STAT604

Lesson SAS 10

Assigning Initial Values to an ARRAY

The ARRAY statement has an option to assign initial values to the array elements.

General form of an ARRAY statement:

ARRAY *array-name* {*subscript*} <$> <*length*>
<*array-elements*> <(*initial-value-list*)>**;**

Example:

```
array Target{5} (50,100,125,150,200);
```

Use commas or spaces to separate values in the list.

Assigning Initial Values to an ARRAY

When an *initial-value-list* is specified, all elements behave as if they were named in a RETAIN statement. This is often used to create a *lookup table*, that is, a list of values to refer to during DATA step processing.

```
array Target{5} (50,100,125,150,200);
```

PDV

Target1	Target2	Target3	Target4	Target5
R N 8	R N 8	R N 8	R N 8	R N 8
50	100	125	150	200

Business Scenario

Read **orion.employee_donations** to determine the difference between employee contributions and the quarterly goals of $10, $20, $20, and $15. Use a lookup table to store the quarterly goals.

```
data compare(drop=i Goal1-Goal4);
   set orion.employee_donations;
   array Contrib{4} Qtr1-Qtr4;
   array Diff{4};
   array Goal{4} (10,20,20,15);
   do i=1 to 4;
      Diff{i}=Contrib{i}-Goal{i};
   end;
run;
```

p207d16

Compilation: What Variables Are Created?

```
data compare(drop=i Goal1-Goal4);
   set orion.employee_donations;
   array Contrib{4} Qtr1-Qtr4;
   array Diff{4};
   array Goal{4} (10,20,20,15);
   do i=1 to 4;
      Diff{i}=Contrib{i}-Goal{i};
   end;
run;
```

Partial PDV

Employee_ ID	Qtr1	Qtr2	Qtr3	Qtr4

...

Compilation: What Variables Are Created?

```
data compare(drop=i Goal1-Goal4);
   set orion.employee_donations;
   array Contrib{4} Qtr1-Qtr4;
   array Diff{4};
   array Goal{4} (10,20,20,15);
   do i=1 to 4;
      Diff{i}=Contrib{i}-Goal{i};
   end;
run;
```

No variables created

Partial PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4

...

Compilation: What Variables Are Created?

```
data compare(drop=i Goal1-Goal4);
   set orion.employee_donations;
   array Contrib{4} Qtr1-Qtr4;
   array Diff{4};
   array Goal{4} (10,20,20,15);
   do i=1 to 4;
      Diff{i}=Contrib{i}-Goal{i};
   end;
run;
```

Partial PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	Diff1

Diff2	Diff3	Diff4

...

Compilation: What Variables Are Created?

```
data compare(drop=i Goal1-Goal4);
   set orion.employee_donations;
   array Contrib{4} Qtr1-Qtr4;
   array Diff{4};
   array Goal{4} (10,20,20,15);
   do i=1 to 4;
      Diff{i}=Contrib{i}-Goal{i};
   end;
run;
```

Partial PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	Diff1

Diff2	Diff3	Diff4	Goal1	Goal2	Goal3	Goal4

...

Compilation: What Variables Are Created?

```
data compare(drop=i Goal1-Goal4);
   set orion.employee_donations;
   array Contrib{4} Qtr1-Qtr4;
   array Diff{4};
   array Goal{4} (10,20,20,15);
   do i=1 to 4;
      Diff{i}=Contrib{i}-Goal{i};
   end;
run;
```

Partial PDV

Employee_ ID	Qtr1	Qtr2	Qtr3	Qtr4	Diff1

Diff2	Diff3	Diff4	Goal1	Goal2	Goal3	Goal4	i

...

Compilation: Drop Flags Are Set

```
data compare (drop=i Goal1-Goal4);
   set orion.employee_donations;
   array Contrib{4} Qtr1-Qtr4;
   array Diff{4};
   array Goal{4} (10,20,20,15);
   do i=1 to 4;
      Diff{i}=Contrib{i}-Goal{i};
   end;
run;
```

Partial PDV

Employee_ ID	Qtr1	Qtr2	Qtr3	Qtr4	Diff1

Diff2	Diff3	Diff4	D ▶ Goal1	D ▶ Goal2	D ▶ Goal3	D ▶ Goal4	D ▶ i

...

Compilation: Retain Flags Are Set

```
data compare(drop=i Goal1-Goal4);
   set orion.employee_donations;
   array Contrib{4} Qtr1-Qtr4;
   array Diff{4};
   array Goal{4} (10,20,20,15);
   do i=1 to 4;
      Diff{i}=Contrib{i}-Goal{i};
   end;
run;
```

Partial PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	Diff1

Diff2	Diff3	Diff4	D R Goal1	D R Goal2	D R Goal3	D R Goal4	D i

...

PDV Is Initialized

```
data compare(drop=i Goal1-Goal4);
   set orion.employee_donations;
   array Contrib{4} Qtr1-Qtr4;
   array Diff{4};
   array Goal{4} (10,20,20,15);
   do i=1 to 4;
      Diff{i}=Contrib{i}-Goal{i};
   end;
run;
```

Initialize PDV

Partial PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	Diff1
.	.	.	.	.	.

Diff2	Diff3	Diff4	D R Goal1	D R Goal2	D R Goal3	D R Goal4	D i
.	.	.	10	20	20	15	.

Creating a Temporary Lookup Table

You can use the keyword _TEMPORARY_ in an ARRAY statement to indicate that the elements are not needed in the output data set.

```
data compare(drop=i);
   set orion.employee_donations;
   array Contrib{4} Qtr1-Qtr4;
   array Diff{4};
   array Goal{4} _temporary_ (10,20,20,15);
   do i=1 to 4;
      Diff{i}=Contrib{i}-Goal{i};
   end;
run;
```

p207d17

Output: Creating a Temporary Lookup Table

```
proc print data=compare noobs;
   var employee_id diff1-diff4;
run;
```

Partial PROC PRINT Output

```
  Employee_ID    Diff1    Diff2    Diff3    Diff4

     120265         .        .        .       10
     120267         5       -5       -5        0
     120269        10        0        0        5
     120270        10      -10      -15        .
     120271        10        0        0        5
     120272         0      -10      -10       -5
     120275         5       -5       -5        0
```

What can be done to ignore missing values?

The SUM Function Ignores Missing Values

The SUM function ignores missing values. It can be used to calculate the difference between the quarterly contribution and the corresponding goal.

```
data compare(drop=i);
   set orion.employee_donations;
   array Contrib{4} Qtr1-Qtr4;
   array Diff{4};
   array Goal{4} _temporary_ (10,20,20,15);
   do i=1 to 4;
      Diff{i}=sum(Contrib{i},-Goal{i});
   end;
run;
```

p207d18

Output: Lookup Table Application

```
proc print data=compare noobs;
   var employee_id diff1-diff4;
run;
```

Partial PROC PRINT Output

```
 Employee_ID    Diff1    Diff2    Diff3    Diff4

    120265       -10      -20      -20       10
    120267         5       -5       -5        0
    120269        10        0        0        5
    120270        10      -10      -15      -15
    120271        10        0        0        5
    120272         0      -10      -10       -5
    120275         5       -5       -5        0
```

The missing values were handled as if no contribution were made for that quarter.

Poll
Quiz

7.09 Quiz

Write an ARRAY statement to define a temporary lookup table named **`Country`** with three elements, each two characters long. Initialize the elements to AU, NZ, and US. Refer to the syntax below.

ARRAY *array-name* {*subscript*} <$> <*length*>
<*array-elements*> <(*initial-value-list*)>**;**

7.09 Quiz – Correct Answer

Write an ARRAY statement to define a temporary lookup table named **`Country`** with three elements, each two characters long. Initialize the elements to AU, NZ, and US. Refer to the syntax below.

ARRAY *array-name* {*subscript*} <$> <*length*>
<*array-elements*> <(*initial-value-list*)>**;**

```
array Country{3} $ 2 _temporary_ ('AU','NZ','US');
```

Chapter 8: Restructuring a Data Set

8.1 Rotating with the DATA Step

8.2 Using the TRANSPOSE Procedure

Chapter 8: Restructuring a Data Set

8.1 Rotating with the DATA Step

8.2 Using the TRANSPOSE Procedure

Objectives

- Use a DATA step with arrays and DO loop processing to restructure a data set.

Data Set Structure

Some data sets store all the information about one entity in a single observation. This data set structure is useful for data mining and generating reports. For convenience this is referred to as a *wide* data set.

```
Customer_ID  Qtr1    Qtr2     Qtr3    Qtr4  Method
  134391       .      125       .       .   Cash
  143561     150       79      67      15   Credit
  158913     208       22       .      33   Credit
```

All information for Customer 143561 is in a single observation.

Data Set Structure

Other data sets have multiple observations per entity. Each observation typically contains a small amount of data, and missing values might or might not be stored. For convenience, this is referred to as a *narrow* data set.

```
Customer_ID  Period    Amount
     134391  Qtr2         125
     143561  Qtr1         150
     143561  Qtr2          79
     143561  Qtr3          67
     143561  Qtr4          15
     158913  Qtr1         208
     158913  Qtr2          22
```

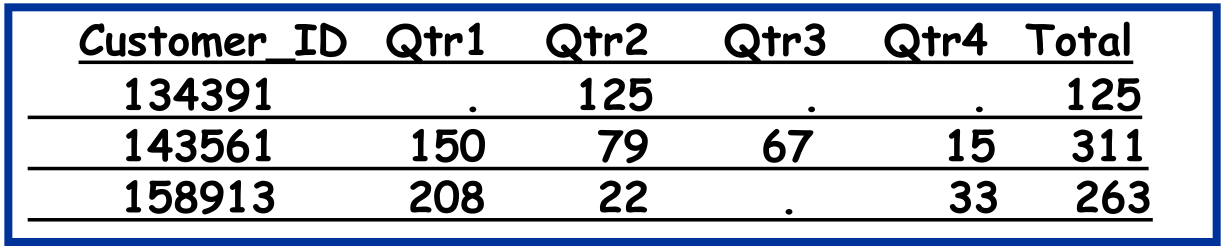
✎ The information for Customer 143561 is stored in four observations.

Why Restructure a Data Set?

Before writing a program, you need to consider the data available, the desired output, and the processing required. Sometimes restructuring the data can simplify a task.

The Sales Manager requested the following report showing customer information:

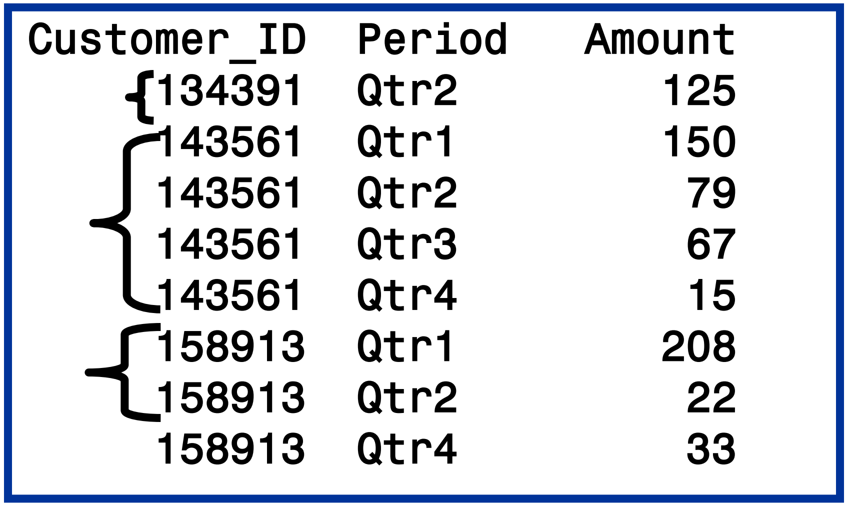
Sketch of desired report:



Customer_ID	Qtr1	Qtr2	Qtr3	Qtr4	Total
134391	.	125	.	.	125
143561	150	79	67	15	311
158913	208	22	.	33	263

Why Restructure a Data Set?

You explore the available data and locate the following data set:



```
Customer_ID  Period    Amount
     134391  Qtr2         125
     143561  Qtr1         150
     143561  Qtr2          79
     143561  Qtr3          67
     143561  Qtr4          15
     158913  Qtr1         208
     158913  Qtr2          22
     158913  Qtr4          33
```

This data set has the required data, but the current structure would require a DATA step with First. and Last. processing.

Why Restructure a Data Set?

If the data set were in this form, a simple assignment statement is all that would be needed to create the new variable, **`Total`**.

```
  Customer_ID  Qtr1    Qtr2     Qtr3    Qtr4
     134391       .     125        .       .
     143561     150      79       67      15
     158913     208      22        .      33
```

Restructuring from a narrow to a wide data set simplifies the processing.

Poll

Quiz

8.01 Quiz

Which data set structure is more appropriate for using PROC FREQ to determine the number of charitable donations made in each of the four quarters (**`Qtr1–Qtr4`**)?

a.

```
   Employee_ID  Qtr1     Qtr2      Qtr3     Qtr4
     120265       .        .         .       25
     120267      15       15        15       15
     120269      20       20        20       20
```

b.

```
Employee_ID  Period    Amount
     120265  Qtr4          25
     120267  Qtr1          15
     120267  Qtr2          15
     120267  Qtr3          15
     120267  Qtr4          15
```

8.01 Quiz – Correct Answer

Which data set structure is more appropriate for using PROC FREQ to determine the number of charitable donations made in each of the 4 quarters (**`Qtr1`**– **`Qtr4`**)?

Proposed SAS program

```
proc freq data=b;
   tables Period /nocum nopct;
run;
```

[✓ b.]

Employee_ID	Period	Amount
120265	Qtr4	25
120267	Qtr1	15
120267	Qtr2	15
120267	Qtr3	15
120267	Qtr4	15
120269	Qtr1	20
120269	Qtr2	20

PROC FREQ Output

```
      The FREQ Procedure

   Period          Frequency
   ─────────────────────────
   Qtr1                    2
   Qtr2                    2
   Qtr3                    1
   Qtr4                    2
```

Business Scenario – A Frequency Report

The Orion Payroll Manager asked for a report showing the number of Orion Star employees who made charitable donations in each quarter.

Sketch of the Desired Report

Period	Frequency
Qtr1	56
Qtr2	99
Qtr3	24
Qtr4	75

The FREQ procedure can be used to generate the desired report.

Business Scenario Considerations

The **`orion.employee_donations`** data set contains the needed information, but is not in the form to be easily analyzed using the FREQ procedure.

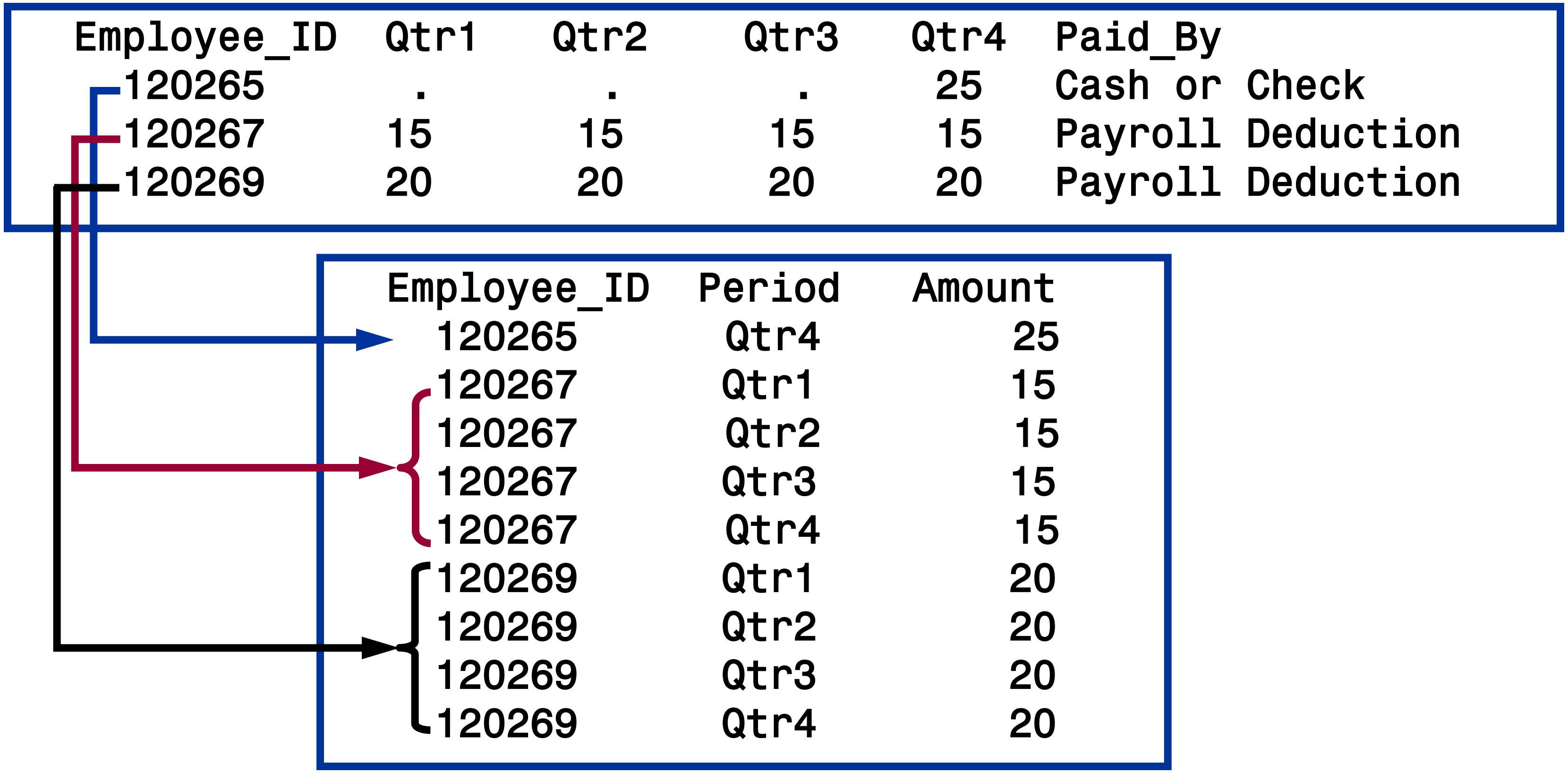
Partial Listing of **`orion.employee_donations`**

```
  Employee_ID   Qtr1     Qtr2      Qtr3     Qtr4   Paid_By
    120265        .        .         .       25    Cash or Check
    120267       15       15        15       15    Payroll Deduction
    120269       20       20        20       20    Payroll Deduction
```

✎ Changing the data set from a wide to a narrow structure can simplify this task.

Business Scenario Considerations

Restructure the input data set, and create a separate observation for each nonmissing quarterly contribution. The output data set, **rotate**, should contain only **Employee_ID**, **Period**, and **Amount**.



```
Employee_ID   Qtr1    Qtr2    Qtr3    Qtr4   Paid_By
  120265        .       .       .      25    Cash or Check
  120267       15      15      15      15    Payroll Deduction
  120269       20      20      20      20    Payroll Deduction
```

```
Employee_ID   Period    Amount
   120265      Qtr4        25
   120267      Qtr1        15
   120267      Qtr2        15
   120267      Qtr3        15
   120267      Qtr4        15
   120269      Qtr1        20
   120269      Qtr2        20
   120269      Qtr3        20
   120269      Qtr4        20
```

Rotating a SAS Data Set

The DATA step below rotates the input data set. An output observation will be written if a contribution was made in a given quarter.

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
      if contrib{i} ne . then do;
         Period=cats("Qtr",i);
         Amount=contrib{i};
         output;
      end;
   end;
run;
```

Only include nonmissing values

Compilation: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount

work.rotate

Employee_ID	Period	Amount

p208d01

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

Initialize PDV

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
.	.	.	.	.	.		.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	.		.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
      if contrib{i} ne . then do;
         Period=cats("Qtr",i);
         Amount=contrib{i};
         output;
      end;
   end;
run;
```

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	1		.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

False

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	1		.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

i+1

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	2		.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	2		.

work.rotate

Employee_ID	Period	Amount

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

False

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	2		.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

i+1

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	3		.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
             (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	3		.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

False

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	3		.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

i+1

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	4		.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
      if contrib{i} ne . then do;
         Period=cats("Qtr",i);
         Amount=contrib{i};
         output;
      end;
   end;
run;
```

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	4		.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

True

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	4		.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	4	Qtr4	.

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	4	Qtr4	25

work.rotate

Employee_ID	Period	Amount

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

Output current observation

PDV

Employee_ID	~~Qtr1~~	~~Qtr2~~	~~Qtr3~~	~~Qtr4~~	~~I~~	Period	Amount
120265	~~.~~	~~.~~	~~.~~	~~25~~	~~4~~	Qtr4	25

work.rotate

Employee_ID	Period	Amount
120265	Qtr4	25

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

i+1

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	5	Qtr4	25

work.rotate

Employee_ID	Period	Amount
120265	Qtr4	25

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

Out of range

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	5	Qtr4	25

work.rotate

Employee_ID	Period	Amount
120265	Qtr4	25

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

No Implicit OUTPUT;
Implicit RETURN;

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	5	Qtr4	25

work.rotate

Employee_ID	Period	Amount
120265	Qtr4	25

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

Reinitialize PDV

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120265	.	.	.	25	.		.

work.rotate

Employee_ID	Period	Amount
120265	Qtr4	25

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120267	15	15	15	15	.		.

work.rotate

Employee_ID	Period	Amount
120265	Qtr4	25

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120267	15	15	15	15	1		.

work.rotate

Employee_ID	Period	Amount
120265	Qtr4	25

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

True

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120267	15	15	15	15	1		.

work.rotate

Employee_ID	Period	Amount
120265	Qtr4	25

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120267	15	15	15	15	1	Qtr1	.

`work.rotate`

Employee_ID	Period	Amount
120265	Qtr4	25

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120267	15	15	15	15	1	Qtr1	15

work.rotate

Employee_ID	Period	Amount
120265	Qtr4	25

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

Output current observation

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120267	15	[illegible]	[illegible]	[illegible]	1	Qtr1	15

work.rotate

Employee_ID	Period	Amount
120265	Qtr4	25
120267	Qtr1	15

...

Execution: Rotating a SAS Data Set

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
            (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
       if contrib{i} ne . then do;
          Period=cats("Qtr",i);
          Amount=contrib{i};
          output;
       end;
   end;
run;
```

Continue until EOF

PDV

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	i	Period	Amount
120267	15	15	15	15	1	Qtr1	15

work.rotate

Employee_ID	Period	Amount
120265	Qtr4	25
120267	Qtr1	15

Output: The Rotate Data Set

```
proc print data=rotate;
run;
```

Partial PROC PRINT Output

```
Obs     Employee_ID    Period    Amount

  1         120265      Qtr4        25
  2         120267      Qtr1        15
  3         120267      Qtr2        15
  4         120267      Qtr3        15
  5         120267      Qtr4        15
  6         120269      Qtr1        20
  7         120269      Qtr2        20
  8         120269      Qtr3        20
  9         120269      Qtr4        20
 10         120270      Qtr1        20
 11         120270      Qtr2        10
 12         120270      Qtr3         5
```

p208d01

Analyzing the Rotated SAS Data Set

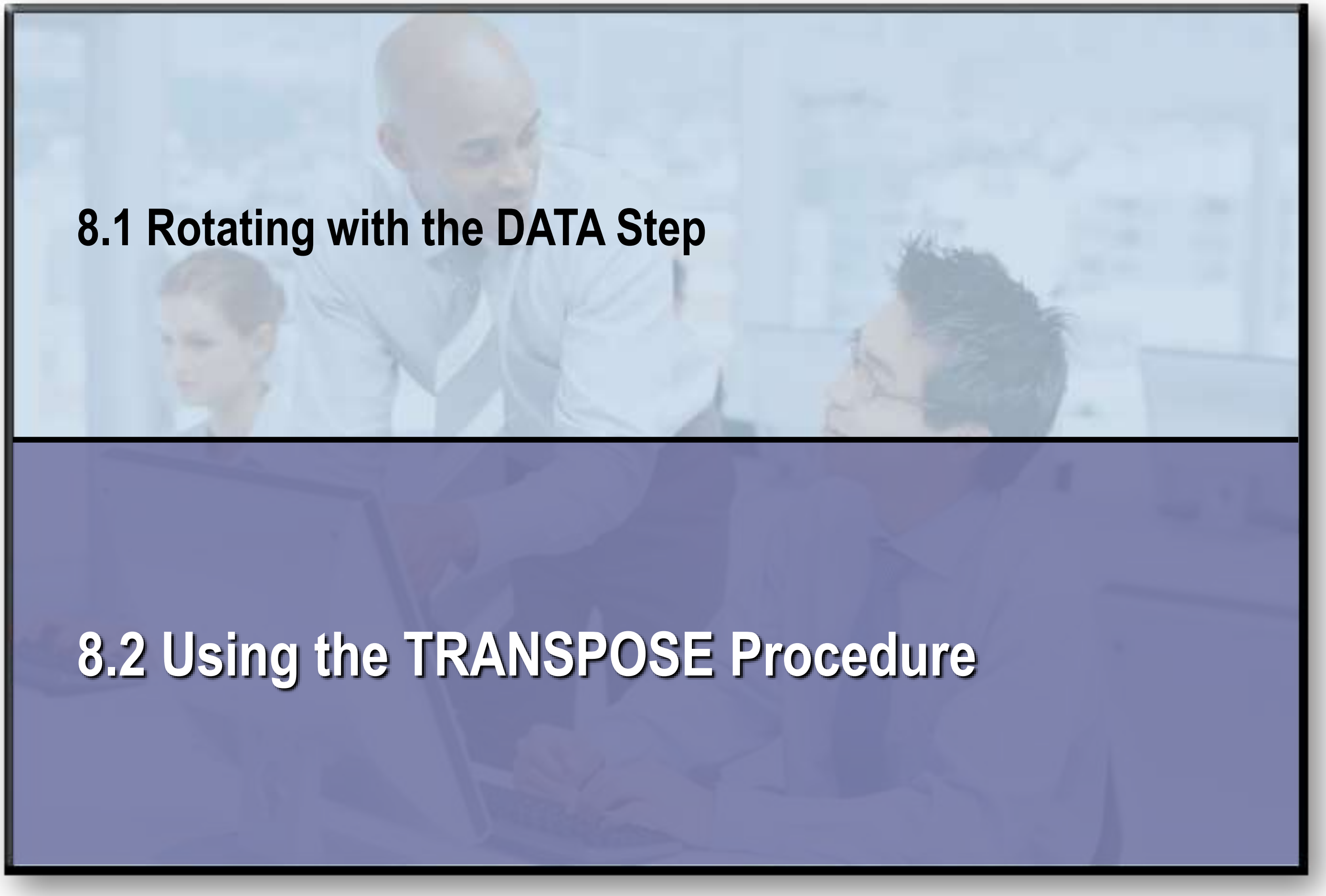
```
proc freq data=rotate;
   tables Period /nocum nopct;
run;
```

PROC FREQ Output

```
Period     Frequency
--------------------
Qtr1             110
Qtr2              98
Qtr3             107
Qtr4             102
```

p208d01

Chapter 8: Restructuring a Data Set



8.1 Rotating with the DATA Step

8.2 Using the TRANSPOSE Procedure

Objectives

- Use the TRANSPOSE procedure to restructure a data set.

Business Scenario (Review)

The Orion Payroll Manager asked for a report showing the number of Orion Star employees who made charitable donations in each quarter.

Sketch of the Desired Report

```
Period      Frequency
Qtr1               56
Qtr2               99
Qtr3               24
Qtr4               75
```

Business Scenario – Review

The data set **orion.employee_donations** has a wide structure with one observation per employee (124 total observations).

Partial Listing of **orion.employee_donations**

```
  Employee_ID   Qtr1    Qtr2     Qtr3    Qtr4   Paid_By
    120265        .       .        .      25    Cash or Check
    120267       15      15       15      15    Payroll Deduction
    120269       20      20       20      20    Payroll Deduction
```

With a restructured, narrow data set, the FREQ procedure can be used to generate the desired output.

This example uses PROC TRANSPOSE to restructure the data.

Poll

Quiz

Setup for the Poll

Open SAS Help and navigate to the PROC TRANSPOSE section:

- **SAS Products** ⇨
  - **Base SAS** ⇨
    - **Base SAS 9.4 Procedures Guide** ⇨
      - **Procedures** ⇨
        - **Transpose Procedure** ⇨
          - **Overview: Transpose Procedure**

Review the Overview section.

8.02 Multiple Answer Poll

Which of the following statements are true of the TRANSPOSE procedure?

a. It produces printed output.

b. It creates a new data set.

c. It often eliminates the need for a complex DATA step.

d. It transposes selected variables into observations.

8.02 Multiple Answer Poll – Correct Answers

Which of the following statements are true of the TRANSPOSE procedure?

a. It produces printed output.
[✓ b.] It creates a new data set.
[✓ c.] It often eliminates the need for a complex DATA step.
[✓ d.] It transposes selected variables into observations.

The TRANSPOSE Procedure

General form of a PROC TRANSPOSE step:

```
PROC TRANSPOSE DATA=input-data-set
                             <OUT=output-data-set>
                             <NAME = variable-name>;
      <BY <DESCENDING> variable-1
        <...<DESCENDING> variable-n> <NOTSORTED>;>
      <VAR variable(s);>
      <ID variable;>
RUN;
```

NAME=	specifies a new name for the _NAME_ column. The values in this column identify the variable that supplied the values in the row.
BY	specifies the variable(s) to use to form BY groups.
VAR	specifies the variable(s) to transpose.
ID	specifies the variable whose values will become the new variables.

The TRANSPOSE Procedure

The TRANSPOSE procedure

- transposes selected variables into observations
- transposes numeric variables by default
- transposes character variables only if explicitly listed in a VAR statement.

Using the Transpose Procedure

Start with a simple PROC TRANSPOSE step:

```
proc transpose
     data=orion.employee_donations
     out=rotate2;
run;
```

Partial Listing of **rotate2**

```
_NAME_        _LABEL_       COL1     COL2     COL3    ...   COL124
Employee_ID  Employee ID  120265   120267   120269          121147
Qtr1                           .       15       20              10
Qtr2                           .       15       20              10
Qtr3                           .       15       20              10
Qtr4                          25       15       20              10
```

The output is very different from the desired results. A row was created for each variable. A column was created for each of the 124 observations.

p208d02

Results of a Simple Transposition

Compare PROC TRANSPOSE output to the original data:

Partial Listing of **`orion.employee_donations`**

```
Employee_ID     Qtr1     Qtr2     Qtr3     Qtr4          Paid_By
     120265        .        .        .       25     Cash or Check
     120267       15       15       15       15     Payroll Deduction
     120269       20       20       20       20     Payroll Deduction
```

Partial Listing of **`rotate2`**

```
_NAME_              _LABEL_      COL1      COL2      COL3     ...     COL124
Employee_ID    Employee ID     120265    120267    120269             121147
Qtr1                                .        15        20                 10
Qtr2                                .        15        20                 10
Qtr3                                .        15        20                 10
Qtr4                               25        15        20       .         10
```

All the numeric variables were transposed by default.

`Paid_By`, a character variable, was not transposed.

Results of a Simple Transposition

Partial Listing of **orion.employee_donations**

```
Employee_ID     Qtr1     Qtr2     Qtr3     Qtr4        Paid_By
     120265        .        .        .       25    Cash or Check
     120267       15       15       15       15    Payroll Deduction
     120269       20       20       20       20    Payroll Deduction
     120270       20       10        5        .    Cash or Check
     120271       20       20       20       20    Payroll Deduction
```

Partial Listing of **rotate2**

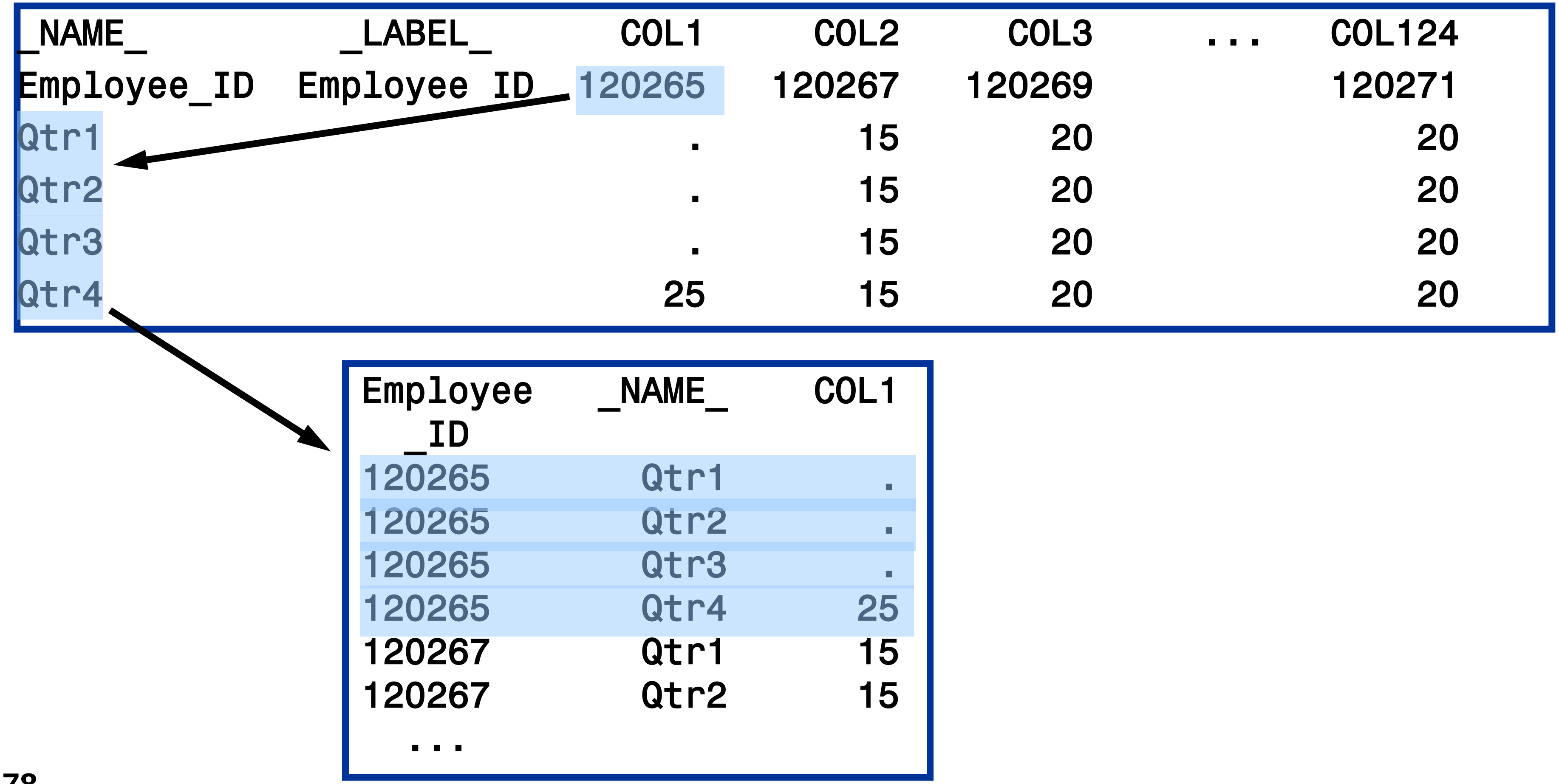
```
_NAME_         _LABEL_           COL1      COL2      COL3    ...    COL124
Employee_ID   Employee ID      120265    120267    120269           121147
Qtr1                                .        15        20               10
Qtr2                                .        15        20               10
Qtr3                                .        15        20               10
Qtr4                               25        15        20      .        10
```

Each observation (row) in the input data set becomes a variable (column) in the output data set.

PROC TRANSPOSE Results

The data should be grouped by **Employee_ID** with a separate observation for each transposed variable.

Partial Listing of **rotate2**



```
_NAME_          _LABEL_        COL1      COL2      COL3      ...     COL124
Employee_ID  Employee ID     120265    120267    120269              120271
Qtr1                              .        15        20                  20
Qtr2                              .        15        20                  20
Qtr3                              .        15        20                  20
Qtr4                             25        15        20                  20
```

```
Employee     _NAME_     COL1
  _ID
120265        Qtr1         .
120265        Qtr2         .
120265        Qtr3         .
120265        Qtr4        25
120267        Qtr1        15
120267        Qtr2        15
  ...
```

The BY Statement

Use a BY statement to group the output by **`Employee_ID`**.

```
proc transpose
          data=orion.employee_donations
          out=rotate2;
     by Employee_ID;
run;
proc print data=rotate2 noobs;
run;
```

All numeric variables other than the BY variable are transposed.

p208d03

Improved PROC TRANSPOSE Results

Use of the BY statement results in one observation for each transposed variable per **`Employee_ID`**, and includes missing values.

Partial PROC PRINT Output

```
Employee_ID     _NAME_     COL1

     120265      Qtr1        .
     120265      Qtr2        .
     120265      Qtr3        .
     120265      Qtr4       25
     120267      Qtr1       15
     120267      Qtr2       15
     120267      Qtr3       15
     120267      Qtr4       15
```

If there were additional numeric variables, an observation would be created for each.

The VAR Statement

The VAR statement is used to specify which variables to transpose. It can include character and numeric variables.

```
proc transpose
     data=orion.employee_donations
     out=rotate2;
   by Employee_ID;
   var Qtr1-Qtr4;
run;
proc print data=rotate2 noobs;
run;
```

The VAR statement has no effect in this example because **`Qtr1`**-**`Qtr4`** will be transposed by default.

p208d03b

Enhancing PROC TRANSPOSE Results

The final step is to change the default names of the new variables.

Partial PROC PRINT Output

```
Employee_ID     _NAME_     COL1

     120265      Qtr1         .
     120265      Qtr2         .
     120265      Qtr3         .
     120265      Qtr4        25
     120267      Qtr1        15
     120267      Qtr2        15
     120267      Qtr3        15
     120267      Qtr4        15
```

- Change **`_NAME_`** to **`Period`**.
- Change **`COL1`** to **`Amount`**.

Renaming Variables in PROC TRANSPOSE

```
proc transpose
        data=orion.employee_donations
        out=rotate2
        name=Period;
   by Employee_ID;
run;
proc print data=rotate2 noobs;
run;
```

The PROC TRANSPOSE option, NAME=, is used to rename _NAME_.

Partial Listing of **rotate2**

```
Employee_ID     Period     COL1

   120265        Qtr1        .
   120265        Qtr2        .
   120265        Qtr3        .
   120265        Qtr4       25
   120267        Qtr1       15
   120267        Qtr2       15
```

p208d04

Renaming Variables in PROC TRANSPOSE

```
proc transpose
        data=orion.employee_donations
        out=rotate2 (rename=(col1=Amount))
        name=Period;
   by employee_id;
run;
proc print data=rotate2 noobs;
run;
```

The RENAME= data set option is used to change the name of COL1.

Partial Listing of **`rotate2`**

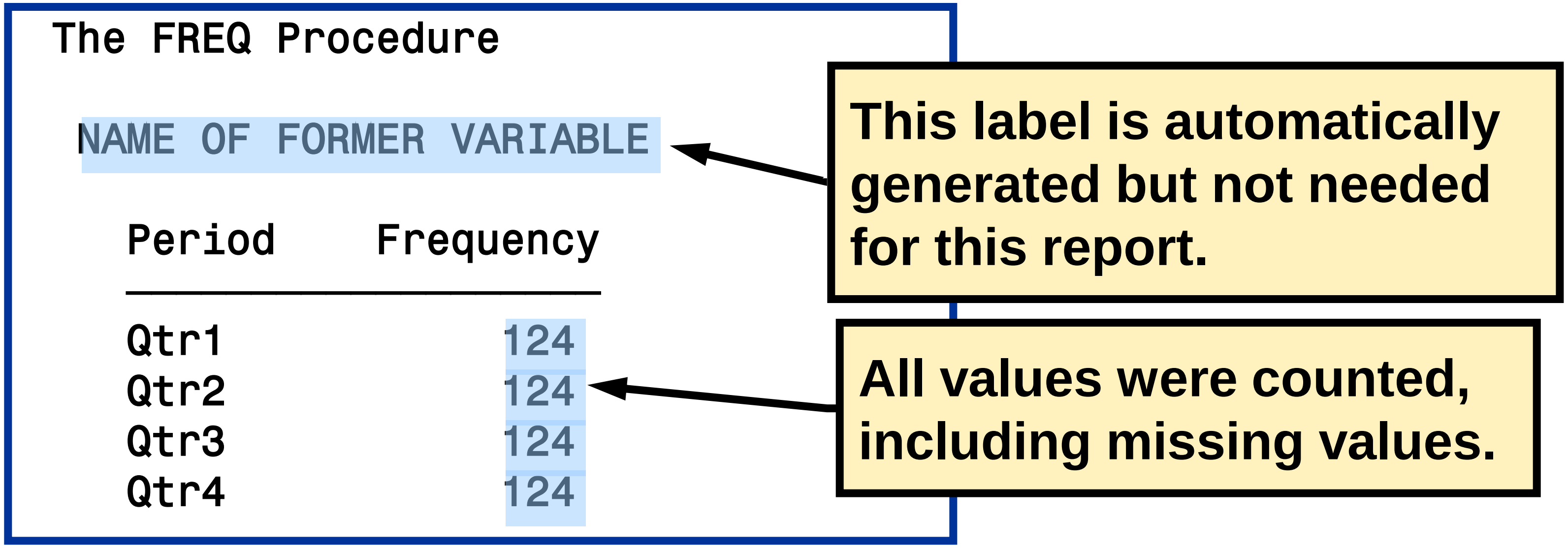
```
Employee_ID     Period    Amount
     120265     Qtr1           .
     120265     Qtr2           .
     120265     Qtr3           .
     120265     Qtr4          25
     120267     Qtr1          15
     120267     Qtr2          15
```

p208d04

Analyze the Restructured Data Set

The FREQ procedure generates the report below. The frequency is 124 for all four variables.

```
proc freq data=rotate2;
   tables Period /nocum nopct;
run;
```



```
 The FREQ Procedure

  NAME OF FORMER VARIABLE

    Period      Frequency
    ---------------------
    Qtr1              124
    Qtr2              124
    Qtr3              124
    Qtr4              124
```

This label is automatically generated but not needed for this report.

All values were counted, including missing values.

p208d04

Poll

Quiz

8.03 Quiz

Open **p208a01** and submit it. A LABEL statement was already added to suppress the label. Add a WHERE statement to select only observations with nonmissing **`Amount`** values.

```
proc freq data=rotate2;
   tables Period/nocum nopct;
   label Period=" ";
run;
```

p208a01

8.03 Quiz – Correct Answer

Any of the following WHERE statements can be used to select observations with nonmissing **`Amount`** values.

```
where Amount ne .;

where Amount is not missing;

where not missing(amount);

where Amount is not null;
```

FREQ Output

```
proc freq data=rotate2;
   where Amount ne .;
   tables Period/nocum nopct;
   label Period=" ";
run;
```

```
The FREQ Procedure

Period       Frequency
______________________

Qtr1               110
Qtr2                98
Qtr3               107
Qtr4               102
```

p208a01s

The WHERE= Data Set Option

The WHERE= data set option specifies conditions to use to subset a SAS data set.

General form of the WHERE= option:

SAS-data-set(WHERE=(*where-expression*))

The WHERE= option

- can be used on both input and output data sets
- applies only to the data set for which it is specified.

The WHERE= Data Set Option

There is no option or statement in PROC TRANSPOSE to eliminate observations with missing values for the transposed variable. However, this can be achieved using a WHERE= data set option in the output data set.

```
proc transpose
        data=orion.employee_donations
        out=rotate2(rename=(col1=Amount)
                    where=(Amount ne .))
        name=Period;
   by employee_id;
run;
proc print data=rotate2 noobs;
run;
proc freq data=rotate2;
   tables Period/nocum nopct;
   label Period=" ";
run;
```

p208d05

No Missing Values

Partial PROC PRINT Output

```
Employee_ID     Period     Amount

     120265      Qtr4         25
     120267      Qtr1         15
     120267      Qtr2         15
     120267      Qtr3         15
     120267      Qtr4         15
     120269      Qtr1         20
     120269      Qtr2         20
     120269      Qtr3         20
     120269      Qtr4         20
     120270      Qtr1         20
     120270      Qtr2         10
     120270      Qtr3          5
```

PROC FREQ Output

```
The FREQ Procedure

Period    Frequency
-------------------
Qtr1            110
Qtr2             98
Qtr3            107
Qtr4            102
```

The resulting data set has no missing values. Now PROC FREQ produces the desired results.

Business Scenario

The manager of Sales asked for a report showing monthly sales and a total for each customer.

Sketch of the Desired Report

Monthly Sales by Customer

Customer_ID	Month1	Month2	…	Month12	Total
1	1000	.		500	2000
2	.	.		200	750
3	1200	.		.	2200
4	500	150		350	1000
5	.	1000		.	2500

Business Scenario Considerations

The data set **orion.order_summary** contains an observation for each month in which a customer placed an order (101 total observations). The data set is sorted by **Customer_ID** and has no missing values.

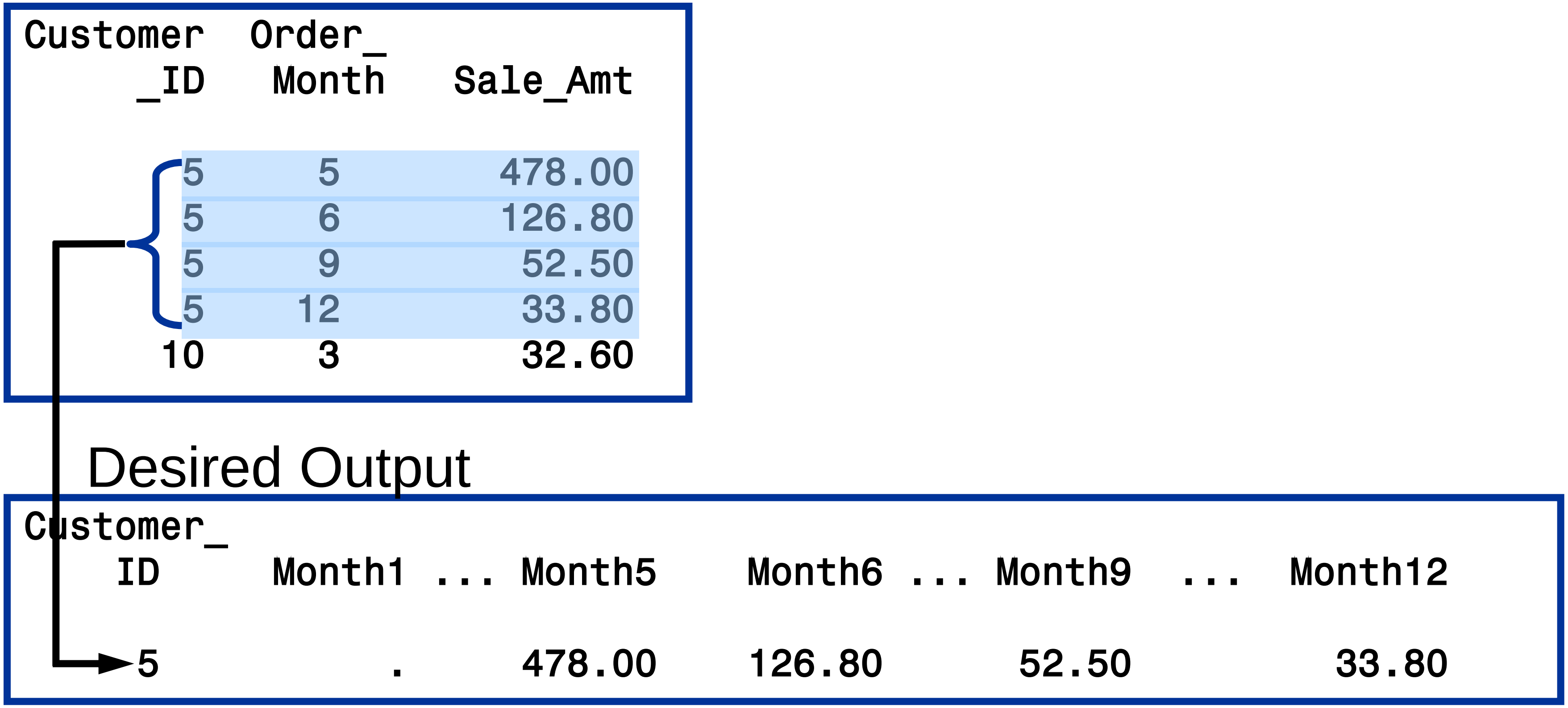
Partial Listing of **orion.order_summary**

```
                    Order_
Customer_ID          Month      Sale_Amt

          5             5         478.00
          5             6         126.80
          5             9          52.50
          5            12          33.80
         10             3          32.60
         10             4         250.80
         10             5          79.80
         10             6          12.20
         10             7         163.29
```

The number of observations per customer varies.

Business Scenario Considerations

The report requires rotating the columns into rows. Use PROC TRANSPOSE again to restructure the data set, and this time from narrow to wide.



```
Customer   Order_
     _ID    Month     Sale_Amt

       5        5       478.00
       5        6       126.80
       5        9        52.50
       5       12        33.80
      10        3        32.60
```

Desired Output

```
Customer_
    ID      Month1 ... Month5    Month6 ... Month9   ...   Month12

     5           .     478.00    126.80      52.50           33.80
```

Using PROC TRANSPOSE

Start with a simple PROC TRANSPOSE.

Partial Listing of **orion.order_summary**

```
                       Order_
   Customer_ID          Month       Sale_Amt

             5              5         478.00
             5              6         126.80
             5              9          52.50
             5             12          33.80
            10              3          32.60
            10              4         250.80
            10              5          79.80
```

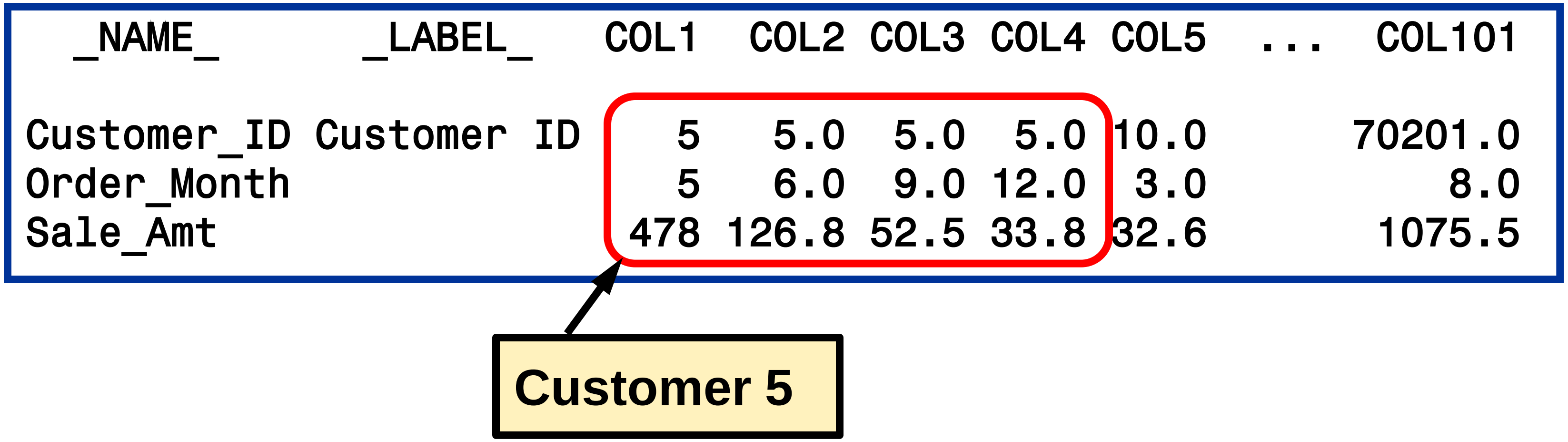
101 observations

```
proc transpose data=orion.order_summary
               out=annual_orders;
run;
proc print data=annual_orders noobs;
run;
```

p208d06

Using PROC TRANSPOSE

The resulting data set has three observations, one for each numeric variable in the input data set: **Customer_ID**, **Order_Month**, and **Sale_Amt**.



```
  _NAME_       _LABEL_    COL1   COL2  COL3  COL4  COL5   ...   COL101

Customer_ID Customer ID    5     5.0   5.0   5.0  10.0         70201.0
Order_Month                5     6.0   9.0  12.0   3.0             8.0
Sale_Amt                 478   126.8  52.5  33.8  32.6          1075.5
```

Customer 5

The variables **COL1**-**COL101** represent the 101 observations in the input data set.

Group the output by **Customer_ID**.

The BY Statement

The BY statement groups by **`Customer_ID`** and produces an observation for each transposed variable, **`Order_Month`** and **`Sale_Amt`**.

```
proc transpose data=orion.order_summary
               out=annual_orders;
   by Customer_ID;
run;
```

Notice the varying number of columns for each customer.

```
Customer
     _ID      _NAME_         COL1     COL2     COL3     COL4     COL5      COL6      COL7      COL8     COL9

       5    Order_Month       5.0      6.0      9.0     12.0       .         .         .         .        .
       5    Sale_Amt        478.0    126.8     52.5     33.8       .         .         .         .        .
      10    Order_Month       3.0      4.0      5.0      6.0     7.00       8.0      11.0      12.0       .
      10    Sale_Amt         32.6    250.8     79.8     12.2   163.29     902.5    1894.6     143.3       .
      11    Order_Month       9.0       .        .        .        .         .         .         .        .
      11    Sale_Amt         78.2       .        .        .        .         .         .         .        .
```

p208d07

Creating Columns Based on a Variable

Instead of transposing **`Order_Month`**, use its values to create new variables. A value of 5.0 represents orders placed in May, 6.0 represents orders placed in June, and so on.

```
Customer
     _ID     _NAME_        COL1     COL2     COL3     COL4     COL5      COL6      COL7     COL8    COL9

       5   Order_Month      5.0      6.0      9.0     12.0       .         .         .        .       .
       5   Sale_Amt       478.0    126.8     52.5     33.8       .         .         .        .       .
      10   Order_Month      3.0      4.0      5.0      6.0     7.00       8.0      11.0     12.0       .
      10   Sale_Amt        32.6    250.8     79.8     12.2   163.29     902.5    1894.6    143.3       .
      11   Order_Month      9.0       .        .        .        .         .         .        .       .
      11   Sale_Amt        78.2       .        .        .        .         .         .        .       .
```

Add an ID statement.

The ID Statement

The ID statement identifies the variable whose values will become the names of the new columns.

```
proc transpose data=orion.order_summary
               out=annual_orders;
   by Customer_ID;
   id Order_Month;
run;
```

The values of `Order_Month` (1, 2, 3, … 12) are used to create variable names `_1` through `_12`.

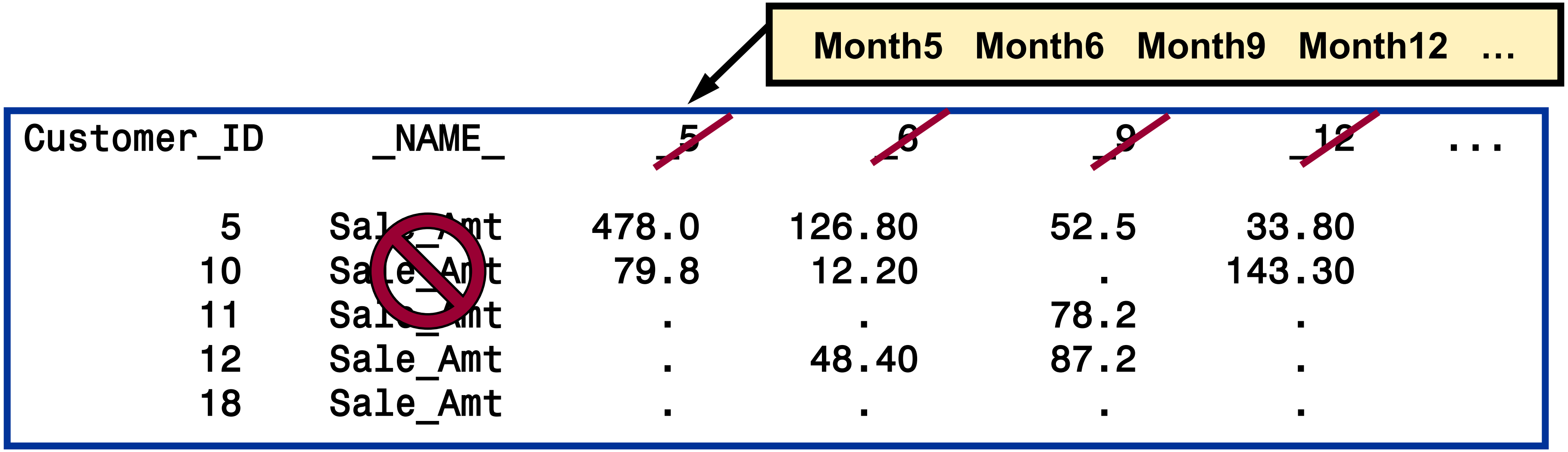
```
Customer_ID     _NAME_         _5        _6        _9      _12    ...

      5       Sale_Amt     478.0    126.80     52.5     33.80
     10       Sale_Amt      79.8     12.20       .     143.30
     11       Sale_Amt        .         .      78.2        .
     12       Sale_Amt        .      48.40     87.2        .
     18       Sale_Amt        .         .        .         .
```

The remaining variable, **`Sale_Amt`**, is transposed.

p208d08

Enhancing PROC TRANSPOSE Results

What other changes can enhance the report?



Month5   Month6   Month9   Month12   ...

```
Customer_ID     _NAME_        ~~_5~~          ~~_6~~          ~~_9~~          ~~_12~~      ...

          5    Sale_Amt     478.0    126.80       52.5      33.80
         10    Sale_Amt      79.8     12.20         .      143.30
         11    Sale_Amt        .        .         78.2        .
         12    Sale_Amt        .      48.40       87.2        .
         18    Sale_Amt        .        .           .         .
```

- Change the variable names from **_*n*** to **Month*n***.
- Drop the **_NAME_** variable.

Changing the Variable Names

The PREFIX= option is used to set a prefix for each new variable name. The prefix replaces the underscore.

```
proc transpose data=orion.order_summary
               out=annual_orders
               prefix=Month;
     by Customer_ID;
     id Order_Month;
run;
```

```
  Customer_ID     _NAME_     Month5     Month6    Month9   ...

          5      Sale_Amt    478.0      126.80     52.5
         10      Sale_Amt     79.8       12.20      .
         11      Sale_Amt      .          .        78.2
         12      Sale_Amt      .         48.40     87.2
         18      Sale_Amt      .          .         .
```

p208d09

Dropping the _NAME_ Column

Use the DROP= data set option to drop the **_NAME_** variable.

```
proc transpose data=orion.order_summary
               out=annual_orders(drop=_name_)
               prefix=Month;
   by Customer_ID;
   id Order_Month;
run;
```

```
 Customer_ID   Month5   Month6   Month9   Month12   Month3   ...

           5    478.0   126.80     52.5     33.80       .
          10     79.8    12.20       .     143.30     32.6
          11       .       .       78.2       .         .
          12       .      48.40    87.2       .         .
```

p208d10

Poll

Quiz

8.04 Quiz

Notice the column order in the PROC PRINT output. Why are the variables out of sequence?

```
 Customer_ID  Month5  Month6   Month9  Month12  Month3    ...

           5   478.0  126.80    52.5    33.80      .
          10    79.8   12.20      .    143.30    32.6
          11      .      .      78.2      .        .
          12      .    48.40    87.2      .        .
```

8.04 Quiz – Correct Answer

Notice the column order in the PROC PRINT output. Why are the variables out of sequence?

```
 Customer_ID  Month5   Month6   Month9   Month12   Month3    ...

           5   478.0   126.80     52.5     33.80        .
          10    79.8    12.20        .    143.30     32.6
          11       .        .     78.2         .        .
          12       .    48.40     87.2         .        .
```

Partial Listing of **orion.order_summary**

```
                   Order_
Customer_ID         Month       Sale_Amt

          5           5          478.00
          5           6          126.80
          5           9           52.50
          5          12           33.80
         10           3           32.60
```

The variables were created in the order that they appeared in the input data set.

Print the Transposed Data Set

A VAR statement in the PRINT procedure specifies the desired order of the variables.

```
proc print data=annual_orders noobs;
   var Customer_ID Month1-Month12;
run;
```

```
Customer_ID  Month1   Month2   Month3   Month4   Month5   ...

          5       .        .        .        .    478.0
         10       .        .      32.6    250.8    79.8
         11       .        .        .        .        .
         12       .     117.6       .        .        .
         18       .      29.4       .        .        .
         24   195.6        .      46.9       .        .
         27   174.4        .     140.7    205.0       .
```

p208d11

Reorder Data Set Variables (Self-Study)

The RETAIN statement can be used in a DATA step to permanently change the order of the variables in an existing data set.

```
data annual_orders;
     retain Customer_ID Month1-Month12;
     set annual_orders;
run;
```

The data set **annual_orders** is used for input and output.

⚠ It is recommended that no additional processing be performed in the DATA step

p208d12

Examine the Resulting Data Set (Self-Study)

The variables are now in the desired order.

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

Partial PROC CONTENTS Output

```
Variables in Creation Order

 #    Variable        Type    Len     Format     Label

 1    Customer_ID     Num       8     12.        Customer ID
 2    Month1          Num       8
 3    Month2          Num       8
 4    Month3          Num       8
 5    Month4          Num       8
 6    Month5          Num       8
 7    Month6          Num       8
 8    Month7          Num       8
 9    Month8          Num       8
10    Month9          Num       8
11    Month10         Num       8
12    Month11         Num       8
13    Month12         Num       8
```

p208d12

Print the Resulting Data Set (Self-Study)

```
proc print data=annual_orders;
run;
```

Partial PROC PRINT Output

```
 Customer_ID  Month1    Month2    Month3    Month4    Month5    ...

           5       .         .         .         .     478.0
          10       .         .      32.6     250.8      79.8
          11       .         .         .         .         .
          12       .     117.6         .         .         .
          18       .      29.4         .         .         .
          24   195.6         .      46.9         .         .
          27   174.4         .     140.7     205.0         .
```

p208d12

Advantages of Each Restructuring Method

The TRANSPOSE Procedure
Might eliminate the need for a complex DATA step
Requires very little code to restructure data

The DATA Step
Can create multiple data sets
Can direct output to data sets based on data set contributors
Enables First. and Last. processing
Enables complex data manipulation