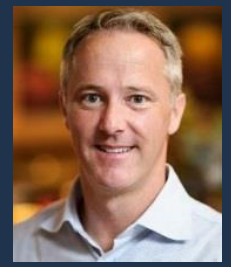
Sorting Through the Features of PROC SORT

Presented by Jeff Simpson

SAS Customer Loyalty



Proc Sort – Overview

The SORT procedure orders SAS data set observations by the values of one or more character or numeric variables.

The SORT procedure either replaces the original data set or creates a new data set.

```
proc sort data=class out=class_srt;
   by Descending weight;
run;

proc print data=class_srt;
run;
```

Note: **Proc sort** does not create a report.

Obs	Name	Sex	Age	Height	Weight
1	Alice	F	13	56.5	84.0
2	Carol	F	14	62.8	102.5
3	James	M	12	57.3	83.0
4	Janet	F	15	62.5	112.5
5	John	M	12	59.0	99.5
6	Judy	F	14	64.3	90.0
7	Mary	F	15	66.5	112.0
8	Robert	M	12	64.8	128.0
9	Thomas	M	11	57.5	85.0

Proc Sort – Basics

Dataset Replacement Options

Without the OUT= option, PROC SORT replaces the original data set with the sorted observations when the procedure executes without errors. When you specify the OUT= option using a new data set name, PROC SORT creates a new data set that contains the sorted observations.

Data Set Replacement Options	
Task	Options
implicit replacement of input data set	proc sort data=class;
explicit replacement of input data set	proc sort data=class out=class;
no replacement of input data set	proc sort data=class out=class_srt;

Proc Sort – Basics

Descending Option

What if you wanted to reverse the order of the BY statement? You can do this using the DESCENDING option.

```
proc sort data=class out=class_srt;
  by Descending weight;
run;

proc print data=class_srt;
run;
```

Obs	Name	Sex	Age	Height	Weight
1	Robert	M	12	64.8	128.0
2	Janet	F	15	62.5	112.5
3	Mary	F	15	66.5	112.0
4	Carol	F	14	62.8	102.5
5	John	M	12	59.0	99.5
6	Judy	F	14	64.3	90.0
7	Thomas	M	11	57.5	85.0
8	Alice	F	13	56.5	84.0
9	James	M	12	57.3	83.0

Proc Sort – Basics

OBS= and FIRSTOBS= Options

What if you wanted to only process only portion of the data.

```
proc sort data=class(Firstobs=2 Obs=10) out=class_srt;
  by Name;
run;
```

Obs	Name	Sex	Age	Height	Weight
1	Carol	F	14	62.8	102.5
2	James	M	12	57.3	83.0
3	Janet	F	15	62.5	112.5
4	John	M	12	59.0	99.5
5	Judy	F	14	64.3	90.0
6	Mary	F	15	66.5	112.0
7	Robert	M	12	64.8	128.0
8	Thomas	M	11	57.5	85.0

Proc Sort – Basics

Formatting your data

What if you wanted to **format** the values of M and F to Male and Female? You can do this using the **format** statement.

```
proc format;
  value $SEX
    'M'='Male'
    'F'='Female';
run;

proc sort data=class out=class_srt;
   FORMAT sex $SEX.;
   by Name;
run;
```

Obs	Name	Sex	Age	Height	Weight
1	Alice	Female	13	56.5	84.0
2	Carol	Female	14	62.8	102.5
3	James	Male	12	57.3	83.0
4	Janet	Female	15	62.5	112.5
5	John	Male	12	59.0	99.5
6	Judy	Female	14	64.3	90.0
7	Mary	Female	15	66.5	112.0
8	Robert	Male	12	64.8	128.0
9	Thomas	Male	11	57.5	85.0

Proc Sort – Basics

Where= , Drop= & Keep = and Rename=

- You can filter observations using Where= dataset option or the Where Statement.
- You can limit columns using the Keep = and/or Drop= dataset options
- You can Rename a variable

```
proc sort data=class(Where=(Age > 13)) out=class_srt (Drop=Weight
Rename=(Sex=gender));
  by Name;
run;
```

Obs	Name	gender	Age	Height
1	Carol	F	14	62.8
2	Janet	F	15	62.5
3	Judy	F	14	64.3
4	Mary	F	15	66.5

Proc Sort – Basics

NODUPKEY AND DUPOUT

NODUPKEY

checks for and eliminates observations with duplicate BY values

It eliminates multiple values of the by variable values and keeping only the first occurrence.

```
proc sort data=account DUPOUT=towns NODUPKEY;
  by town;
run;
```

DUPOUT= SAS-data-set

specifies the output data set to which duplicate observations are written.

Towns of Customers with Past-Due Accounts

Obs	Town	Company	Debt	AccountNumber
1	Apex	Paul's Pizza	83.00	1019
2	Garner	World Wide Electronics	119.95	1122
3	Holly Springs	Ice Cream Delight	299.98	2310
4	Morrisville	Strickland Industries	657.22	1675

Proc Sort – Basics

NODUPLICATES

Although still supported, the **NODUPLICATES** option is no longer documented.

The same results can be generated using

```
Proc SQL;
    select DISTINCT <list_of_variables>…
run;
```

And is more efficient.

Proc Sort – Advanced Functionality

NOUNIQUEKEY AND UNIQUEOUT

NOUNIQUEKEY
checks for and eliminates observations from the output data set that have a unique sort key. A sort key is unique when the observation containing the key is the only observation within a BY group.

The **UNIQUEOUT=** option can be used with the **NOUNIQUEKEY** option.

UNIQUEOUT= SAS-data-set
specifies the output data set for observations eliminated by the **NOUNIQUEKEY** option.

```
proc sort data=account NOUNIQUEKEY UNIQUEOUT=Unique_observations;
  by town;
run;
```

Proc Sort – Advanced Functionality

OVERWRITE

OVERWRITE
enables the input data set to be deleted **before** the replacement output data set of the same name is populated with observations.

CAUTION:
Use the **OVERWRITE** option only with a data set that is backed up or with a data set that you can reconstruct.

Using the **OVERWRITE** option can reduce disk space requirements.

Proc Sort – Advanced Functionality

PRESORTED

PRESORTED

before sorting, checks within the input data set to determine whether the sequence of observations is in order. Use the PRESORTED option when you know or strongly suspect that a data set is already in order according to the key variables that are specified in the BY statement.

This can eliminate the unnecessary overhead of the cost of sorting the data set.

```
Proc sort data=Class_sorted presorted;
          by name;
run;
```

```
NOTE: Sort order of input data set has been verified.
NOTE: There were 100000 observations read from the data set WORK.Class_SORTED.
NOTE: Input data set is already sorted, no sorting done.
```

(Warning: Using the PRESORTED option with ACCESS engines and DBMS data is not recommended)

Proc Sort – Advanced Functionality

DATECOPY AND EQUALS

DATECOPY

copies the SAS internal date and time at which the SAS data set was created and the date and time at which it was last modified before the sort to the resulting sorted data set.

```
proc sort data=Cities datecopy;
   by City;
proc print;
run;
```

Note: The DATECOPY option is only available when replacing the input data set.

Proc Sort – Advanced Functionality

EQUALS | NOEQUALS

EQUALS | NOEQUALS

specifies the order of the observations in the output data set. For observations with identical BY-variable values, EQUALS maintains the relative order of the observations from within the input data set to the output data set. NOEQUALS does not necessarily preserve this order in the output data set.

```
proc sort data=Employees Equals;
   by YearsWorked;
run;
```

Proc Sort – Advanced Functionality

EQUALS | NOEQUALS

EQUALS | NOEQUALS

5 421
5 336
1 209
1 564
3 711
3 343
4 212
4 616

Sort with EQUALS

Obs	YearsWorked	InsuranceID
1	1	209
2	1	564
3	3	711
4	3	343
5	4	212
6	4	616
7	5	421
8	5	336

Sort with NOEQUALS

Obs	YearsWorked	InsuranceID
1	1	209
2	1	564
3	3	343
4	3	711
5	4	212
6	4	616
7	5	421
8	5	336

The Modern Day Proc Sort

COLLATING SEQUENCE

PROC SORT <other options> <collating-sequence-option> <other option(s)>;

In the past it was based on **EBCDIC** or **ASCII**

The Modern Day Proc Sort

COLLATING SEQUENCE

PROC SORT <other options> <collating-sequence-option> <other option(s)>;

Collating sequence options

- Part of National Language Support (NLS)
- Collating performed by Translation Tables or rules(Linguistic)
- Only one Collating sequence per Proc Sort

The Modern Day Proc Sort

COLLATING SEQUENCE

National Collating Sequences of Alphanumeric Characters

DANISH: 0123456789aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZæÆøØåÅ
FINNISH: 0123456789aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZåÅäÄöÖ
ITALIAN: 0123456789aAàÀbBcCçÇdDeEéÉèÈfFgGhHiIìÌjJkKlLmMnNoOòÒpPqQrRsStTuUùÙvVwWxXyYzZ
NORWEGIAN: 0123456789aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZæÆøØåÅ
SPANISH: 0123456789aAáÁbBcCçÇdDeEéÉfFgGhHiIíÍjJkKlLmMnNñÑoOóÓpPqQrRsStTuUúÚüÜvVwWxXyYzZ
SWEDISH: 0123456789aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZåÅäÄöÖ

The Modern Day Proc Sort

COLLATING SEQUENCE

PROC SORT <other options> <collating-sequence-option> <other option(s)>;

Translation Tables

ASCII
sorts character variables using the ASCII collating sequence. You need this option only when you want to achieve an ASCII ordering on a system where EBCDIC is the native collating sequence (mainframe).

DANISH
sorts characters according to the Danish and Norwegian convention.

EBCDIC
sorts character variables using the EBCDIC collating sequence. You need this option only when you want to achieve an EBCDIC ordering on a system where ASCII is the native collating sequence.

The Modern Day Proc Sort

COLLATING SEQUENCE

PROC SORT <other options> <collating-sequence-option>
<other option(s)>;

FINNISH
sorts characters according to the Finnish and Swedish convention.

NATIONAL
sorts character variables using an alternate collating sequence, as defined by your installation, to reflect a country's National Use Differences. **To use this option, your site must define a customized national sort sequence.** Check with the SAS Installation Representative at your site to determine whether a customized national sort sequence is available.

NORWEGIAN
sorts characters according to the Danish and Norwegian convention.

The Modern Day Proc Sort

COLLATING SEQUENCE

PROC SORT <other options> <collating-sequence-option> <other option(s)>;

Translation Tables

REVERSE

sorts character variables using a collating sequence that is reversed from the normal collating sequence. It can be Operating System specific.

Restriction	Only one collating-sequence-option can be specified.
Interaction	Using REVERSE with the DESCENDING option in the BY statement restores the sequence to the normal order. ☺

The Modern Day Proc Sort

COLLATING SEQUENCE

PROC SORT <other options> <collating-sequence-option> <other option(s)>;

Translation Tables

SWEDISH
Sorts characters according to the Finnish and Swedish convention

The Modern Day Proc Sort

COLLATING SEQUENCE

PROC SORT <other options> <collating-sequence-option> <other option(s)>;

Translation Tables

User- Defined Translation Tables

You also create your own customized translation table

LINGUISTIC<(collating-options)>

ALTERNATE_HANDLING=SHIFTED

Controls the handling of variable characters like spaces, punctuation, and symbols

```
proc sort data=Cities out=Cities_Alt;
   by City;
run;
```

Obs	City
1	New Bern
2	New York
3	Newport News
4	Newton
5	Newtown

```
proc sort data=Cities out=Cities_Alt
sortseq=linguistic (ALTERNATE_HANDLING=SHIFTED);
   by City;
run;
```

Obs	City
1	New Bern
2	Newport News
3	Newton
4	Newtown
5	New York

LINGUISTIC<(collating-options)>

CASE_FIRST=specifies the order of uppercase and lowercase letters. This argument is valid for only TERTIARY, QUATERNARY, or IDENTICAL levels

Value	Description
UPPER	Sorts uppercase letters first, then the lowercase letters.
LOWER	Sorts lowercase letters first, then the uppercase letters.

LINGUISTIC<(collating-options)>

COLLATION= specifies character ordering. It takes in to consideration language differences

NUMERIC_COLLATION= Specifies if numbers in text are treated as numbers or text

STRENGTH= Determines how to differentiate between numbers, characters and punctuation

LINGUISTIC<(collating-options)>

STRENGTH= Determines how to differentiate between numbers, characters and punctuation

- PRIMARY - differences between base characters (default)
- SECONDARY- differences in the accents in the characters
- TERTIARY - Upper and lowercase differences in characters, also for detecting numerals in text string
- QUATERNARY - distinguish words with and without punctuation
- IDENTICAL – Beyond my pay grade

LINGUISTIC<(collating-options)>

Working with special names with uppercasing and punctuation in the middle of the values

Macdonald

MacDonald

McDonald

O'Brien

Obrien

O Brien

LINGUISTIC<(collating-options)>

Working with special names with Uppercasing and punctuation in the middle of the values

```
proc sort data=test out=testout;
  by Name;
  run;
```

Obs	name
1	MacDonald
2	Macdonald
3	McDonald
4	O Brien
5	O'Brien
6	Obrien

LINGUISTIC<(collating-options)>

Working with special names with Uppercasing and punctuation in the middle of the values

```
proc sort data=test out=testout2 sortseq=linguistic(strength=primary);
  by Name;
  run;
```

Obs	name
1	Macdonald
2	MacDonald
3	McDonald
4	O Brien
5	O'Brien
6	Obrien

LINGUISTIC<(collating-options)>

What if you were processing numeric month values that were stored as character fields 1-12.

The default Proc Sort would do the following:

```
proc sort data = us_trade2009   sortseq=linguistic (numeric_collation=off);
by month;
run;
```

This is the default value.

Default Effect of PROC Sort on character variable MONTH

month	amount
1	$161,528
10	$170,396
11	$174,270
12	$180,485
2	$153,286
3	$154,002
4	$152,565
5	$150,696
6	$155,009
7	$162,702
8	$161,421
9	$169,343

LINGUISTIC<(collating-options)>

What if you were processing numeric month values that were stored as character fields 1-12.

The default Proc Sort would do the following:

```
proc sort data = us_trade2009 sortseq=linguistic
(numeric_collation=on);
by month;
run;
```

Effect of "numeric_collation=on" for character variable MONTH

month	amount
1	$161,528
2	$153,286
3	$154,002
4	$152,565
5	$150,696
6	$155,009
7	$162,702
8	$161,421
9	$169,343
10	$170,396
11	$174,270
12	$180,485

LINGUISTIC<(collating-options)>

Solving the street address issue with sorting

- 14 12th Street
- 14 121st Street
- 141 12th Street Apt 1
- 141 12th Street Apt 11
- 141 12th Street Apt 2
- 111 121st Street
- 1111 12th Street
- 1411 12th Street

LINGUISTIC<(collating-options)>

Solving the street address issue with sorting

```
Proc Sort Data =addresses out=addresses_num;
  by address;
run;
```

Obs	Address
1	111 121st Street
2	1111 12th Street
3	14 121st Street
4	14 12th Street
5	141 12th Street Apt 1
6	141 12th Street Apt 11
7	141 12th Street Apt 2
8	1411 12th Street

LINGUISTIC<(collating-options)>

Solving the street address issue with sorting

```
Proc Sort Data =addresses out=addresses_num
sortseq=linguistic(numeric_Collation=on);
  by address;
run;
```

Obs	Address
1	14 12th Street
2	14 121st Street
3	111 121st Street
4	141 12th Street Apt 1
5	141 12th Street Apt 2
6	141 12th Street Apt 11
7	1111 12th Street
8	1411 12th Street

Proc Sort – Summary

The SORT procedure has been around for almost 40 years. Most SAS programs have at least one sort procedures, if not more.

The SAS Proc Sort procedure has continued to involve and is no longer your father's basic sort functionality. As organizations and reporting needs are becoming more and more global, so have the needs for more advanced sorting.

Much of the functionality replaces custom data step and macro programming and makes sorting highly versatile.

Sorting Through the Features of PROC SORT

Thank you!