

# STAT604

## Lesson SAS 06

Portions Copyright © 2009 SAS Institute Inc., Cary, NC, USA. All rights reserved. Reproduced with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor.

**THE  
POWER  
TO KNOW®**

# Chapter 2: Controlling Input and Output



**2.1 Outputting Multiple Observations**

**2.2 Writing to Multiple SAS Data Sets**

**2.3 Selecting Variables and Observations**

# 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.

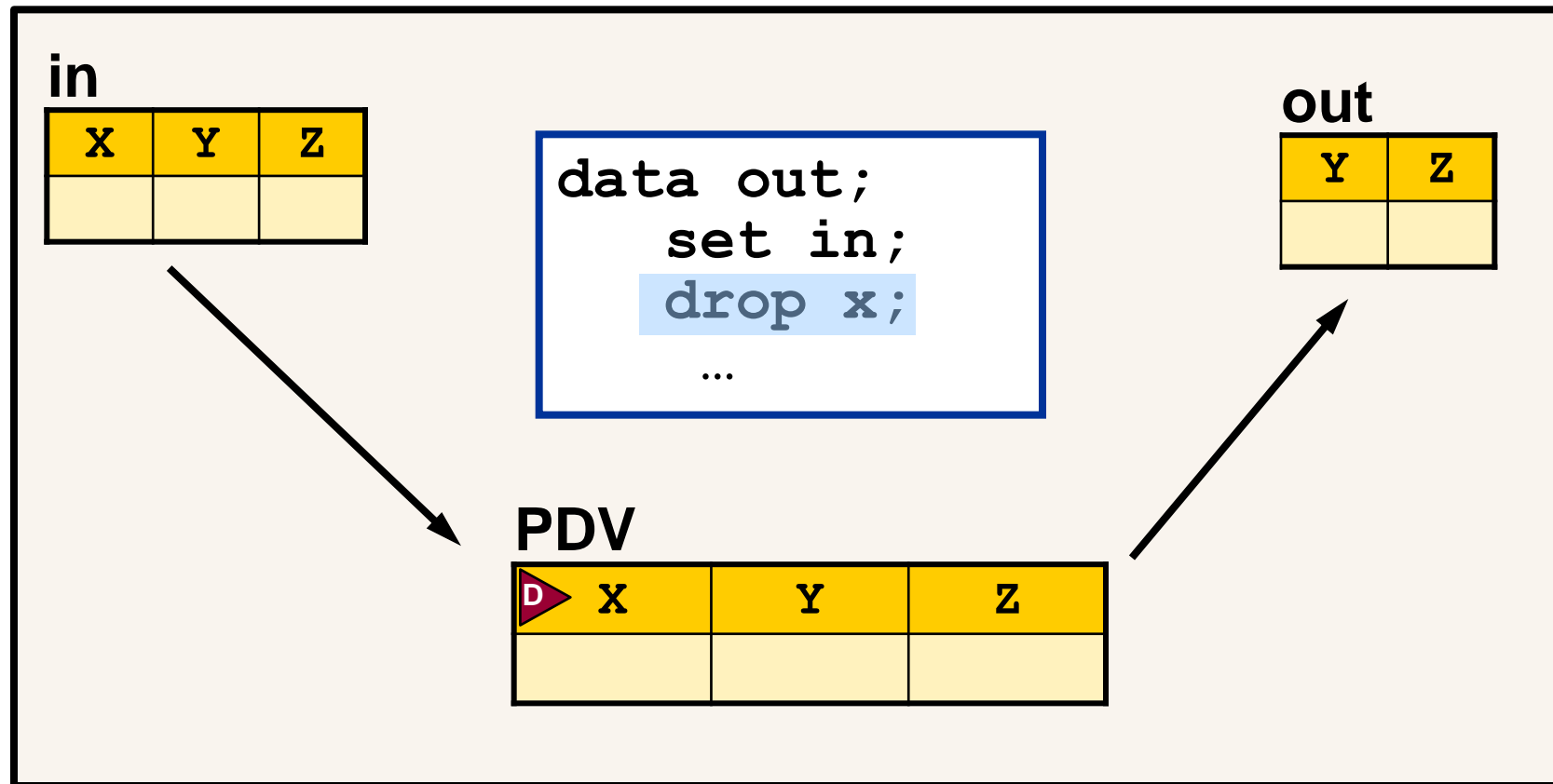
Name the data sets **usa**, **australia**, and **other**, and write different variables to each output data set.

# 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

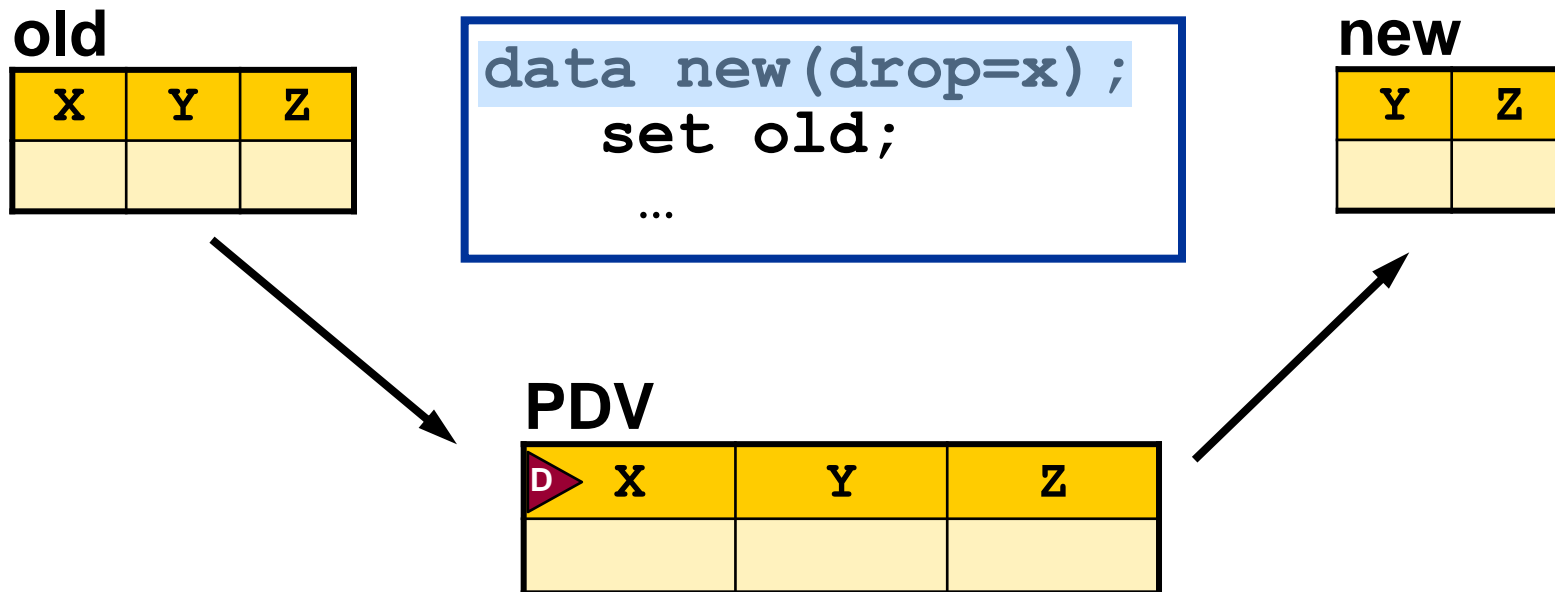
# The DROP= Data Set Option

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.

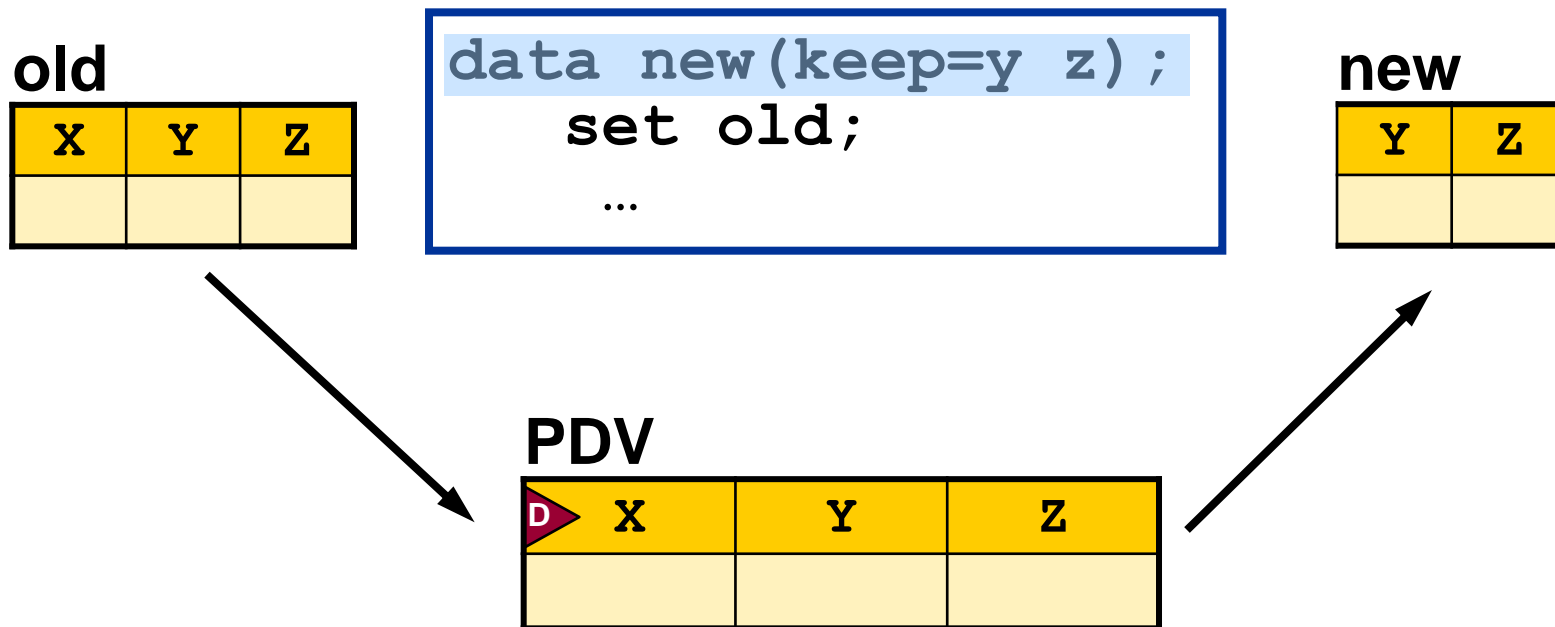
# The KEEP= Data Set Option

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. (Drop occurs first)

# Poll

# Quiz





## 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;
```

## 2.06 Quiz – Correct Answer

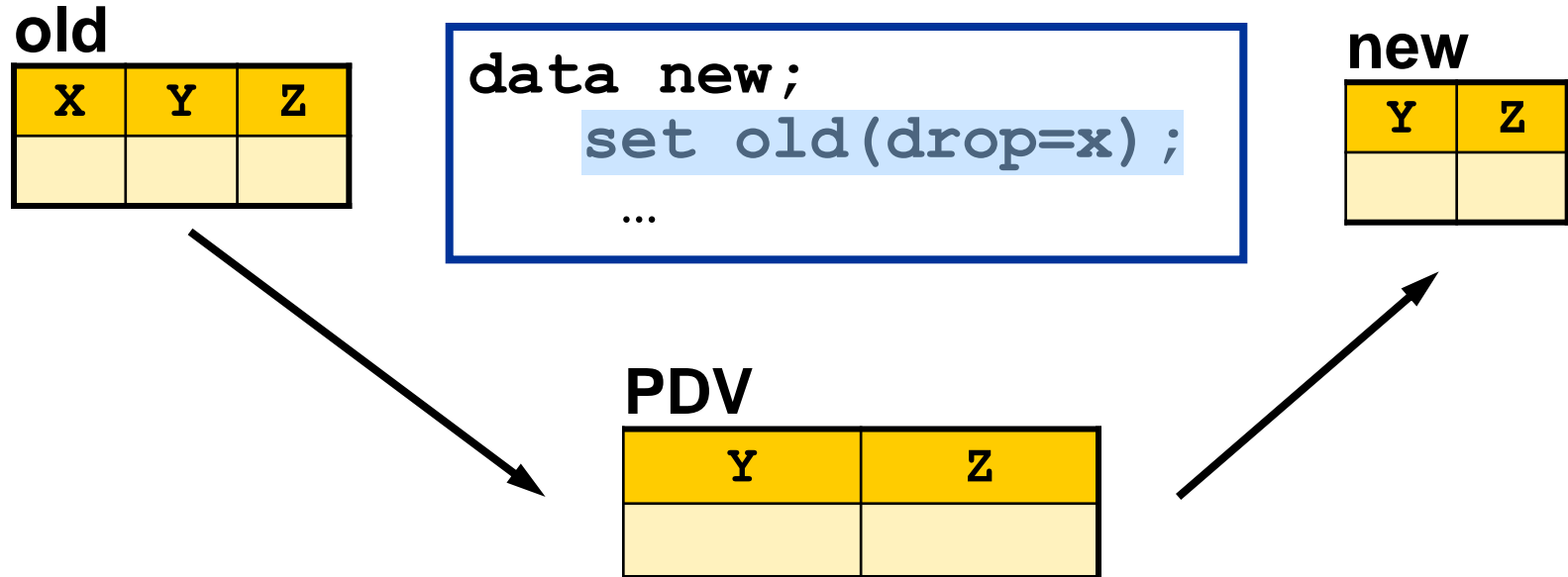
The data set **orion.employee\_addresses** contains nine variables. How many variables will be in the **usa**, **australia**, and **other** data sets? **4, 6, 9**

```
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;
```

Four variables are kept in **usa**, three are dropped from **australia**, and there is no DROP or KEEP for **other**.

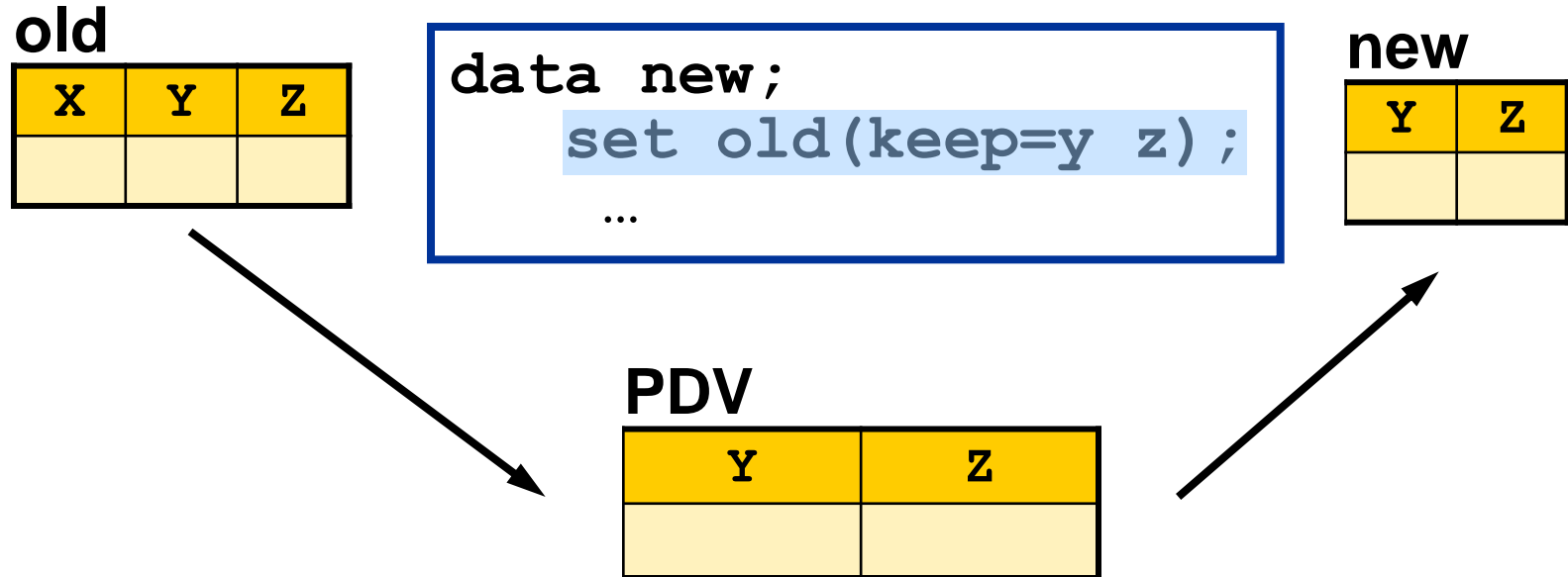
# Using DROP= on an Input Data Set

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**.



# Poll

# Quiz



## 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;
```

## 2.07 Quiz – Correct Answer

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;
```

**Country** is dropped on input, and therefore it is not available for processing. Every observation is written to **other**.

# 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;
```

**PDV**

City	Country	Employee _Name	Postal _Code	State	Street _Name	Street_ Number

State is only dropped  
from australia

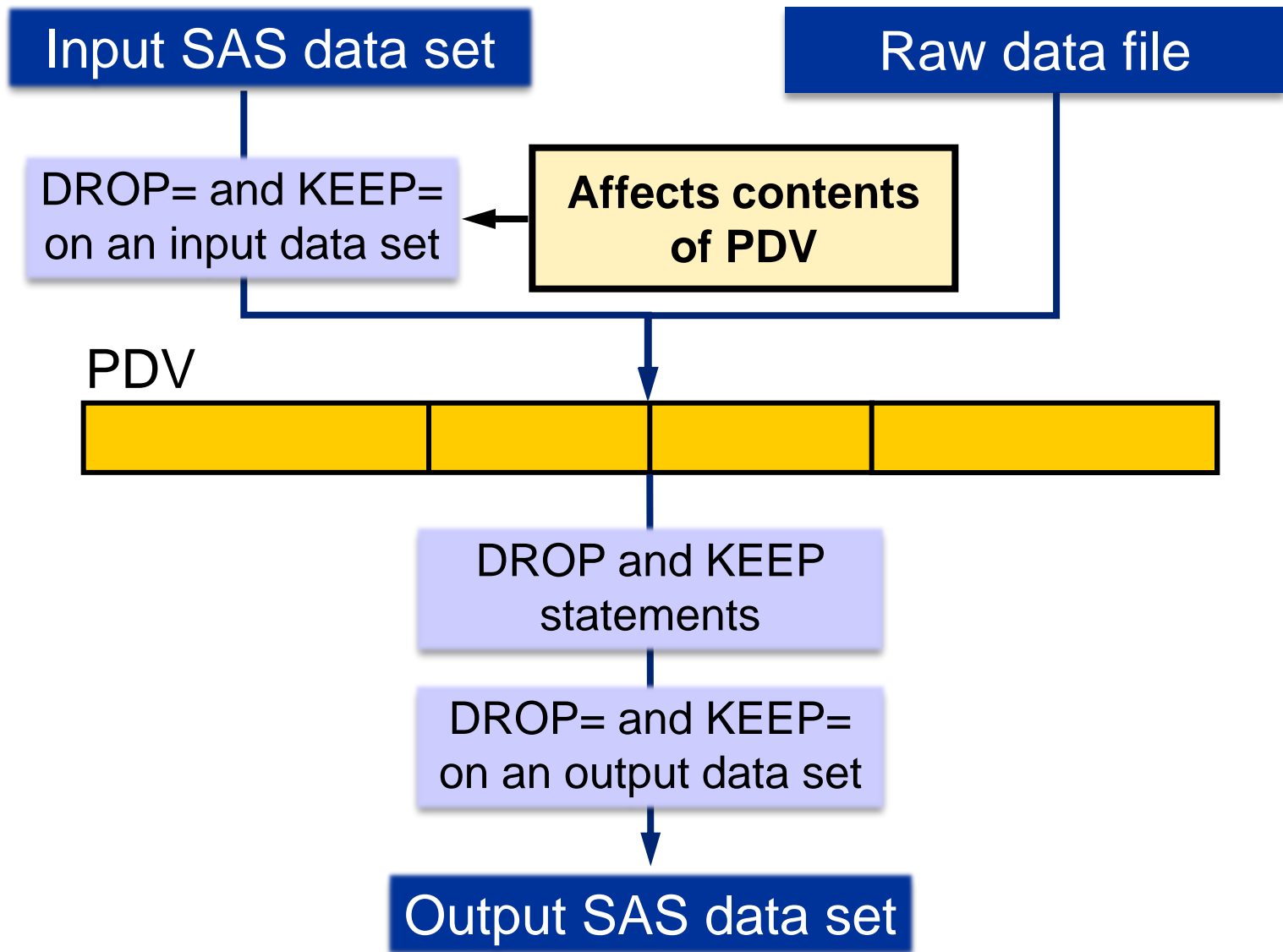


# Check the SAS Log

## 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	City	Country
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 5: Data Transformations



**5.1 Introduction**

**5.2 Manipulating Character Values (Part 1)**

**5.3 Manipulating Character Values (Part 2)**

**5.4 Manipulating Numeric Values**

**5.5 Converting Variable Type**

# Chapter 5: Data Transformations

## **5.1 Introduction**

## **5.2 Manipulating Character Values (Part 1)**

## **5.3 Manipulating Character Values (Part 2)**

## **5.4 Manipulating Numeric Values**

## **5.5 Converting Variable Type**



# Objectives

- Review the syntax of SAS functions.
- Introduce SAS variable lists.

# SAS Functions

SAS provides a large library of functions for manipulating data during DATA step execution.

A SAS function is often categorized by the type of data manipulation performed:

- Array
- Character
- Date And Time
- Descriptive Statistics
- Financial
- Mathematical
- Probability
- Random Number
- Trigonometric
- Special
- State and ZIP Code

# Syntax for SAS Functions

A *SAS function* is a routine that performs a computation and returns a value. Functions use arguments supplied by the user or by the operating environment.

General form of a SAS function call:

*function-name(argument-1,argument-2,...,argument-n)*

An *argument* can be a constant, a variable, or any SAS expression, including another function.

# Using SAS Functions

You can use functions in DATA step statements anywhere that an expression can appear.

```
data contrib;  
    set orion.employee_donations;  
    Total=sum(Qtr1,Qtr2,Qtr3,Qtr4);  
    if Total ge 50;  
run;  
  
proc print data=contrib noobs;  
    title 'Contributions $50 and Over';  
    var Employee_ID Qtr1 Qtr2 Qtr3 Qtr4  
        Total;  
run;
```

# Using SAS Functions

## Partial PROC PRINT Output

Contributions \$50 and Over					
Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	Total
120267	15	15	15	15	60
120269	20	20	20	20	80
120271	20	20	20	20	80
120275	15	15	15	15	60
120660	25	25	25	25	100
120669	15	15	15	15	60
120671	20	20	20	20	80

# SAS Variable Lists

An alternative to typing variables separately is to use a *SAS variable list*.

```
data contrib;  
    set orion.employee_donations;  
    Total=sum(of Qtr1-Qtr4) ;  
    if Total ge 50;  
run;
```

When you use a SAS variable list in a SAS function, use the keyword OF in front of the first variable name in the list.

# SAS Variable Lists

A SAS *variable list* is an abbreviated method of referring to a group of variable names. SAS enables you to use the following variable lists:

- Numbered range (X1-Xn)
- Name range (X--A, X-numeric-A, X-character-A)
- Name prefix (X:)
- Special SAS name lists (\_ALL\_, \_NUMERIC\_, \_CHARACTER\_)

# SAS Variable Lists – Examples

**PDV**

*Numbered Range List*

Qtr1	Qtr2	Var1	Qtr3	Qtr4

`Total = sum(of Qtr1-Qtr4)`

**Var1** not  
included

**PDV**

*Name Range List*

Qtr1	Second	Q3	Fourth	Var2

`Total = sum(of Qtr1--Fourth)`

**Var2** not  
included



# SAS Variable Lists – Examples

PDV

*Name Prefix List*

TotJan	Qtr2	TotFeb	TotMar

```
Total = sum(of Tot:)
```

Qtr2 not  
included

PDV

*Special Name Lists*

Qtr1	Name	Q3	Fourth
N	\$	N	N

```
Total = sum(of _Numeric_)
```

Name not  
included

# Poll

# Quiz



## 5.02 Quiz

Complete the assignment statement for **Total** by using a SAS variable list and the SUM function to add the values for **Year1**, **Year2**, **Year3**, and **Year4**.

**PDV**

Year2	Year1	Year3	Year4	Sales

**Total** =

?

## 5.02 Quiz – Correct Answer

Any of these assignment statements would give the correct value for **Total**.

**PDV**

Year2	Year1	Year3	Year4	Sales

```
Total = sum(of Year1-Year4) ;
```

```
Total = sum(of Year2--Year4) ;
```

```
Total = sum(of Year:) ;
```

# Chapter 5: Data Transformations

**5.1 Introduction**

**5.2 Manipulating Character Values (Part 1)**

**5.3 Manipulating Character Values (Part 2)**

**5.4 Manipulating Numeric Values**

**5.5 Converting Variable Type**

# Objectives

- Use SAS functions to extract, edit, and search character values.

# Business Scenario – Create a List of Charities

A manager in the Finance department asked for a list of all the charities that Orion Star contributes to. She would like to see the name of the charity as well as the ID code assigned to it.

Here is a sketch of the desired output:

Charity Names and ID Codes	
ID	Name
AQI	Aquamissions International
CCI	Cancer Cures, Inc.
CNI	Conserve Nature, Inc.

# Input Data

The **orion.biz\_list** data set is extracted from the accounting system and contains the names of Orion Star's U.S. suppliers, charities, and consultants.

## Partial Listing of **orion.biz\_list**

Acct_ Code	Name
AEK3	ANGELA E. KEARNEY
AQI2	AQUAMISSIONS INTERNATIONAL
ATS1	A TEAM SPORTS
CB03	CLAIRE B. OWENS
CCI2	CANCER CURES, INC.
CNI2	CONSERVE NATURE, INC.
CS1	CAROLINA SPORTS



# Input Data – Details

**Acct\_Code** is a character variable defined as length 6. Its last digit represents the type of organization: 1 denotes a supplier, 2 a charity, and 3 a consultant.

The other characters in the **Acct\_Code** variable represent the ID for the organization, so the **ID** value can have as many as five characters.

Example:

Acct_Code	ID
\$6	\$5
AQI2	AQI

- 2 denotes a charity.
- AQI is the ID.

# Input Data – Details

The name of the organization is stored as all capital letters. In the desired output, only the first letter of each word is capitalized.

Example:

Name	
AQUAMISSIONS	INTERNATIONAL

Change to:

Name	
Aquamissions	International

# Business Scenario – Desired Results

Create a new data set, **charities**, that has the information that the finance manager would like to see.

## Partial Listing of **charities**

ID	Acct_ Code	Name
AQI	AQI2	Aquamissions International
CCI	CCI2	Cancer Cures, Inc.
CNI	CNI2	Conserve Nature, Inc.
CS	CS2	Child Survivors
CU	CU2	Cuidadores Ltd.
DAI	DAI2	Disaster Assist, Inc.

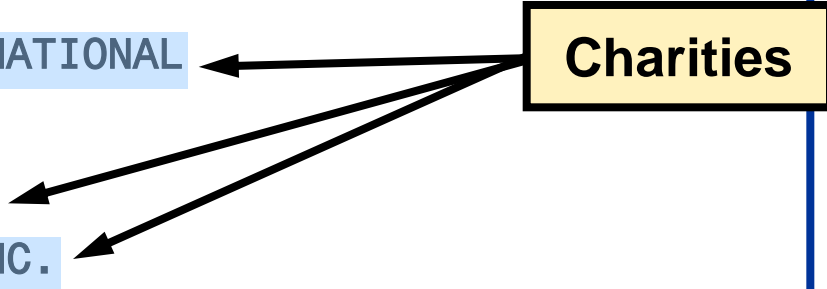
This data set can then be used to create the manager's report.

# Create the List of Charities – Step 1

The first step is to subset the data based on the last character of **Acct\_Code**.

Partial Listing of **orion.biz\_list**

Acct_ Code	Name
AEK3	ANGELA E. KEARNEY
AQI2	AQUAMISSIONS INTERNATIONAL
ATS1	A TEAM SPORTS
CB03	CLAIRE B. OWENS
CCI2	CANCER CURES, INC.
CNI2	CONSERVE NATURE, INC.
CS1	CAROLINA SPORTS



A yellow box labeled "Charities" is positioned to the right of the table. Three arrows point from this box to the rows corresponding to "AQUAMISSIONS INTERNATIONAL", "CANCER CURES, INC.", and "CONSERVE NATURE, INC.". The names in these rows are highlighted in light blue.

SAS character functions make this task easy.

# The SUBSTR Function (Right Side)

The SUBSTR function on the right side of an assignment statement is used to extract characters.

General form of the SUBSTR function:

```
NewVar=SUBSTR(string,start<,length>);
```

<i>string</i>	can be a character constant, variable, or expression.
<i>start</i>	specifies the starting position.
<i>length</i>	specifies the number of characters to extract. If omitted, the substring consists of the remainder of string.
<i>NewVar</i>	If <i>NewVar</i> is a new variable it will be created with the same length as <i>string</i> . To set a different length for <i>NewVar</i> , use a LENGTH statement prior to the assignment statement.

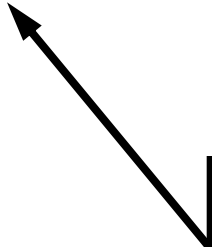
# The SUBSTR Function – Example

Extract the first three characters from the value in the **Item\_Code** variable and store them in **Item\_Type**.

```
Item_Type=substr(Item_Code,1,3);
```

**PDV**

Item_Code \$ 20	Item_Type \$ 20
978-1-59994-397-8	978



Starting at position 1  
for a length of 3

Poll 

Quiz

# Setup for the Poll

This is the current value of **Item\_Code**:

**PDV**

Item_Code	
\$ 20	
978-1-59994	-397-8

The SUBSTR function is a good method to extract the highlighted digits.



## 5.03 Multiple Choice Poll

Which SUBSTR function can extract the group of five numbers from the middle of the **Item\_Code** value?

- a. `substr(Item_Code, 5, 7)`
- b. `substr(Item_Code, 5)`
- c. `substr(Item_Code, 7, 5)`
- d. `substr(Item_Code, 'mid', 5)`

**PDV**

Item_Code
\$ 20
978-1-59994-397-8

## 5.03 Multiple Choice Poll – Correct Answer

Which SUBSTR function can extract the group of five numbers from the middle of the **Item\_Code** value?

- a. `substr(Item_Code, 5, 7)`
- b. `substr(Item_Code, 5)`
- ☒ c. `substr(Item_Code, 7, 5)`
- d. `substr(Item_Code, 'mid', 5)`

# Create the List of Charities – Step 1

The last non-blank character in the **Acct\_Code** value occurs in different positions for different observations.

Partial Listing of  
**orion.biz\_list**

Acct\_  
Code

AEK3

AQI2

ATS1

CB03

CCI2

CNI2

CS1

CS2

CU2

Last character in position 4

Last character in position 3

You need some way to determine the position of the last character so that the SUBSTR function can extract it.

# The LENGTH Function

The LENGTH function returns the length of a non-blank character string, excluding trailing blanks.

General form of the LENGTH function:

```
NewVar=LENGTH(argument);
```

Example:

```
Code = 'ABCD  ' ;  
Last_NonBlank=length (Code) ;
```

## PDV

Code \$ 6	Last_NonBlank N 8
ABCD	4

# Create the List of Charities – Step 1

This program uses the SUBSTR and LENGTH functions to create the **charities** data set.

The LENGTH function is *nested*, or used as an argument to the SUBSTR function.

```
data charities;  
  length ID $ 5;  
  set orion.biz_list;  
  if substr(Acct_Code, length(Acct_Code), 1) = '2' ;  
  ID = substr(Acct_Code, 1, length(Acct_Code) - 1) ;  
run;
```

Partially stepping through the execution for the first charity observation shows how the functions transform the data.

# Execution: Step 1

Read the first  
charity observation.

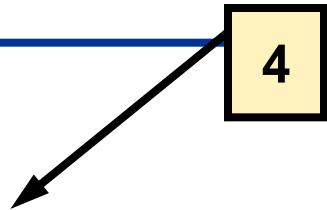
```
data charities;  
  length ID $ 5;  
  set orion.biz_list;  
  if substr(Acct_Code,length(Acct_Code),1)='2';  
  ID=substr(Acct_Code,1,length(Acct_Code)-1);  
run;
```

## PDV

ID \$ 5	Acct_Code \$ 6	Name \$ 30
	AQI2	AQUAMISSIONS INTERNATIONAL

# Execution: Step 1

```
data charities;  
  length ID $ 5;  
  set orion.biz_list;  
  if substr(Acct_Code, length(Acct_Code), 1) = '2' ;  
  ID=substr(Acct_Code, 1, length(Acct_Code) - 1) ;  
run;
```



## PDV

ID \$ 5	Acct_Code \$ 6	Name \$ 30
	AQI2	AQUAMISSIONS INTERNATIONAL

# Execution: Step 1

```
data charities;  
  length ID $ 5;  
  set orion.biz_list;  
  if substr(Acct_Code,length(Acct_Code),1)='2';  
  ID=substr(Acct_Code,1,length(Acct_Code)-1);  
run;
```

## PDV

ID \$ 5	Acct_Code \$ 6	Name \$ 30
	AQI2	AQUAMISSIONS INTERNATIONAL



# Execution: Step 1

```
data charities;  
  length ID $ 5;  
  set orion.biz_list;  
  if substr(Acct_Code,length(Acct_Code),1)='2';  
  ID=substr(Acct_Code,1,length(Acct_Code)-1);  
run;
```

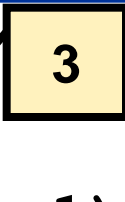
True

## PDV

ID \$ 5	Acct_Code \$ 6	Name \$ 30
	AQI2	AQUAMISSIONS INTERNATIONAL

# Execution: Step 1

```
data charities;  
  length ID $ 5;  
  set orion.biz_list;  
  if substr(Acct_Code,length(Acct_Code),1)='2';  
  ID=substr(Acct_Code,1,length(Acct_Code)-1);  
run;
```



## PDV

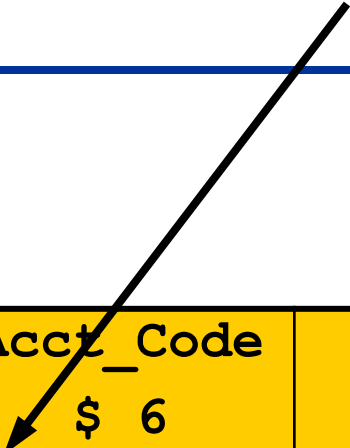
ID \$ 5	Acct_Code \$ 6	Name \$ 30
	AQI2	AQUAMISSIONS INTERNATIONAL

# Execution: Step 1

```
data charities;  
  length ID $ 5;  
  set orion.biz_list;  
  if substr(Acct_Code,length(Acct_Code),1)='2';  
  ID=substr(Acct_Code,1,length(Acct_Code)-1);  
run;
```

## PDV

ID \$ 5	Acct_Code \$ 6	Name \$ 30
	AQI2	AQUAMISSIONS INTERNATIONAL



# Execution: Step 1

```
data charities;  
  length ID $ 5;  
  set orion.biz_list;  
  if substr(Acct_Code,length(Acct_Code),1)='2';  
  ID=substr(Acct_Code,1,length(Acct_Code)-1);  
run;
```

## PDV

ID \$ 5	Acct_Code \$ 6	Name \$ 30
AQI	AQI2	AQUAMISSIONS INTERNATIONAL

# Execution: Step 1

```
data charities;  
  length ID $ 5;  
  set orion.biz_list;  
  if substr(Acct_Code,length(Acct_Code),1)='2';  
  ID=substr(Acct_Code,1,length(Acct_Code)-1);  
run;
```

**Implicit OUTPUT;  
Implicit RETURN;**

## PDV

ID \$ 5	Acct_Code \$ 6	Name \$ 30
AQI	AQI2	AQUAMISSIONS INTERNATIONAL

# Create the List of Charities – Step 1 Complete

## Listing of **charities**

ID	Acct_ Code	Name
AQI	AQI2	AQUAMISSIONS INTERNATIONAL
CCI	CCI2	CANCER CURES, INC.
CNI	CNI2	CONSERVE NATURE, INC.
CS	CS2	CHILD SURVIVORS
CU	CU2	CUIDADORES LTD.
DAI	DAI2	DISASTER ASSIST, INC.
ES	ES2	EARTHSALVORS
FFC	FFC2	FARMING FOR COMMUNITIES
MI	MI2	MITLEID INTERNATIONAL
SBA	SBA2	SAVE THE BABY ANIMALS
V2	V22	VOX VICTIMAS
YYCR	YYCR2	YES, YOU CAN RECYCLE

Step 2 is to transform the values in **Name** to a mix of uppercase and lowercase.

# The PROPCASE Function

The PROPCASE function converts all words in an argument to *proper case*, in which the first letter is uppercase and the remaining letters are lowercase.

General form for the PROPCASE function:

```
NewVar=PROPCASE(argument <,delimiter(s)>);
```

<i>argument</i>	can be a character constant, variable, or expression.
<i>delimiter(s)</i>	delimiters are characters which separate words. If omitted, the default delimiters are the blank, /, - , ( , ., and tab characters.
<i>NewVar</i>	If <i>NewVar</i> is a new variable, it is created with the same length as <i>argument</i> .

# The PROPCASE Function

Example:

```
Name = 'SURF&LINK SPORTS';  
Pname = propcase(Name);  
Pname2 = propcase(Name, ' & ');
```

## PDV

Name	Pname
\$ 16	\$ 16
SURF&LINK SPORTS	Surf&link Sports

Pname2
\$ 16
Surf&Link Sports



# Poll

# Quiz



## 5.04 Quiz

This PDV shows the current value of **Name**:

Name
HEATH*BARR*LITTLE EQUIPMENT SALES

Write an assignment statement that converts the value of **Name** to this:

Name
Heath*Barr*Little Equipment Sales

## 5.04 Quiz – Correct Answer

This PDV shows the current value of **Name**:

Name
HEATH*BARR*LITTLE EQUIPMENT SALES

Write an assignment statement that will convert the value of **Name** to this:

Name
Heath*Barr*Little Equipment Sales

```
Name = propcase (Name, ' *' ) ;
```

The second argument to the PROPCASE function must list all the characters to use as delimiters. In this example, the space and \* both need to be listed.

## Create the List of Charities – Step 2

Adding an assignment statement to convert **Name** to proper case completes the **charities** data set.

```
data charities;  
  length ID $ 5;  
  set orion.biz_list;  
  if substr(Acct_Code,length(Acct_Code),1)='2';  
  ID=substr(Acct_Code,1,length(Acct_Code)-1);  
  Name = propcase(Name);  
run;
```

# Create the List of Charities – Complete

## Listing of **charities**

ID	Acct_ Code	Name
AQI	AQI2	Aquamissions International
CCI	CCI2	Cancer Cures, Inc.
CNI	CNI2	Conserve Nature, Inc.
CS	CS2	Child Survivors
CU	CU2	Cuidadores Ltd.
DAI	DAI2	Disaster Assist, Inc.
ES	ES2	Earthsaviors
FFC	FFC2	Farming For Communities
MI	MI2	Mitleid International
SBA	SBA2	Save The Baby Animals
V2	V22	Vox Victimias
YYCR	YYCR2	Yes, You Can Recycle

# Other Useful Character Functions

Function	Purpose
RIGHT( <i>string</i> )	right-aligns a character expression.
LEFT( <i>string</i> )	left-aligns a character expression.
UPCASE( <i>string</i> )	converts all letters in an argument to uppercase.
LOWCASE( <i>string</i> )	converts all letters in an argument to lowercase.
CHAR( <i>string,position</i> )	returns a single character from a specified <i>position</i> in a character <i>string</i> .

Poll 

Quiz

## 5.05 Quiz

Find the syntax error in the code below. Product\_Name has a length of 45.

Partial listing of product\_list:

Product_ID	Product_Name	Supplier_ID	Product_Level	Product_Ref_ID
220100700023	Armadillo Road Dmx Men's Running Shoes	16733	1	220100700000
220100700024	Armadillo Road Dmx Women's Running Shoes	16733	1	220100700000
220100700046	Tcp 6 Men's Running Shoes	16733	1	220100700000

```
data shoes;  
    set orion.product_list;  
    if substr(right(Product_Name,33,13))=  
        'Running Shoes';  
run;
```



## 5.05 Quiz – Correct Answer

Misplaced parentheses are some of the most common syntax errors with functions.

Corrected program:

```
data shoes;  
  set orion.product_list;  
  if substr(right(Product_Name),33,13)=  
    'Running Shoes';  
run;
```

Correctly placed





# **Question & Answer**