PROC SORT

This procedure rearranges observations in a dataset according to values of the variables in the BY statement.

The order for numeric variables is from lowest to highest.
The order for character values is the ASCII collating sequence.

ASCII (American Standard Code for Information Interchange)

(Blank) ! " # $ % & ' ( ) * + , - /
0 1 2 3 4 5 6 7 8 9 : ; < = > ? @
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
[\ ] ^ _
a b c d e f g h I j k l m n o p q r s t u v w x y z
{ 1 } ~

Syntax:-

```
Proc sort <collating sequence options> < other options >;
By < descending> variables-1 < descending> variables-2
………………..< descending> variables–n;
Run;
```

Options:-

Data= dataset name

Specify the input dataset to be sorted.

```
PROC SORT DATA=DEMOGRAPHIC_DATA;
BY ZONE SEX;
RUN;
```

Out=dataset name

Specify the output data set

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2;
BY EMP_NAME SEX;
RUN;
```

```
PROC SORT DATA=DEMOGRAPHIC_DATA
(keep= EMP_ID EMP_NAME SEX EDUCATION INCOME) OUT=DEMOGRAPHIC_DATA2;
BY EMP_NAME SEX;
RUN;
```

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2
(keep= EMP_ID EMP_NAME SEX AGE EDUCATION INCOME Rename=(SEX=GENDER));
BY EMP_NAME SEX;
RUN;
```

Nodupkey

Eliminates values with duplicate by variables.

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA3 NODUPKEY;
BY EMP_NAME;
RUN;
```

Duplicates or Noduprec or Nodup

Checks and eliminates duplicate observations after sort.
Delete duplicates if entire observations are duplicate.

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA4 NODUPREC;
BY EMP_NAME;
RUN;
```

Difference between Nodupkey and Nodup

Nodupkey eliminates values with duplicate by values but Nodup check the entire Observations for duplicate values and eliminate, not just the by values, eliminate entire row.

Dupout=Dataset_Name;

Create a dataset with duplicate values.

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA4
NODUPKEY DUPOUT=DEMO1;
BY EMP_NAME;
RUN;
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA4
NODUPREC DUPOUT=DEMO2;
BY EMP_NAME;
RUN;
```

Descending

Rearranges the observations in descending order according to values of the by variable. By default it is Ascending order.

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2A;
BY DESCENDING EMP_NAME SEX;
RUN;
```

Reverse

Reverse the collation order for character variables

Not for numeric, if you specify for Numeric variables it sorts ascending only use descending if you want reverse order for numeric.

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2 REVERSE;
BY SEX ;
RUN;
```

```
PROC SORT DATA=DEMOGRAPHIC_DATA
OUT=DEMOGRAPHIC_DATA3 REVERSE;
BY AGE;
RUN;
```

```
PROC SORT DATA=DEMOGRAPHIC_DATA
OUT=DEMOGRAPHIC_DATA4 ;
BY DESCENDING AGE;
RUN;
```

To sort all the variables in dataset

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2A;
BY _all_;
RUN;
```

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2A;
BY descending _all_;
RUN;
```

To sort all character variables in dataset

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2A;
BY _char_;
RUN;
```

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2A;
BY descending _char_;
RUN;
```

To sort all numeric variables in dataset

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2A;
BY _numeric_;
RUN;
```

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2A;
BY descending _numeric_;
RUN;
```

By default it sort the data in ASCII if you want to change

SORTSEQ= *collating-sequence*

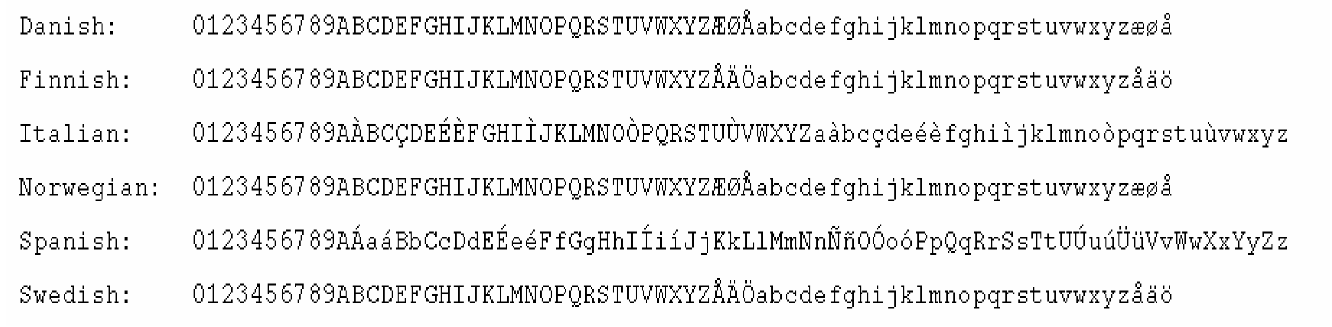
Specify any of the collating sequences listed above (ASCII, EBCDIC, DANISH, FINNISH, ITALIAN, NORWEGIAN, POLISH, SPANISH, SWEDISH, or NATIONAL)

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2 SORTSEQ=DANISH;
BY EMP_NAME SEX;
RUN;
```

Collating sequence are

ASCII / EBCICIC / DANISH / FINNISH / ITALIAN / NORWEGIAN / POLISH / REVERSE / SPANISH / SWEDISH

See below table is collating sequence how sorts alphanumeric values.



Danish:	0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÆØÅabcdefghijklmnopqrstuvwxyzæøå
Finnish:	0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖabcdefghijklmnopqrstuvwxyzåäö
Italian:	0123456789AÀBCÇDEÉÈFGHIÌJKLMNOÒPQRSTUÙVWXYZaàbcçdeéèfghiìjklmnoòpqrstuùvwxyz
Norwegian:	0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÆØÅabcdefghijklmnopqrstuvwxyzæøå
Spanish:	0123456789AÁaáBbCcDdEÉeéFfGgHhIÍiíJjKkLlMmNnÑñOÓoóPpQqRrSsTtUÚuúÜüVvWwXxYyZz
Swedish:	0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖabcdefghijklmnopqrstuvwxyzåäö

OVERWRITE

Delete the input data set before the replacement output data set is populated.

Observe below log.

```
96   PROC SORT DATA=DEMOGRAPHIC_DATA
97   OUT=DEMOGRAPHIC_DATA2 OVERWRITE;
98   BY NAME SEX;
99   RUN;

NOTE: There were 19 observations read from the data set WORK.DEMOGRAPHIC_DATA.
NOTE: The data set WORK.DEMOGRAPHIC_DATA2 has 19 observations and 5 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
100
101  PROC SORT DATA=DEMOGRAPHIC_DATA OVERWRITE;
102  BY  SEX;
103  RUN;

NOTE: The data set WORK.DEMOGRAPHIC_DATA has 19 observations and 5 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.03 seconds
      cpu time            0.01 seconds
```

DATECOPY

Sort a SAS data set without changing the created and modified dates
Copies the SAS internal date and time when the SAS data set was created

```
DATA DEMOGRAPHIC_DATA;
SET SASHELP.CLASS;
X=HEIGHT-WEIGHT;
RUN;
```

```
PROC SORT DATA=DEMOGRAPHIC_DATA DATECOPY;
BY SEX;
RUN;
```

```
101  PROC SORT DATA=DEMOGRAPHIC_DATA
102  OUT=DEMOGRAPHIC_DATA2 DATECOPY;
103  BY SEX;
104  RUN;

NOTE: There were 19 observations read from the data set WORK.DEMOGRAPHIC_DATA.
WARNING: The DATECOPY option is only available when replacing the input data set. The
DATECOPY option is being ignored.
NOTE: The data set WORK.DEMOGRAPHIC_DATA2 has 19 observations and 6 variables.
```

Improving performance for sorting data
SORTSIZE=*memory-specification*
Specifies the maximum amount of memory that is available for PROC SORT.

For ***memory-specification*** are as follows:

MAX :- specifies that all available memory can be used.
N :- specifies the amount of memory in bytes, where *n* is a real number.
nK :- specifies the amount of memory in kilobytes, where *n* is a real number.
nM :- specifies the amount of memory in megabytes, where *n* is a real number.
nG :- specifies the amount of memory in gigabytes, where *n* is a real number.

The SORTSIZE= option specifies the amount of memory that is available for PROC SORT to use and can reduce the amount of swapping that the SAS System must do to sort the data set. A value of 2 M is optimal for all memory configurations. If your machine has more than 12 MB of physical memory and you are sorting large data sets, setting this option to a value between 2 MB and 8 MB may improve performance. If PROC SORT needs more memory than you specify, it creates a temporary utility file in your WORK folder to complete the sort.
If you do not use the SORTSIZE= option, PROC SORT uses the value of the SORTSIZE system option. The default value of the SORTSIZE system option is 2M.

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2A SORTSIZE=8MB;
BY DESCENDING EMP_NAME SEX;
RUN;
```

Tagsort

When we have space issues in SAS Server/CPU (When data memory is more than CPU/Server memory) we can use this Tagsort

Stores only the BY variables and the observation number in temporary files.
Reduce temporary disk usage

The TAGSORT option in the PROC SORT statement is useful when there may not be enough disk space to sort a large SAS data set. When you specify the TAGSORT option, only sort keys (that is, the variables specified in the BY statement) and the observation number for each observation are stored in the temporary files. The sort keys, together with the observation number, are referred to as **tags**. At the completion of the sorting process, the tags are used to retrieve the records from the input data set in sorted order. Thus, in cases where the total number of bytes of the sort keys is small compared with the length of the record, temporary disk use is reduced considerably. You should have enough disk space to hold another copy of the data (the output data set) or two copies of the tags, whichever is greater. Note that while using the TAGSORT option may reduce temporary disk use, the processing time may be much higher. However, on PCs with limited available disk space, the TAGSORT option may allow sorts to be performed in situations where they would otherwise not be possible.

```
PROC SORT DATA=DEMOGRAPHIC_DATA OUT=DEMOGRAPHIC_DATA2A Tagsort;
BY DESCENDING EMP_NAME SEX;
RUN;
```

Dataset options with Proc sort.

```
Data demo;
Input patient 1-2 sex $ 3-4 age 5-7 ps 8-9;
Datalines;
1 F 45 0
4 M 63 2
3 M 57 1
5 F 72 3
2 F 39 0
3 M 57 1
4 M 63 0
;
Run;
```

```
Proc sort data=demo(keep=patient age) out=demo1;
By patient;
Run;
```

```
Proc sort data=demo out=demo2(rename=(patient=pt));
By patient;
Run;
```

```
Proc format;
Value $SEX 'F'='Female'
'M'='Male';
Run;

Proc sort data=demo out=demo3;
Format sex $SEX.;
By patient;
Run;

Proc sort data=demo out=demo4;
Label ps='Performance Status'
Age='Age at Diagnosis';
By patient;
Run;

Proc print data=demo4 label;
Run;

Proc print data=demo4;
Run;

Proc sort data=demo(where=(age>50)) out=demo5;
By patient;
Run;

Proc sort data=demo out=demo6;
Where age>50;
By patient;
Run;

Proc sort data=demo(firstobs=3 obs=5) out=demo7;
By patient;
Run;
```

How to sort entire dataset in reverse order

#way1

```
DATA DS;
DO K=NOBS TO 1 BY -1;
SET SASHELP.CLASS NOBS=NOBS POINT=K;
OUTPUT;
END;
STOP;
RUN;
```

#way2

```
Data ds;
Set sashelp.class;
X=_n_;
Run;
Proc sort data=ds;
By descending x;
Run;
```

How to arrange the data except ascending and descending (In required order)

```
Data ds1;
Retain date amount x y no;
Length y $10.;
Set sashelp.buy;
x=Weekday(date);
If x=1 then y='Sunday';
Else if x=2 then y='Monday';
Else if x=3 then y='Tuesday';
Else if x=4 then y='Wednesday';
Else if x=5 then y='Thursday';
Else if x=6 then y='Friday';
Else y='Saturday';
If y='Sunday' then no=1;
Else if y='Monday' then no=2;
Else if y='Tuesday' then no=3;
Else if y='Wednesday' then no=4;
Else if y='Thursday' then no=5;
Else if y='Friday' then no=6;
Else no=7;
Run;

Proc sort data=ds1 out=ds2(drop=x no);
By no;
Run;
```

How to find second maximum salary using sort.

```
Data ds1;
Infile datalines;
Input id name$ sal;
Datalines;
101 abc 5000
102 def 6000
103 jkl 7000
104 mno 4000
105 xyz 8000
;
Run;
Proc sort data=ds1 out=ds2;
By descending sal;
Run;
Data ds3;
Set ds2;
if _n_=2;
Run;
```

or

```
Data ds4;
Set ds2 (firstobs=2 obs=2);
Run;
```

or

```
Data ds5;
Slice=2;
Set ds2 point=slice;
Output;
Stop;
Run;
```

An important documents which helps to understand the power of PROC SORT

The SORT Procedure - Beyond the Basics.p
The SORT Procedure - Beyond the Basics 2
Rearranging Your Data Using PROC SOI
A Different View of PROC SORT.pdf
Dup, Dedup, DUPOUT - New in PR(
Y'S NOTSORTED OPTION BY'S NOTSOI

Useful Options in the SAS® Sort Procedure
Using the 'PRESORTED' Option
SAS Learning Module.htm
PROC SORT Statement.htm
Proc sort nodup.htm

Frequently asking interview questions about Proc Sort

1) Syntax of proc sort ?
2) What is difference between Nodupkey and Noduprec ?
3) How can you find out third max salary from EMP table using proc sort?
4) How can you create a dataset with deleted duplicate records code it ?
5) What is Sortseq option in proc sort ?
6) How can you increase performance of sorting dataset?
7) What is multi threaded sorting ?
8) How do I sort the values of two related arrays?
9) How can you delete duplicate observations from sas dataset using proc sort?
10) How can you delete duplicate observations from sas dataset without using proc sort?
11) I have a huge data set. Is there a quick way for me to sort my whole data out?
12) Can we perform merge without sorting datasets? How?
13) How would you check the uniqueness of a data set?
14) How would you keep sas from overlaying the sas dataset with it is sorted version?