Chapter 4
BY-Group Processing in the DATA Step

Arthur Li

Introduction to BY-Group Processing

- Longitudinal data: Multiple observations per subject

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

- Identify the beginning/end of measurement for each subject
- This can be accomplished by using the BY-group processing method

Introduction to BY-Group Processing

- **BY-group processing**: is a method of processing records from data sets that can be grouped by the values of one or more common variables.

- **BY variable**: the “grouping” variables

- **BY value**. The value of a BY variable

- **BY group**: all observations with the same BY value.

The FIRST.VARIABLE and the LAST.VARIABLE

- BY-group processing method:

```
proc sort data=b;
    by by_variable;
run;
data a;
    set b;
    by by_variable;
    ...
    ...
run;
```

- For each BY-variable, SAS creates two temporary variables:
  - FIRST.VARIABLE
  - LAST.VARIABLE
- FIRST.VARIABLE & LAST.VARIABLE are set to 1 at the beginning of the execution phase
- They are not being output to the final dataset

The FIRST.VARIABLE and the LAST.VARIABLE

- Suppose ID is the “BY” variable:

	ID	SCORE	“GROUPING”	FIRST.ID	LAST.ID
1	A01	3	1	1	0
2	A01	3		0	0
3	A01	2		0	1
4	A02	4	2	1	0
5	A02	2		0	1

Grouping
based ID

The FIRST.VARIABLE and the LAST.VARIABLE

- Suppose ID is the “BY” variable:

	ID	SCORE	“GROUPING”	FIRST.ID	LAST.ID
1	A01	3	1	1	0
2	A01	3		0	0
3	A01	2		0	1
4	A02	4	2	1	0
5	A02	2		0	1

SAS reads the 1st observation for ID = A01 (group 1)

SAS reads the first observation for ID = A02 (group2)

Grouping based ID

The FIRST.VARIABLE and the LAST.VARIABLE

- Suppose ID is the “BY” variable:

	ID	SCORE	“GROUPING”	FIRST.ID	LAST.ID
1	A01	3	1	1	0
2	A01	3		0	0
3	A01	2		0	1
4	A02	4	2	1	0
5	A02	2		0	1

SAS reads the last observation for ID = A01 (group 1)

SAS reads the last observation for ID = A02 (group 2)

Grouping based ID

The FIRST.VARIABLE and the LAST.VARIABLE

- Suppose ID and SCORE are the “BY” variables:

	ID	SCORE	“GROUPING”	FIRST.ID	LAST.ID	“GROUPING”	FIRST.SCORE	LAST.SCORE
1	A01	3	1	1	0	1	1	0
2	A01	3		0	0		0	1
3	A01	2		0	1	2	1	1
4	A02	4	2	1	0	3	1	1
5	A02	2		0	1	4	1	1

Grouping based ID

Grouping based ID and SCORE

The Execution Phase of By-Group Processing

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

	ID	TOTAL
1	A01	8
2	A02	6

Approach:

- ❖Initialize TOTAL to 0 when starting to read the first observation of each subject
- ❖Accumulate TOTAL by adding the values from SCORE
- ❖Output the ID and TOTAL to the output dataset when reading the last observation of each subject

The Execution Phase of By-Group Processing

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

	ID	TOTAL
1	A01	8
2	A02	6

Program 4.1

```
proc sort data=sas4_1;
    by id;
run;

data sas4_2 (drop = score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
1	1	1		.	0

ERROR is not shown for purpose of simplicity

1st Iteration:

❖ _N_ ←1

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
1	1	1		.	0

1st Iteration:

- ❖ _N_ ←1
- ❖ FIRST.ID ← 1, LAST.ID ← 1

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
1	1	1		.	0

1st Iteration:

❖ _N_ ←1

❖ FIRST.ID ← 1, LAST.ID ← 1

❖ ID, Score ← missing

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
1	1	1		.	0

1st Iteration:

- ❖ _N_ ←1
- ❖ FIRST.ID ← 1, LAST.ID ← 1
- ❖ ID, Score ← missing
- ❖ TOTAL ← 0 because of the SUM statement

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
1	1	0	A01	3	0

1st Iteration:

- ❖ 1st observation → PDV

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
1	1	0	A01	3	0

1st Iteration:

❖ 1st observation → PDV

❖ FIRST.ID ← 1 and LAST.ID ← 0

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
 ➔ by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
1	1	0	A01	3	0

1st Iteration:

- BY statement is a declarative statement

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

	_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	SCORE	D	TOTAL	K
PDV	1		1		0		A01		3		0	

1st Iteration:

❖ FIRST.ID = 1: TOTAL ← 0

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
 ➡  total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
1	1	0	A01	3	3

1st Iteration:

- TOTAL is calculated

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
➔   if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
1	1	0	A01	3	3

1^{st} Iteration:

- Since LAST.ID ≠ 1, (the subsetting IF statement is false), no further statements are processed for the current observation. SAS immediately returns to the beginning of the DATA step

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
2	1	0	A01	3	3

2nd Iteration:

- _N_ ↑ 2

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
2	1	0	A01	3	3

2nd Iteration:

- _N_ ↑ 2
- FIRST.ID & LAST.ID are retained (automatic variables)

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
2	1	0	A01	3	3

2nd Iteration:

❖ _N_ ↑ 2

❖ FIRST.ID & LAST.ID are retained (automatic variables)

❖ ID & SCORE are retained (read from input data)

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
2	1	0	A01	3	3

2nd Iteration:

❖_N_ ↑ 2

❖FIRST.ID & LAST.ID are retained (automatic variables)

❖ID & SCORE are retained (read from input data)

❖TOTAL is retained (SUM statement)

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
2	0	0	A01	3	3

2nd Iteration:

❖ 2nd observation → PDV

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
  ➡ set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
2	0	0	A01	3	3

2nd Iteration:

- ❖ 2nd observation → PDV
- ❖ Not the first observation for A01: FIRST.ID ←0
- ❖ Not the last observation for A01: LAST.ID ← 0

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
2	0	0	A01	3	3

2nd Iteration:

- ❖ FIRST.ID ≠ 1: no execution

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
  ➡ total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	SCORE	D	TOTAL	K
2		0		0		A01		3		6	

2nd Iteration:

❖ TOTAL is calculated

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
  ➡ if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
2	0	0	A01	3	6

⇧

2nd Iteration:

- ❖ Since LAST.ID ≠ 1 (the subsetting IF statement is false), SAS immediately returns to the beginning of the DATA step

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
3	0	0	A01	3	6

3rd Iteration:

- ❖ _N_ ↑3
- ❖ The values for the rest of the variables are retained

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
   set sas4_1;
   by id;
   if first.id then total = 0;
   total + score;
   if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
3	0	1	A01	2	6

3rd Iteration:

❖3rd observation →PDV

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV	_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
	3	0	1	A01	2	6

3rd Iteration:

- 3rd observation →PDV
- Not the first observation: FIRST.ID ← 0
- Last observation for A01: LAST.ID ← 1

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
  ➡ if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
3	0	1	A01	2	6

3rd Iteration:

❖FIRST.ID ≠ 1: no execution

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
  ➡  total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
3	0	1	A01	2	8

3rd Iteration:

❖TOTAL is calculated

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
  ➡ if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
3	0	1	A01	2	8

3rd Iteration:

❖ Since LAST.ID = 1 (the subsetting IF statement is true), SAS continues to execute the remaining statements in the DATA step

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
3	0	1	A01	2	8

3rd Iteration:

- ❖ SAS reaches the end of the 3rd iteration
  - ❑ The implicit OUTPUT statement copies ID and TOTAL in the PDV to the output data set
  - ❑ SAS returns to the beginning of the DATA step to begin the 4th iteration

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
4	0	1	A01	2	8

4th Iteration:

❖ _N_ ↑ 4

❖ The values for the remaining variables are retained

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
   set sas4_1;
   by id;
   if first.id then total = 0;
   total + score;
   if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
4	1	0	A02	4	8

4th Iteration:

- 4th observation → PDV

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
 ➔ set sas4_1;
   by id;
   if first.id then total = 0;
   total + score;
   if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
4	1	0	A02	4	8

4th Iteration:

- ❖4th observation → PDV
- ❖FIRST.ID ←1
- ❖LAST.ID ← 0

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
4	1	0	A02	4	0

4th Iteration:

❖FIRST.ID = 1: TOTAL ←0

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
   ➡ total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	SCORE	D	TOTAL	K
4		1		0		A02		4		4	

<u>4th Iteration:</u>

❖ TOTAL is calculated

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
  ➡ if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
4	1	0	A02	4	4

⇧

4th Iteration:

❖ Since LAST.ID ≠ 1(the subsetting IF statement is false), SAS immediately returns to the beginning of the DATA step

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
5	1	0	A02	4	4

5th Iteration:

❖ _N_ ↑ 5

❖ The values for the remaining variables are retained

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
   set sas4_1;
   by id;
   if first.id then total = 0;
   total + score;
   if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
5	0	1	A02	2	4

5th Iteration:

- ❖ 5th observation → PDV

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
   set sas4_1;
   by id;
   if first.id then total = 0;
   total + score;
   if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
5	0	1	A02	2	4

5th Iteration:

- ❖ 5th observation → PDV
- ❖ FIRST.ID ← 0
- ❖ LAST.ID ← 1

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
5	0	1	A02	2	4

5th Iteration:

- ❖ FIRST.ID ≠ 1: no execution

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
5	0	1	A02	2	6

5th Iteration:

❖ TOTAL is calculated

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
  ➜ if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV	_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
	5	0	1	A02	2	6

5th Iteration:

- ❖ Since LAST.ID equals 1 (the subsetting IF statement is true), SAS continues to execute the remaining statements in the DATA step

SAS4_2:

	ID	TOTAL
1	A01	8

Execution Phase of Program 4.1

```
data sas4_2(drop=score);
     set sas4_1;
     by id;
     if first.id then total = 0;
     total + score;
     if last.id;
run;
```

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

PDV

_N_ D	FIRST.ID D	LAST.ID D	ID K	SCORE D	TOTAL K
5	0	1	A02	2	6

5th Iteration:

- ❖ SAS reaches the end of the DATA step.
- ❖ The implicit OUTPUT statement copies ID and TOTAL in the PDV to the output data

SAS4_2:

	ID	TOTAL
1	A01	8
2	A02	6

Applications Utilizing BY-Group Processing

- A DATA step program that uses by-group processing frequently contains the following:

1. A cumulating variable is initialized to 0 when the FIRST.VARIABLE equals 1
2. A cumulating variable is accumulated with some values at every iteration of the DATA step
3. Some calculation needs to be performed when the LAST.VARIABLE equals 1
4. The contents of the PDV are outputted only when the LAST.VARIABLE equals 1
5. In addition to the BY variable, an additional variable will also need to be previously sorted. However, only the BY variable is used in the SET statement in the DATA step

Applications Utilizing BY-Group Processing

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
1 → if first.id then total = 0;
    total + score;
    if last.id;
run;
```

1. A cumulating variable is initialized to 0 when the FIRST.VARIABLE equals 1

Applications Utilizing BY-Group Processing

2

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
    if last.id;
run;
```

2. A cumulating variable is accumulated with some values at every iteration of the DATA step

Applications Utilizing BY-Group Processing

```
data sas4_2(drop=score);
    set sas4_1;
    by id;
    if first.id then total = 0;
    total + score;
4 → if last.id;
run;
```

4. The contents of the PDV are outputted only when the LAST.VARIABLE equals 1

Calculating Mean Score within Each By Group

SAS4_1:

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

```
data sas4_mean (drop=score);
    set sas4_1;
    by id;
    if first.id then do;
        total = 0;
        n = 0;
    end;
    total + score;
    n + 1;
    if last.id then do;
        mean_score = total/n;
        output;
    end;
run;
```

Creating Data sets with Duplicate or Non-duplicate Observations

	ID	SCORE
1	A01	3
2	A01	3
3	A01	2
4	A02	4
5	A02	2

These two records are identical

Creating Data sets with Duplicate or Non-duplicate Observations

	ID	SCORE	FIRST.SCORE	LAST.SCORE
1	A01	3	1	0
2	A01	3	0	1
3	A01	2	1	1
4	A02	4	1	1
5	A02	2	1	1

BY-variables:
ID & SCORE

Otherwise:

	ID	SCORE
1	A01	3
2	A01	3

if FIRST.SCORE=1 &
LAST.SCORE =1:

	ID	SCORE
1	A01	2
2	A02	4
3	A02	2

Creating Data sets with Duplicate or Non-duplicate Observations

	ID	SCORE	FIRST.SCORE	LAST.SCORE
1	A01	3	1	0
2	A01	3	0	1
3	A01	2	1	1
4	A02	4	1	1
5	A02	2	1	1

Otherwise:

	ID	SCORE
1	A01	3
2	A01	3

if FIRST.SCORE=1 &
LAST.SCORE =1:

	ID	SCORE
1	A01	2
2	A02	4
3	A02	2

```
proc sort data=sas4_1;
    by id score;
run;
data sas4_1_s sas4_1_d;
    set sas4_1;
    by id score;
    if first.score and last.score then
        output sas4_1_s;
    else output sas4_1_d;
run;
```

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

- This data set PATIENTS contains the triglyceride (TGL) measurement and smoking status (SMOKE) for patients for different time periods.
- Some patients only have one measurement whereas others were measured more than once in different years.

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

- Create a data set that contains the most recent non-missing data.

	PATID	TGL_NEW	SMOKE_NEW
1	A01	150	Y

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

- Create a data set that contains the most recent non-missing data.

	PATID	TGL_NEW	SMOKE_NEW
1	A01	150	Y
2	A02	210	N

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

- Create a data set that contains the most recent non-missing data.

	PATID	TGL_NEW	SMOKE_NEW
1	A01	150	Y
2	A02	210	N
3	A03	.	Y

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

- ❖ Create a data set that contains the most recent non-missing data.

	PATID	TGL_NEW	SMOKE_NEW
1	A01	150	Y
2	A02	210	N
3	A03	.	Y
4	A04	190	N

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

- Create a data set that contains the most recent non-missing data.

	PATID	TGL_NEW	SMOKE_NEW
1	A01	150	Y
2	A02	210	N
3	A03	.	Y
4	A04	190	N
5	A05	189	

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

❖ Create a data set that contains the most recent non-missing data.

	PATID	TGL_NEW	SMOKE_NEW
1	A01	150	Y
2	A02	210	N
3	A03	.	Y
4	A04	190	N
5	A05	189	

Strategy:

1. Sort the data by PATID and VISIT
2. Use PATID as the BY-variable

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

- Create a data set that contains the most recent non-missing data.

	PATID	TGL_NEW	SMOKE_NEW
1	A01	150	Y
2	A02	210	N
3	A03	.	Y
4	A04	190	N
5	A05	189	

Strategy:

3. you initially assign TGL_NEW and SMOKE_NEW to missing values when FIRST.PATID = 1

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

❖ Create a data set that contains the most recent non-missing data.

RETAIN

	PATID	TGL_NEW	SMOKE_NEW
1	A01	150	Y
2	A02	210	N
3	A03	.	Y
4	A04	190	N
5	A05	189	

Strategy:

4. At each iteration of the DATA step:
   - ❑ TGL_NEW ←TGL if TGL is not missing
   - ❑ SMOKE_NEW ←SMOKE if SMOKE is not missing

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

❖ Create a data set that contains the most recent non-missing data.

	PATID	TGL_NEW	SMOKE_NEW
1	A01	150	Y
2	A02	210	N
3	A03	.	Y
4	A04	190	N
5	A05	189	

Strategy:

5. you will output the values in the PDV when reading the last observation of each patient.

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

```
proc sort data=patients
           out=patients_sort;
    by patid visit;
run;

data patients_single
    (drop= visit tgl smoke);
    set patients_sort;
    by patid;
    retain tgl_new smoke_new;
    if first.patid then do;
        tgl_new = .;
        smoke_new = " ";
    end;
    if not missing(tgl)
        then tgl_new=tgl;
    if not missing(smoke)
        then smoke_new=smoke;
    if last.patid;
run;
```

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
   if not missing(smoke) then smoke_new=smoke;
   if last.patid;
run;
```

_N_ D	PATID K	VISIT D	TGL D	SMOKE D
1		.	.	

1st iteration:

FIRST.PATID D	LAST.PATID D
1	1

TGL_NEW K	SMOKE_NEW K
.	

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
   if not missing(smoke) then smoke_new=smoke;
   if last.patid;
run;
```

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

_N_	D	PATID	K	VISIT	D	TGL	D	SMOKE	D
1		A01		2005		.		Y	

1st iteration:

FIRST.PATID	D	LAST.PATID	D
1		0	

TGL_NEW	K	SMOKE_NEW	K
.			

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
   if not missing(smoke) then smoke_new=smoke;
   if last.patid;
run;
```

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

_N_ D	PATID K	VISIT D	TGL D	SMOKE D
1	A01	2005	.	Y

1st iteration:

FIRST.PATID D	LAST.PATID D
1	0

TGL_NEW K	SMOKE_NEW K
.	

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
   if not missing(smoke) then smoke_new=smoke;
   if last.patid;
run;
```

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

_N_ D	PATID K	VISIT D	TGL D	SMOKE D
1	A01	2005	.	Y

1st iteration:

FIRST.PATID D	LAST.PATID D
1	0

TGL_NEW K	SMOKE_NEW K
.	

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
   if not missing(smoke) then smoke_new=smoke;
   if last.patid;
run;
```

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

_N_ D	PATID K	VISIT D	TGL D	SMOKE D
1	A01	2005	.	Y

1st iteration:

FIRST.PATID D	LAST.PATID D
1	0

TGL_NEW K	SMOKE_NEW K
.	Y

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
   if not missing(smoke) then smoke_new=smoke;
➜  if last.patid;
run;
```

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

_N_	D	PATID	K	VISIT	D	TGL	D	SMOKE	D
1		A01		2005		.		Y	

1st iteration:

FIRST.PATID	D	LAST.PATID	D
1		0	

TGL_NEW	K	SMOKE_NEW	K
.		Y	

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
   if not missing(smoke) then smoke_new=smoke;
   if last.patid;
run;
```

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

_N_ D	PATID K	VISIT D	TGL D	SMOKE D
2	A01	2005	.	Y

2nd iteration:

FIRST.PATID D	LAST.PATID D
1	0

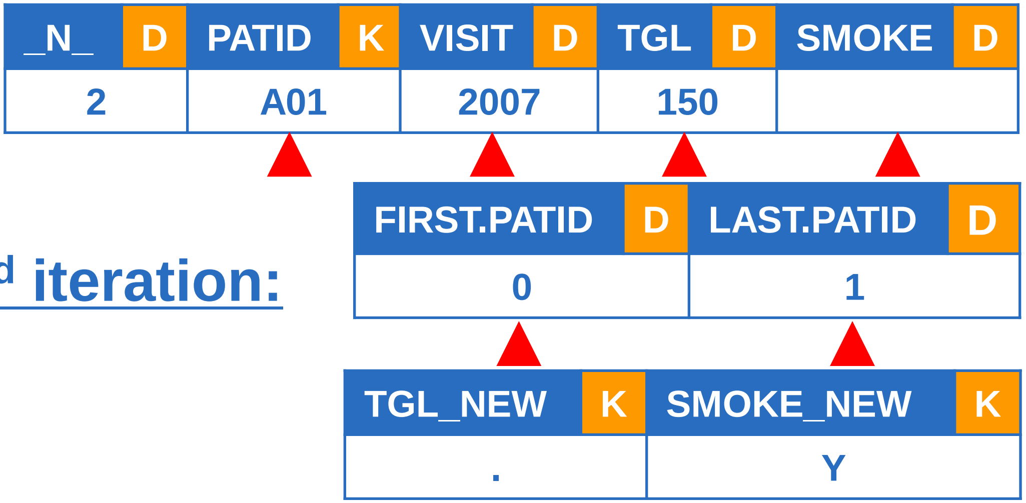
TGL_NEW K	SMOKE_NEW K
.	Y

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
   if not missing(smoke) then smoke_new=smoke;
   if last.patid;
run;
```

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

2nd iteration:



_N_ D	PATID K	VISIT D	TGL D	SMOKE D
2	A01	2007	150	

FIRST.PATID D	LAST.PATID D
0	1

TGL_NEW K	SMOKE_NEW K
.	Y

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
   if not missing(smoke) then smoke_new=smoke;
   if last.patid;
run;
```

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

_N_ D	PATID K	VISIT D	TGL D	SMOKE D
2	A01	2007	150	

2nd iteration:

FIRST.PATID D	LAST.PATID D
0	1

TGL_NEW K	SMOKE_NEW K
.	Y

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
   if not missing(smoke) then smoke_new=smoke;
   if last.patid;
run;
```

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

_N_ D	PATID K	VISIT D	TGL D	SMOKE D
2	A01	2007	150	

2nd iteration:

FIRST.PATID D	LAST.PATID D
0	1

TGL_NEW K	SMOKE_NEW K
150	Y

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
➡  if not missing(smoke) then smoke_new=smoke;
   if last.patid;
run;
```

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

_N_ D	PATID K	VISIT D	TGL D	SMOKE D
2	A01	2007	150	

2nd iteration:

FIRST.PATID D	LAST.PATID D
0	1

TGL_NEW K	SMOKE_NEW K
150	Y

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
   if not missing(smoke) then smoke_new=smoke;
➡  if last.patid;
run;
```

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

_N_ D	PATID K	VISIT D	TGL D	SMOKE D
2	A01	2007	150	

2nd iteration:

FIRST.PATID D	LAST.PATID D
0	1

TGL_NEW K	SMOKE_NEW K
150	Y

OBTAINING THE MOST RECENT NON-MISSING DATA WITHIN EACH BY-GROUP

```
data patients_single (drop= visit tgl smoke);
   set patients_sort;
   by patid;
   retain tgl_new smoke_new;
   if first.patid then do;
       tgl_new = .;
       smoke_new = " ";
   end;
   if not missing(tgl) then tgl_new=tgl;
   if not missing(smoke) then smoke_new=smoke;
   if last.patid;
run;
```

	PATID	VISIT	TGL	SMOKE
1	A01	2005	.	Y
2	A01	2007	150	
3	A02	2004	.	
4	A02	2005	200	N
5	A02	2006	210	N
6	A03	2005	.	Y
7	A04	2002	164	
8	A04	2004	170	Y
9	A04	2006	190	
10	A04	2007	.	N
11	A05	2005	189	

_N_	D	PATID	K	VISIT	D	TGL	D	SMOKE	D
2		A01		2007		150			

2nd iteration:

FIRST.PATID	D	LAST.PATID	D
0		1	

TGL_NEW	K	SMOKE_NEW	K
150		Y	

	PATID	TGL_NEW	SMOKE_NEW
1	A01	150	Y

Restructuring Data Sets from Long Format to Wide Format

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

→

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Restructuring Data Sets from Long Format to Wide Format

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

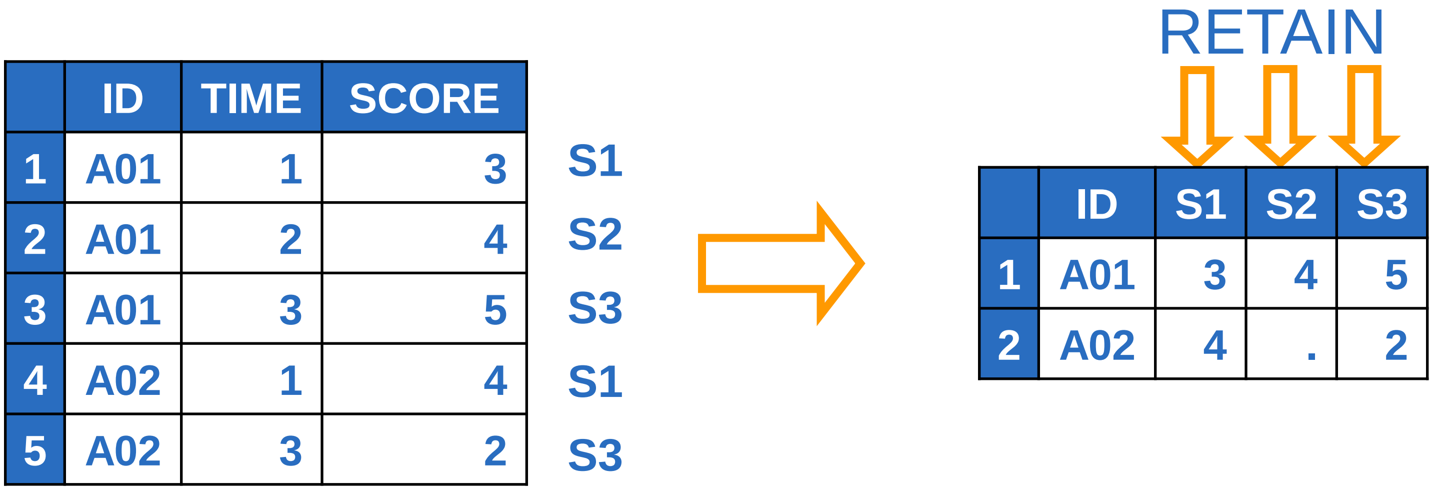
⇨

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

❖Reading 5 observations but only creating 2 observations

- ❑ You are *not* copying data from the PDV to the final dataset at each iteration
- ❑ You only need to generate one observation once all the observations for each subject have been processed

Restructuring Data Sets from Long Format to Wide Format



	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

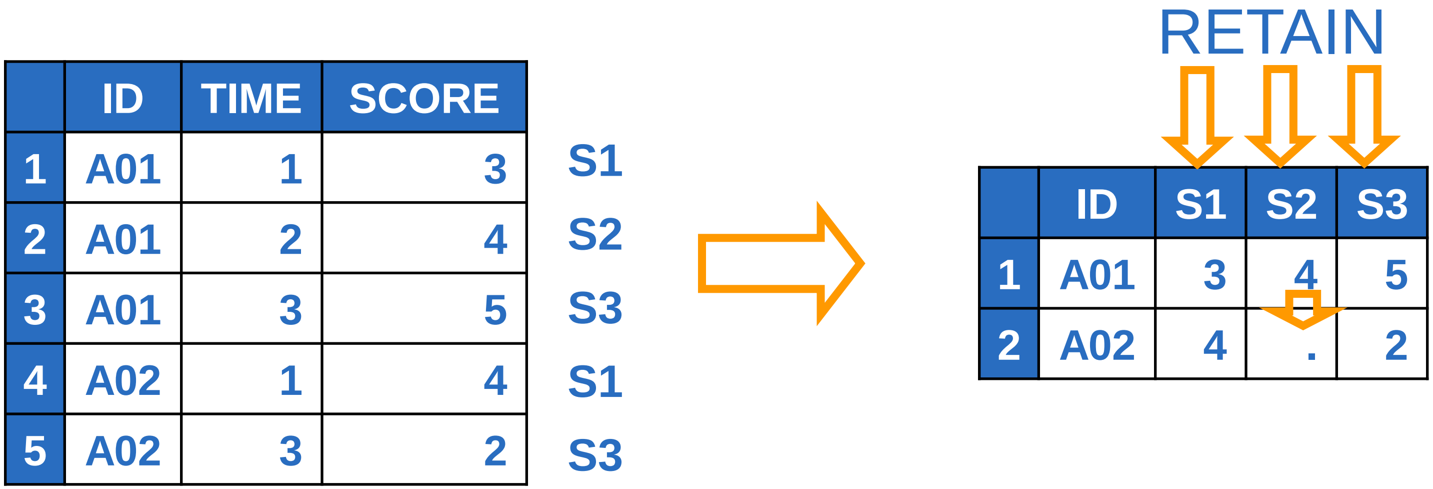
	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

- ❖Use BY-group processing: BY ID
  Output to the final data when LAST.ID = 1

- ❖SCORE → S1, S2 S3

```
if time = 1 then s1 = score;
else if time = 2 then s2 = score;
else s3 = score;
```

Restructuring Data Sets from Long Format to Wide Format



	ID	TIME	SCORE	
1	A01	1	3	S1
2	A01	2	4	S2
3	A01	3	5	S3
4	A02	1	4	S1
5	A02	3	2	S3

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

```
if first.id then do;
    s1 = .;  s2 = .; s3 = .;
end;
```

Restructuring Data Sets from Long Format to Wide Format

	ID	TIME	SCORE	
1	A01	1	3	S1
2	A01	2	4	S2
3	A01	3	5	S3
4	A02	1	4	S1
5	A02	3	2	S3

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

```
proc sort data=long;
    by id time;
run;
data wide (drop=time score);
    set long;
    by id;
    retain s1-s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
1		1		1				.		.		.		.		.	

1st iteration:

- ❖ _N_ ← 1

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
1	1	1		.	.	.	.	.

1st iteration:

- _N_ ← 1
- FIRST.ID ←1, LAST.ID ← 1

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
1		1		1				.		.		.		.		.	

1st iteration:

- _N_ ← 1
- FIRST.ID ←1, LAST.ID ← 1
- Other variables ← *missing*

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
1	1	0	A01	1	3	.	.	.

1st iteration:

❖ The SET statement copies the 1st observation →PDV

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
1		1		0		A01		1		3		.		.		.	

1st iteration:

- The SET statement copies the 1st observation →PDV
- FIRST.ID ←1 since this is the 1st observation for A01
- LAST.ID ← 0 since this is not the last observation for A01

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
1		1		0		A01		1		3		.		.		.	

1st iteration:

❖ Both BY and RETAIN statements are declarative statements

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
1		1		0		A01		1		3		.		.		.	

1st iteration:

- Since FIRST.ID =1, S1 – S3 are set to missing

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
1	1	0	A01	1	3	3	.	.

1st iteration:

❖ Since TIME = 1, S1 ← SCORE (3)

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
1		1		0		A01		1		3		3		.		.	

1st iteration:

- Since LAST.ID ≠1, (the subsetting IF statement is false), no further statements are processed for the current observation. SAS immediately returns to the beginning of the DATA step

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
2		1		0		A01		1		3		3		.		.	

2nd iteration:

❖ _N_ ↑2

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
2	1	0	A01	1	3	3	.	.

2nd iteration:

❖ _N_ ↑2

❖ FIRST.ID and LAST.ID are retained; they are automatic variables

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
2		1		0		A01		1		3		3		.		.	

2nd iteration:

❖ _N_ ↑2

❖ FIRST.ID and LAST.ID are retained; they are automatic variables

❖ ID, TIME, SCORE are retained; they are from input dataset

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
2	1	0	A01	1	3	3	.	.

2nd iteration:

- _N_ ↑2
- FIRST.ID and LAST.ID are retained; they are automatic variables
- ID, TIME, SCORE are retained; they are from input dataset
- S1, S2, and S3 are retained because of the RETAIN statement

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
2	0	0	A01	2	4	3	.	.

2nd iteration:

❖ The SET statement copies the 2nd observation to the PDV

Execution Phase of Program 4.5

```
data wide (drop=time score);
   set long;
   by id;
   retain s1 - s3;
   if first.id then do;
       s1 = .;  s2 = .; s3 = .;
   end;
   if time = 1 then s1 = score;
   else if time = 2 then s2 = score;
   else s3 = score;
   if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
2		0		0		A01		2		4		3		.		.	

2nd iteration:

- ❖ The SET statement copies the 2nd observation to the PDV
- ❖ FIRST.ID ← 0; this is not the first observation for A01
- ❖ LAST.ID ←0; this is not the last observation for A01 either

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
2	0	0	A01	2	4	3	.	.

2nd iteration:

- ❖ Since FIRST.ID ≠1, no execution

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
2	0	0	A01	2	4	3	4	.

2nd iteration:

❖Since TIME = 2, S2 ← SCORE (4)

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
2		0		0		A01		2		4		3		4		.	

2nd iteration:

❖ Since LAST.ID ≠1, (the subsetting IF statement is false), SAS immediately returns to the beginning of the DATA step

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
3		0		0		A01		2		4		3		4		.	

3rd iteration:

❖_N_ ↑3

❖The rest of the variables are retained

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
3		0		1		A01		3		5		3		4		.	

3rd iteration:

❖ The SET statement copies the 3rd observation →PDV

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
3		0		1		A01		3		5		3		4		.	

3rd iteration:

- ❖ The SET statement copies the 3rd observation →PDV
- ❖ FIRST.ID ← 0; this is not the first observation for A01
- ❖ LAST.ID ← 1; this is the last observation for A01

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
3	0	1	A01	3	5	3	4	.

3rd iteration:

❖ Since FIRST.ID ≠1, no execution

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
3		0		1		A01		3		5		3		4		5	

3rd iteration:

❖Since TIME = 3, S3 ← SCORE (5)

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
3		0		1		A01		3		5		3		4		5	

3rd iteration:

- Since LAST.ID = 1 (the subsetting IF statement is true), SAS continues to execute the remaining statements

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
3		0		1		A01		3		5		3		4		5	

3rd iteration:

- ❖ SAS reaches the end of the 3rd iteration,
  - ❑ The implicit OUTPUT statement executes
  - ❑ SAS returns to the beginning of the DATA step to begin the 4th iteration

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
4	0	1	A01	3	5	3	4	5

4th iteration:

❖ _N_ ↑4

❖ The rest of the variables are retained

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
4	1	0	A02	1	4	3	4	5

4th iteration:

❖ The SET statement copies the 4th observation →PDV

Execution Phase of Program 4.5

```
data wide (drop=time score);
 ➡ set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
4		1		0		A02		1		4		3		4		5	

4th iteration:

- ❖ The SET statement copies the 4th observation →PDV
- ❖ FIRST.ID ← 1; this is the first observation for A02
- ❖ LAST.ID ← 0; this is not the last observation for A02

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
4	1	0	A02	1	4	.	.	.

4th iteration:

❖ Since FIRST.ID = 1, S1 – S3 are set to *missing*

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
4	1	0	A02	1	4	4	.	.

4th iteration:

- ❖ Since TIME = 1, S1 ← SCORE (4)

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
4	1	0	A02	1	4	4	.	.

4th iteration:

- Since LAST.ID ≠1, (the subsetting IF statement is false), SAS immediately returns to the beginning of the DATA step

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
5		1		0		A02		1		4		4		.		.	

5th iteration:

- ❖_N_ ↑5
- ❖The rest of the variables are retained

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
5	0	1	A02	3	2	4	.	.

5th iteration:

❖ The SET statement copies the 5th observation →PDV

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_	D	FIRST.ID	D	LAST.ID	D	ID	K	TIME	D	SCORE	D	S1	K	S2	K	S3	K
5		0		1		A02		3		2		4		.		.	

5th iteration:

- ❖ The SET statement copies the 5th observation →PDV
- ❖ FIRST.ID ← 0; this is not the first observation for A02
- ❖ LAST.ID ← 1; this is the last observation for A02

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
5	0	1	A02	3	2	4	.	.

5th iteration:

- ❖ Since FIRST.ID ≠1, no execution

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
5	0	1	A02	3	2	4	.	2

5th iteration:

❖ Since TIME = 3, S3 ← SCORE (2)

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
5	0	1	A02	3	2	4	.	2

5th iteration:

- ❖ Since LAST.ID = 1 (the subsetting IF statement is true), SAS continues to execute the remaining statements

Execution Phase of Program 4.5

```
data wide (drop=time score);
    set long;
    by id;
    retain s1 - s3;
    if first.id then do;
        s1 = .;  s2 = .; s3 = .;
    end;
    if time = 1 then s1 = score;
    else if time = 2 then s2 = score;
    else s3 = score;
    if last.id;
run;
```

LONG:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

WIDE:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

_N_ D	FIRST.ID D	LAST.ID D	ID K	TIME D	SCORE D	S1 K	S2 K	S3 K
5	0	1	A02	3	2	4	.	2

5th iteration:

❖ SAS reaches the end of the 5th iteration,

❖ The implicit OUTPUT statement copies variables marked with (K) to the data set