controlling Variable Input



Controlling Which Observations Are Read

By default, SAS processes every observation in a SAS data set, from the first observation to the last. The FIRSTOBS= and OBS= data set options can be used to control which observations are processed.

The FIRSTOBS= and OBS= options are used with input data sets. You cannot use either option with output data sets.

The OBS= Data Set Option

The OBS= data set option specifies an ending point for processing an input data set.

General form of OBS= data set option:

SAS-data-set(OBS=*n*)

This option specifies the number of the last observation to process, not how many observations should be processed.

Using the OBS= Data Set Option

This OBS= data set option causes the DATA step to stop processing after observation 100.

```
data australia;
   set orion.employee_addresses (obs=100);
   if Country='AU' then output;
run;
```

Partial SAS Log

```
NOTE: There were 100 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.AUSTRALIA has 24 observations and
      9 variables.
```

The FIRSTOBS= Data Set Option

The FIRSTOBS= data set option specifies a starting point for processing an input data set. This option specifies the number of the first observation to process.

General form of the FIRSTOBS= data set option:

SAS-data-set (FIRSTOBS=*n*)

FIRSTOBS= and OBS= are often used together to define a range of observations to be processed.

Using OBS= and FIRSTOBS= Data Set Options

The FIRSTOBS= and OBS= data set options cause the SET statement below to read 51 observations from **orion.employee_addresses**. Processing begins with observation 50 and ends after observation 100.

```
data australia;
   set orion.employee_addresses
       (firstobs=50 obs=100);
   if Country='AU' then output;
run;
```

Check the SAS Log

Partial SAS Log

```
640  data australia;
641       set orion.employee_addresses(firstobs=50 obs=100);
642       if Country='AU' then output;
643  run;

NOTE: There were 51 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
NOTE: The data set WORK.AUSTRALIA has 13 observations and
      9 variables.
```

Controlling Which Records Are Read

The FIRSTOBS= and OBS= options can be used in an INFILE statement when SAS reads from raw data files.

```
data employees;
  infile 'emps.dat' firstobs=11 obs=15;
  input @1 EmpID 8. @9 EmpName $40.
        @153 Country $2.;
run;
proc print data=employees;
run;
```

✎ The syntax is different. In an INFILE statement, the options are not enclosed in parentheses.

Check the Output

Partial SAS Log

```
45   data employees;
46     infile 'emps.dat' firstobs=11 obs=15;
47     input @1 EmpID 8. @9 EmpName $40. @153 Country $2.;
48   run;

NOTE: 5 records were read from the infile 'emps.dat'.
NOTE: The data set WORK.EMPLOYEES has 5 observations and
      3 variables.
```

PROC PRINT Output

Obs	EmpID	EmpName	Country
1	121017	Arizmendi, Gilbert	US
2	121062	Armant, Debra	US
3	121119	Armogida, Bruce	US
4	120812	Arruza, Fauver	US
5	120756	Asta, Wendy	US

Using OBS= and FIRSTOBS= in a PROC Step

The FIRSTOBS= and OBS= data set options can also be used in SAS procedures. The PROC PRINT step below begins processing at observation 10 and ends after observation 15.

```
proc print data=orion.employee_addresses
             (firstobs=10 obs=15);
   var Employee_Name City State Country;
run;
```

Check the Output

Partial SAS Log

```
417  proc print data=orion.employee_addresses
418              (firstobs=10 obs=15);
419     var Employee_Name City State Country;
420  run;

NOTE: There were 6 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
```

PROC PRINT output shows the original observation numbers.

PROC PRINT Output

Obs	Employee_Name	City	State	Country
10	Areu, Jeryl	Miami-Dade	Fl	US
11	Arizmendi, Gilbert	San Diego	CA	US
12	Armant, Debra	San Diego	CA	US
13	Armogida, Bruce	Philadelphia	PA	US
14	Arruza, Fauver	Miami-Dade	FL	US
15	Asta, Wendy	Philadelphia	PA	US

Adding a WHERE Statement

When the FIRSTOBS= or OBS= option and the WHERE statement are used together, the following occurs:

- the subsetting WHERE is applied first
- the FIRSTOBS= and OBS= options are applied to the resulting observations.

The following step includes a WHERE statement and an OBS= option.

```
proc print data=orion.employee_addresses
                (obs=10);
    where Country='AU';
    var Employee_Name City Country;
run;
```

Check the Output

Partial SAS Log

```
421  proc print data=orion.employee_addresses
422              (obs=10);
423     where Country='AU';
424     var Employee_Name City Country;
425  run;

NOTE: There were 10 observations read from the data set
      ORION.EMPLOYEE_ADDRESSES.
      WHERE Country='AU';
```

The WHERE statement is applied first, and then 10 observations are processed.

PROC PRINT Output

Obs	Employee_Name		
2	Aisbitt, Sandy	Melbourne	AU
17	Bahlman, Sharon	Sydney	AU
18	Baker, Gabriele	Sydney	AU
22	Baran, Shanmuganathan	Sydney	AU
23	Barbis, Viney	Sydney	AU
24	Barcoe, Selina	Melbourne	AU
25	Barreto, Geok-Seng	Sydney	AU
31	Billington, Kareen	Sydney	AU
34	Blanton, Brig	Melbourne	AU
37	Body, Meera	Sydney	AU

Chapter Review

1. What statement is used to request explicit output?

2. To what data set will it write?

3. How can multiple data sets be created in a DATA step?

4. If multiple data sets are being created, to which data set will the OUTPUT statement write?

continued...

Chapter Review

5. What data set option controls which variables are written to a data set?

6. When the KEEP= data set option is specified on an input data set, all variables are available for processing. True or False?

7. When FIRSTOBS= and OBS= options are used in the SET statement, what does the OBS= value indicate?