How can the RETAIN statement be used in SAS? Can you provide some examples?

Authored by

stats writer

June 25, 2024

The RETAIN statement in SAS is used to hold or retain the value of a variable from one iteration of a data step to the next. This allows the value to be carried over and used in subsequent calculations or comparisons. The RETAIN statement is typically used in conjunction with the SET or MERGE statements to control the flow of data through a data step.

For example, if a data set contains monthly sales data for a certain product, the RETAIN statement can be used to retain the value of the total sales for each month and then calculate the cumulative sales for the entire year. This can be achieved by using the RETAIN statement to hold the value of the cumulative sales and then adding the current month's sales to it in each iteration of the data step.

Another example could be in a data set containing customer information, where the RETAIN statement can be used to retain the value of a customer's previous purchase date and then compare it to their current purchase date to calculate the time elapsed between purchases.

In summary, the RETAIN statement is a useful tool in SAS for managing and manipulating data values within a data step. It allows for the retention of values across iterations and can be applied in various scenarios to facilitate data analysis and processing.

Use the RETAIN Statement in SAS (With Examples)

You can use the RETAIN statement in SAS to specify some variable that should not have its value set to missing at the beginning of each iteration of a DATA step.

The RETAIN statement can be used for a variety of tasks in SAS, but here are the three most common use cases:

Case 1: Use RETAIN to Calculate a Cumulative Sum

data new_data;

```
set original_data;
retain cum_sum;
cum_sum + values_variable;
run;
```

Case 2: Use RETAIN to Calculate a Cumulative Sum by Group

```
data new_data;
set original_data;
by grouping_variable
retain cum_sum_by_group;
if first.grouping_variable then cum_sum_by_group = values_variable;
else cum_sum_by_group = cum_sum_by_group + values_variable;
run;
```

Case 3: Use RETAIN to Calculate a Cumulative Count by Group

```
data new_data;
set original_data;
by grouping_variable
```

```
retain count_by_group;
if first.grouping_variable then count_by_group = 1;
else count_by_group = count_by_group + 1;
run;
```

The following examples show how to use each case in practice with the following dataset in SAS that shows the sales made on consecutive days by different stores:

```
/*create dataset*/
data original_data;
input store $ sales;
datalines;
A 4
A 5
A 2
B 6
B 3
B 5
C 3
C 8
C 6
;
run;
```

```
/*view dataset*/
proc printdata=original_data;
```

Obs	store	sales
1	A	4
2	A	5
3	A	2
4	B	6
5	B	3
6	B	5
7	C	3
8	C	8
9	C	6

Example 1: Use RETAIN to Calculate a Cumulative Sum

The following code shows how to use the RETAIN statement to create a new column in the dataset that displays the cumulative sum of sales:

```
/*calculate cumulative sum of sales*/
data new_data;
set original_data;
retain cum_sales;
cum_sales+sales;
run;
```

/*view results*/

proc printdata=new_data;

Obs	store	sales	cum_sales
1	A	4	4
2	A	5	9
3	A	2	11
4	B	6	17
5	B	3	20
6	B	5	25
7	C	3	28
8	C	8	36
9	C	6	42

The new column called cum_sales contains the cumulative sum of values in the sales column.

For example:

Cumulative sum on row 1: 4Cumulative sum on row 2: 4 + 5 = 9Cumulative sum on row 3: 4 + 5 + 2 = 11

And so on.

In this example, the RETAIN statement set the variable called cum_sales to zero and then during each iteration of the DATA step, it simply added the new value of

sales to the running total of cum_sales.

Example 2: Use RETAIN to Calculate a Cumulative Sum by Group

The following code shows how to use the RETAIN statement to create a new column in the dataset that displays the cumulative sum of sales by store:

```
/*calculate cumulative sum of sales by store*/
data new_data;
set original_data;
by store;
retain cum_sales_by_store;
if first.store then cum_sales_by_store = sales;
else cum_sales_by_store = cum_sales_by_store + sales;
run;

/*view results*/
proc printdata=new_data;
```

Obs	store	sales	cum_sales_by_store
1	A	4	4
2	A	5	9
3	A	2	11
4	B	6	6
5	B	3	9
6	B	5	14
7	C	3	3
8	C	8	11
9	C	6	17

The new column called cum_sales_by_store contains the cumulative sum of values in the sales column, grouped by store.

In this example, the RETAIN statement set the variable called cum_sales_by_store to zero and then during each iteration of the DATA step, it checked if the value in the store column was the first occurrence of that particular value.

If it was the first occurrence, then the value of cum_sales_by_store was set to the value in the sales column. Else, the value in the sales column was added to the existing value for cum_sales_by_store.

Example 3: Use RETAIN to Calculate a Cumulative Count by Group

The following code shows how to use the RETAIN statement to create a new column in the dataset that displays the cumulative count of sales by store:

```
/*calculate cumulative count by store*/
data new_data;
set original_data;
by store
retain store_count;
if first.store then store_count = 1;
else store_count = store_count + 1;
run;

/*view results*/
proc printdata=new_data;
```

Obs	store	sales	store_count
1	A	4	1
2	A	5	2
3	A	2	3
4	B	6	1
5	B	3	2
6	B	5	3
7	C	3	1
8	C	8	2
9	C	6	3

The new column called store_count contains the cumulative count of each store.

In this example, the RETAIN statement set the variable called store_count to zero and then during each iteration of the DATA step, it checked if the value in the store column was the first occurrence of that particular value.

If it was the first occurrence, then the value of store_count was set to 1. Else, a value of 1 was added to the existing value for store_count.

The following tutorials explain how to perform other common tasks in SAS: