

# Generation Why: How Generation Data Sets Can Help

**Lisa Eckler, Lisa Eckler Consulting**

# Introduction

- ❑ We tend to work with many data sets
- ❑ It's wise to keep back-ups of our data sets
- ❑ It's good practice to validate our data (or coding changes) by comparing to older versions

# Generation Data Sets

- ❑ **Definition: one in a related series of SAS data sets in a single library, sharing a single name data set name, distinguished by a version number**
- ❑ **The series may also be described as a Generation Group**

# How Do They Behave?

Here's how data gets aged through a series, where GENMAX = 3:

Program action	Name of newest data	Relative generation(-1)	Relative generation (-2)	What happens
Create first generation	EXAMPLE			New EXAMPLE is the only data set in the series.
Create second generation	EXAMPLE	EXAMPLE#001		EXAMPLE is aged to relative generation -1, also accessible as absolute generation 1. New EXAMPLE is created.
Create third generation	EXAMPLE	EXAMPLE#002	EXAMPLE#001	EXAMPLE#001 is aged to relative generation -2, also accessible as absolute generation 1. EXAMPLE is aged to relative generation -1, also accessible as absolute generation 2. New EXAMPLE is created.
Create fourth generation	EXAMPLE	EXAMPLE#003	EXAMPLE#002	EXAMPLE#001 is dropped. EXAMPLE#002 is aged to relative generation -2, also accessible as absolute generation 2. EXAMPLE is aged to relative generation -1, also accessible as absolute generation 3 New EXAMPLE is created.

# How To Define

Specify GENMAX = <n> when creating a SAS data set to define it as a generation data set AND establish <n> as the maximum number of generations to keep.

```
data MY_DATA(genmax = 24);
```

# How To Use

- ❑ Specify GENNUM on a SET statement or any input data set reference in a PROC
- ❑ The value for GENNUM can be an absolute reference (positive integer) or a relative reference (negative integer)
- ❑ Refer to the base data set name alone to use or create the latest generation

# How To Use

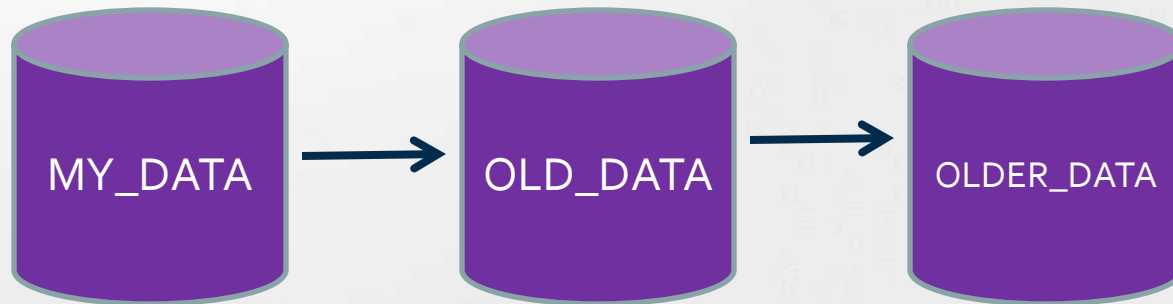
## Example code snippets:

```
proc sort data = MY_DATA(gennum = 27)  
    out = MY_DATA_SORTED;
```

```
data NEW_DATA;  
    set MY_DATA(gennum = -1);
```

```
proc print data = MY_DATA;
```

# To Create Backups...



# GENMAX Can Help

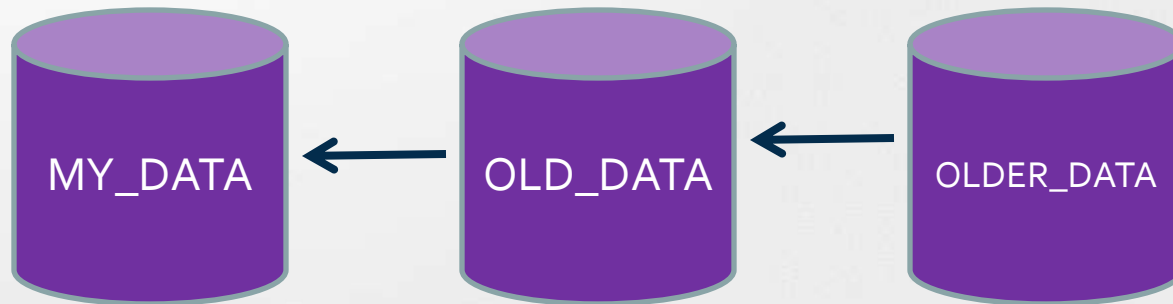


```
data OLDER_DATA;  
    set OLD_DATA;  
  
run;  
data OLD_DATA;  
    set MY_DATA;  
  
run;  
data MY_DATA;  
    ** then create latest MY_DATA ... **;  
  
run;
```

**OR...**

```
data MY_DATA(genmax=3);  
    ** create latest MY_DATA ... **;  
  
run;
```

# To Restore Data...



# GENNUM Can Help



```
data MY_DATA;  
    set OLD_DATA;  
run;
```

```
data OLD_DATA;  
    set OLDER_DATA;  
run;
```

**OR...**

# GENNUM Can Help



```
data MY_DATA;  
    set MY_DATA(gennnum = -1);  
run;
```

**OR...**

**With access to data sets via Explorer or similar tool, manually delete latest MY\_DATA and then refer to MY\_DATA, which will be the previous version restored as current.**

# Why To Use

- Maintain automatic data back-ups
- Allows for easy data recovery
- Prior version of data easily accessible for validation
- No need to manually delete old data
- Can gather series of data sets for reporting
- All of this with very simple coding

# Conclusion

**Using the GENNUM and GENMAX options, SAS generation data sets can help you**

- Save development time**
- Simplify your code**
- Maintain automatic back-ups or snapshots of your data**

# Thank You!



If you have questions or  
comments...

[lisa.eckler@sympatico.ca](mailto:lisa.eckler@sympatico.ca)