Learning to Remove Leading Zeros in SAS: A Step-by-Step Guide

Authored by

Mohammed looti

October 27, 2025

Dealing with data that contains superfluous leading zeros is a common challenge in data cleaning and preparation, particularly when importing source files where identifiers or numeric fields have been stored as text. In SAS, the most straightforward and effective technique for eliminating these leading zeros from a character variable involves leveraging the fundamental data type conversion capabilities of the software.

The primary method relies on the powerful **INPUT function**. By instructing SAS to read the character data as a number, the software automatically discards any non-significant zeros, as they hold no mathematical value in the resulting numerical representation. This process effectively transforms the variable into a numeric variable.

Understanding the Core SAS Solution: The INPUT Function

The core logic behind this conversion is simple yet highly effective. The **INPUT function** requires two arguments: the source character variable and an informat specifying how to read that variable. When SAS interprets the character string using a numeric informat (such as `comma9.`), it strips the leading zeros before storing the value as a floating-point number.

This approach ensures data integrity for calculation purposes, as leading zeros are often artifacts of text storage and not true components of the numerical value. The standard syntax for implementing this solution within a **DATA step** is as follows:

```
data new_data;
set original_data;
no_zeros = input(some_column, comma9.);
run;
```

This code snippet forms the foundation of the technique. The new variable, **no_zeros**, will contain the cleaned, numeric representation of the data originally held in **some_column**. The following comprehensive example illustrates how this syntax is applied to a real-world dataset scenario.

Practical Example: Converting Character to Numeric

Consider a scenario where we have a small SAS dataset recording sales figures. Crucially, the sales amounts were entered as character strings, resulting in inconsistent padding with leading zeros, which complicates aggregation or direct comparison. Our objective is to clean the **sales** column.

We first generate the sample dataset, where the **store** and **sales** variables are defined as character types (indicated by the dollar sign `$` in the `INPUT` statement). Notice the inconsistent formatting of sales figures, ranging from "055" to "00004302".

```
/*create dataset*/
data original_data;
input store $ sales $;
datalines;
A 055
B 145
C 199
D 0000443
E 0093
F 00004302
G 38
H 0055
;
run;

/*view dataset*/
proc print data=original_data;
```

The initial state of the dataset clearly shows the problematic leading zeros, which need to be addressed before any quantitative analysis can be performed accurately. The screenshot below depicts this original data structure:

Obs	store	sales
1	A	055
2	B	145
3	C	199
4	D	0000443
5	E	0093
6	F	00004302
7	G	38
8	H	0055

To execute the removal of leading zeros, we apply the **INPUT function** directly to the **sales** column within a new **DATA step**. This creates the new variable **no_zeros**, which will hold the cleaned numeric values.

```
/*remove leading zeros in sales column*/
```

```
data new_data;
set original_data;
no_zeros = input(sales, comma9.);
run;

/*view results*/
proc print data=new_data;
```

Upon reviewing the output generated by the `PROC PRINT` step, we can immediately observe the success of the operation. Every instance of a leading zero has been automatically stripped away, leaving only the significant digits in the **no_zeros** column.

Obs	store	sales	no_zeros
1	A	055	55
2	B	145	145
3	C	199	199
4	D	0000443	443
5	E	0093	93
6	F	00004302	4302
7	G	38	38
8	H	0055	55

It is crucial to note that the new **no_zeros** column, created by the simple application of the **INPUT function**, is now a standard numeric variable. This means it is immediately ready for mathematical operations, aggregation, and statistical modeling.

Advanced Solution: Maintaining Character Data Type Using PUT

While converting the variable to a numeric variable is often the desired outcome, there are scenarios where the cleaned data must remain a character variable. This requirement often arises when dealing with identification numbers, product codes, or other non-mathematical strings that simply cannot tolerate leading zeros but must retain their string properties for merging or specific reporting formats.

To achieve this, we employ a technique known as "round-tripping" or nested conversion, utilizing both the **INPUT function** and the **PUT function**. The process involves two steps executed simultaneously: first, the `INPUT` function converts the character string to a number (stripping the zeros), and second, the `PUT` function converts that resulting number back into a character string

using a specified format (e.g., `8.`).

This nested function ensures that the final variable, **no_zeros**, is stored as a character string, but its content is derived from the cleaned numeric value, thus successfully removing the leading zero padding.

```
/*remove leading zeros in sales column and keep as character*/
data new_data;
set original_data;
no_zeros = put(input(sales, comma9.), 8.);
run;

/*view results*/
proc print data=new_data;
```

The visual output of the data remains consistent with the previous example--the leading zeros are removed. However, the critical difference lies in the underlying structure of the **no_zeros** variable, which is now maintained as a character data type.

Obs	store	sales	no_zeros
1	A	055	55
2	B	145	145
3	C	199	199
4	D	0000443	443
5	E	0093	93
6	F	00004302	4302
7	G	38	38
8	H	0055	55

Verification of Data Type Using PROC CONTENTS

To definitively confirm the variable's status after utilizing the nested **PUT function** and **INPUT function** approach, we use the standard PROC CONTENTS procedure. This utility provides detailed metadata about the SAS data set, including the name, type, and length of every variable.

Running PROC CONTENTS is essential for verifying that the data manipulation accomplished the required data type preservation. If the type column shows 'Char', the character status was successfully retained; if it shows 'Num', the variable is numeric.

```
/*view data type of each variable in new dataset*/
proc contents data=new_data;
```

The output below confirms that when using the nested `PUT(INPUT())` structure, the resulting variable **no_zeros** is correctly identified as a Character variable (Type: Char), thus fulfilling the requirement to remove leading zeros while maintaining the string data type.

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	no_zeros	Char	8
2	sales	Char	8
1	store	Char	8

Summary of Techniques and Additional Resources

Removing leading zeros in SAS is a fundamental data manipulation task that is efficiently solved through conversion techniques. The choice between the simple **INPUT function** and the complex nested **PUT function** structure depends entirely on whether the resulting column must be a numeric variable (for calculations) or a character variable (for identification or reporting).

Key takeaways for managing this transformation:

Use `INPUT(variable, informat)` to convert a character variable to a numeric variable, automatically stripping leading zeros.
Use `PUT(INPUT(variable, informat), format)` to convert the character variable to numeric, strip zeros, and immediately convert back to a clean character string, thereby maintaining the character data type.
Always confirm the final data type using PROC CONTENTS, especially when dealing with data that originated as text.

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