SAS® GLOBAL FORUM 2018

USERS PROGRAM

Copy That!

Using SAS® to Create Directories and Duplicate Files

Nicole Ciaccia, Educational Testing Service

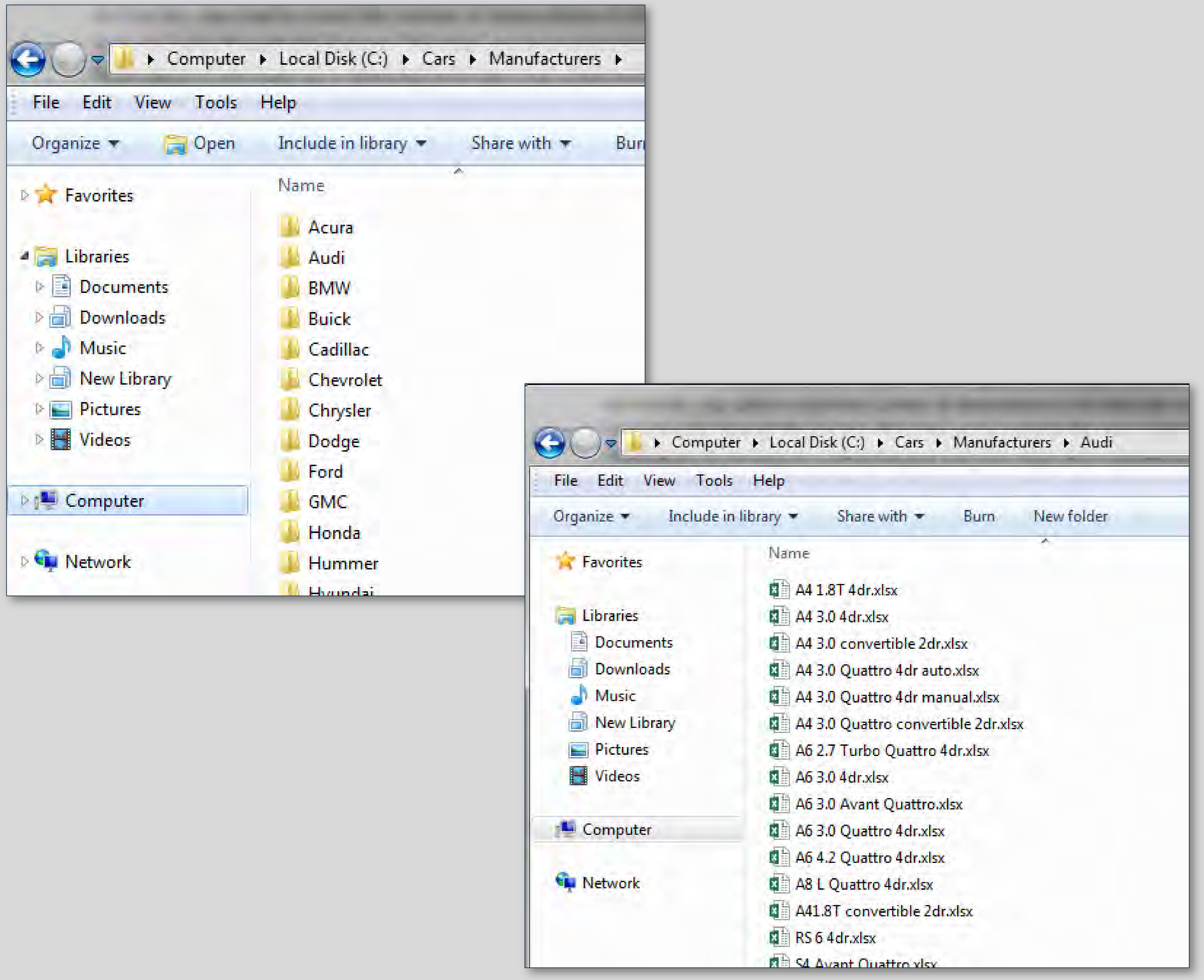
April 8 – 11 | Denver, CO
#SASGF

Copy That! - Using SAS® to Create Directories and Duplicate Files

Nicole Ciaccia
Educational Testing Service

Overview

- **Abstract**
- **Creating a Single Folder or Directory**
    - DLCREATEDIR
    - DCREATE
    - X md
- **Creating Multiple Folders or Directories**
- **Duplicating Files**
    - X copy
    - System Copy
    - Batch File
- **Duplicating Multiple Files**
    - Batch File
    - System Copy
- **Conclusion**


Computer ▸ Local Disk (C:) ▸ Cars ▸ Manufacturers ▸
File Edit View Tools Help
Organize Open Include in library Share with Bur
Favorites
Libraries
Documents
Downloads
Music
New Library
Pictures
Videos
Computer
Network
Name
Acura
Audi
BMW
Buick
Cadillac
Chevrolet
Chrysler
Dodge
Ford
GMC
Honda
Hummer
Hyundai
Computer ▸ Local Disk (C:) ▸ Cars ▸ Manufacturers ▸ Audi
File Edit View Tools Help
Organize Include in library Share with Burn New folder
Favorites
Libraries
Documents
Downloads
Music
New Library
Pictures
Videos
Computer
Network
Name
A4 1.8T 4dr.xlsx
A4 3.0 4dr.xlsx
A4 3.0 convertible 2dr.xlsx
A4 3.0 Quattro 4dr auto.xlsx
A4 3.0 Quattro 4dr manual.xlsx
A4 3.0 Quattro convertible 2dr.xlsx
A6 2.7 Turbo Quattro 4dr.xlsx
A6 3.0 4dr.xlsx
A6 3.0 Avant Quattro.xlsx
A6 3.0 Quattro 4dr.xlsx
A6 4.2 Quattro 4dr.xlsx
A8 L Quattro 4dr.xlsx
A41.8T convertible 2dr.xlsx
RS 6 4dr.xlsx
S4 Avant Quattro.xlsx

Nicole Ciaccia
Educational Testing Service

Abstract

- Whether it is due to a client's request or for organizational purposes, sometimes the creation of several directories is required.
- Along those same lines, business requirements may dictate that the same document or checklist be filled out for each individual client or product that a company manages.
- SAS has a variety of system options, functions and commands that can be used to assist in file management and organization.
- These tools can be leveraged in order to automate such repetitive, routine tasks and save the user a great deal of time.

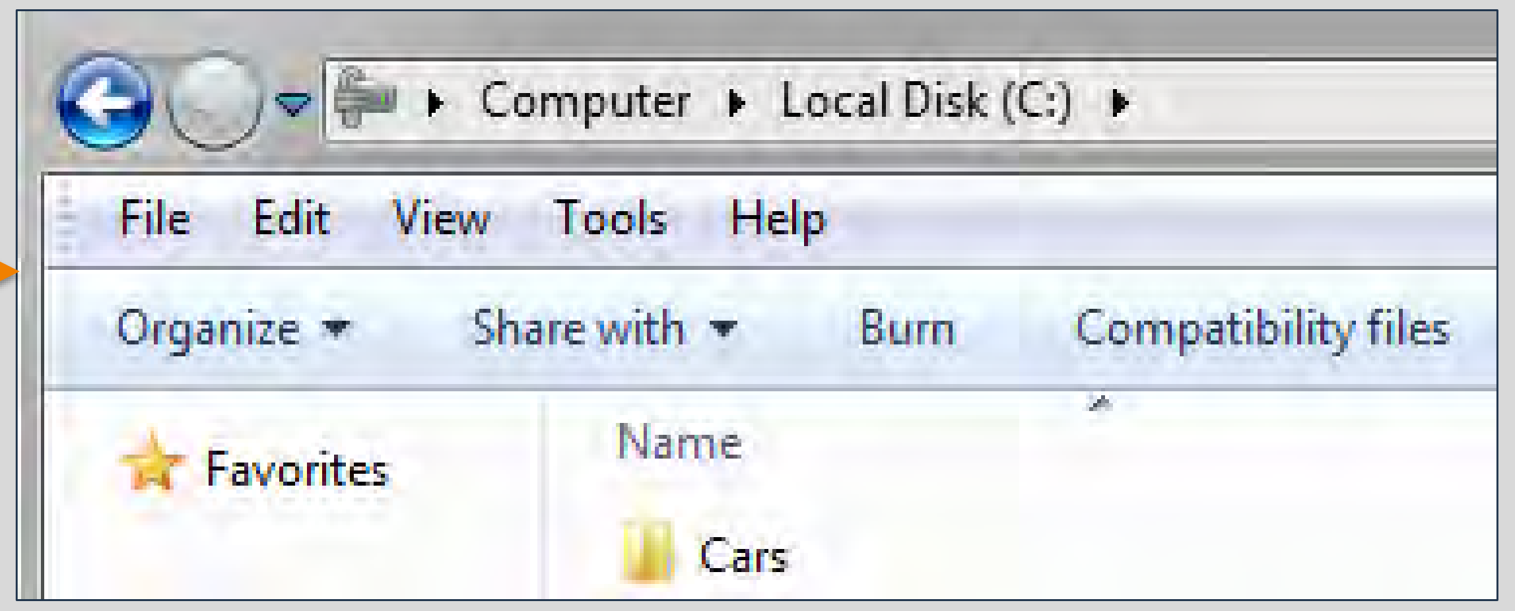
Copy That! - Using SAS® to Create Directories and Duplicate Files

Nicole Ciaccia
Educational Testing Service

Creating a Single Folder or Directory

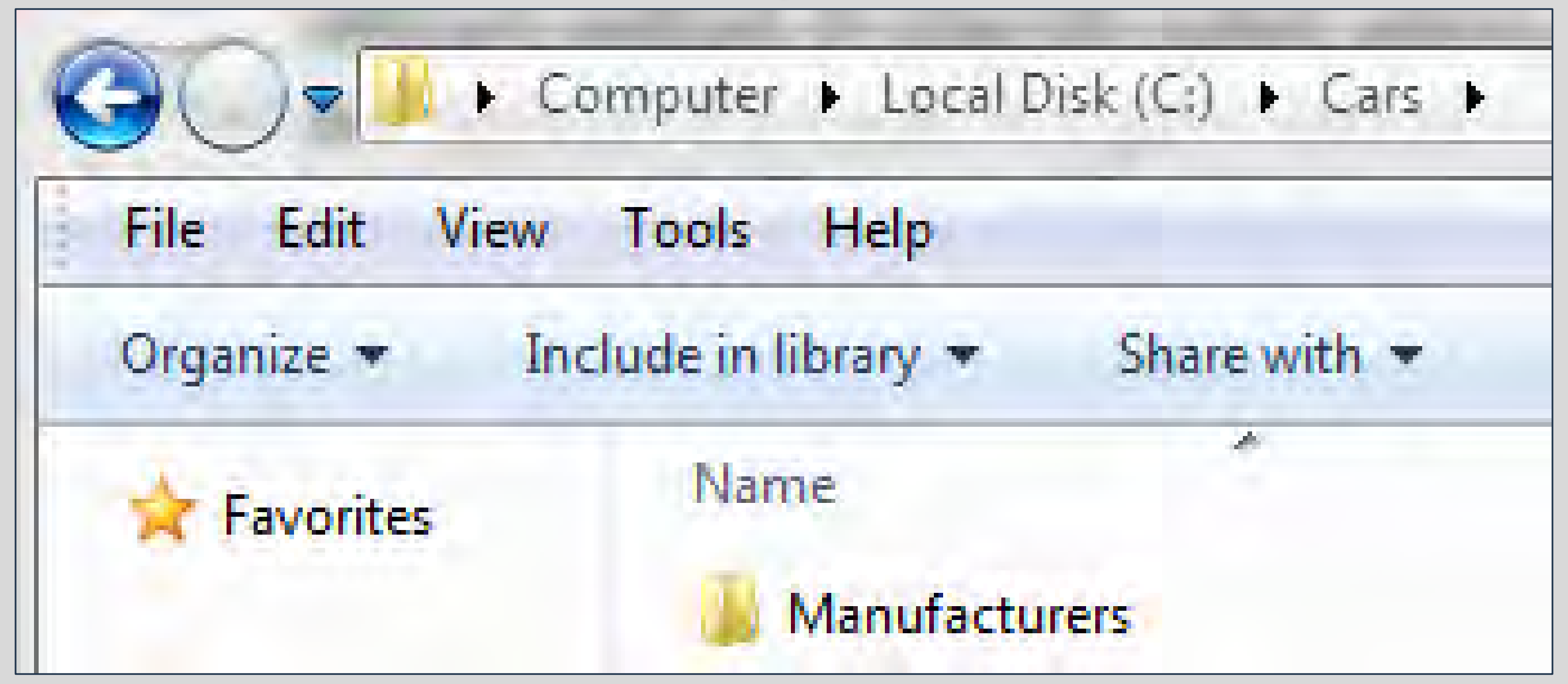
- **DLCREATEDIR (a system option)**
  - Used with a LIBNAME statement, a folder is created with 2 lines of code:

```
options dlcreatedir;
libname Folder "C:\Cars";
```


Computer ▸ Local Disk (C:) ▸
File Edit View Tools Help
Organize ▾ Share with ▾ Burn Compatibility files
Favorites
Name
Cars

  - If subfolders are needed, multiple specifications in the LIBNAME statement can be used:

```
options dlcreatedir;
libname Folder ("C:\Cars", "C:\Cars\Manufacturers");
```


Computer ▸ Local Disk (C:) ▸ Cars ▸
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾
Favorites
Name
Manufacturers

  - Note: It is important to put the individual file paths in the proper order; Manufacturers can't be created under Cars if Cars has not been created first.

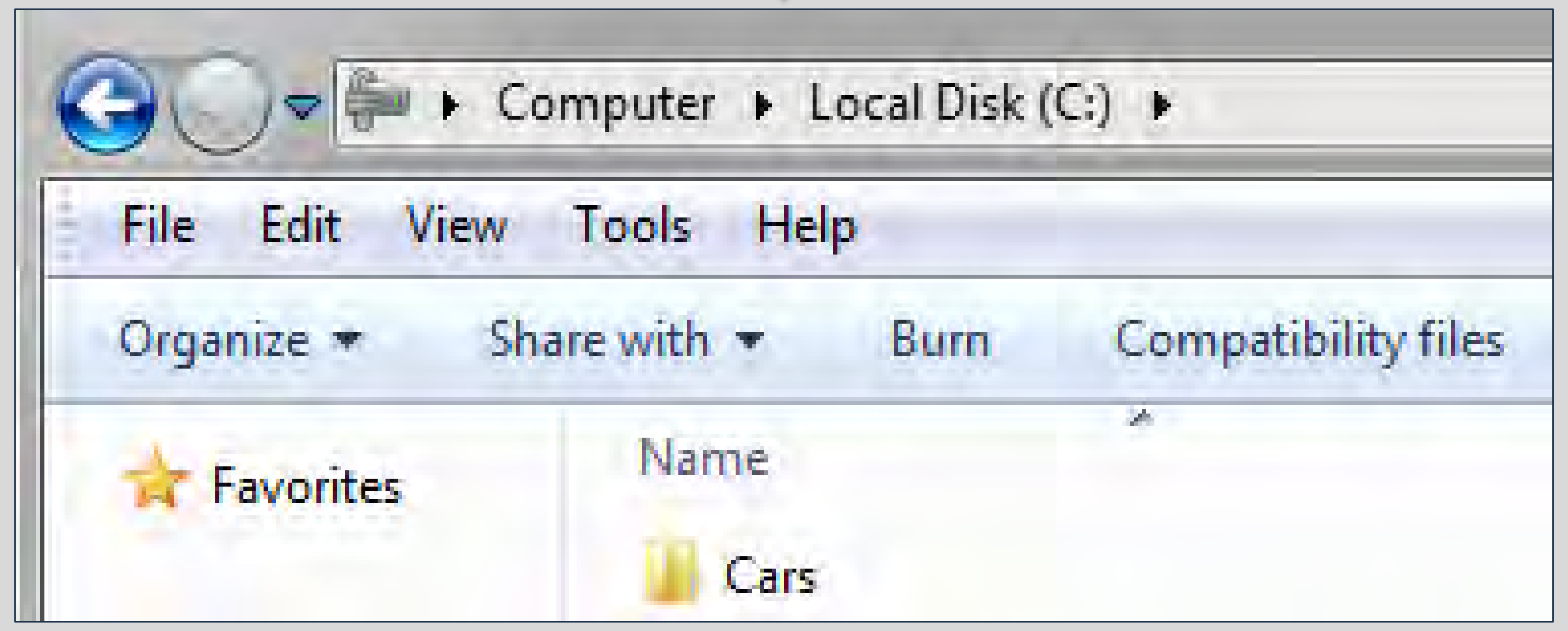
- **DCREATE (a function)**
  - Syntax: ***new-directory*=DCREATE(‘*new-directory-name*’, ’*parent-directory*’);**

    Name of new directory to be created

    Path where new directory will be created

  - Since DCREATE is a function, it can be used in a data step:

```
Data _null_;
 NewDirectory=dcreate('Cars','C:\');
Run;
```


Computer ▸ Local Disk (C:) ▸
File Edit View Tools Help
Organize ▾ Share with ▾ Burn Compatibility files
Favorites
Name
Cars

Copy That! - Using SAS® to Create Directories and Duplicate Files

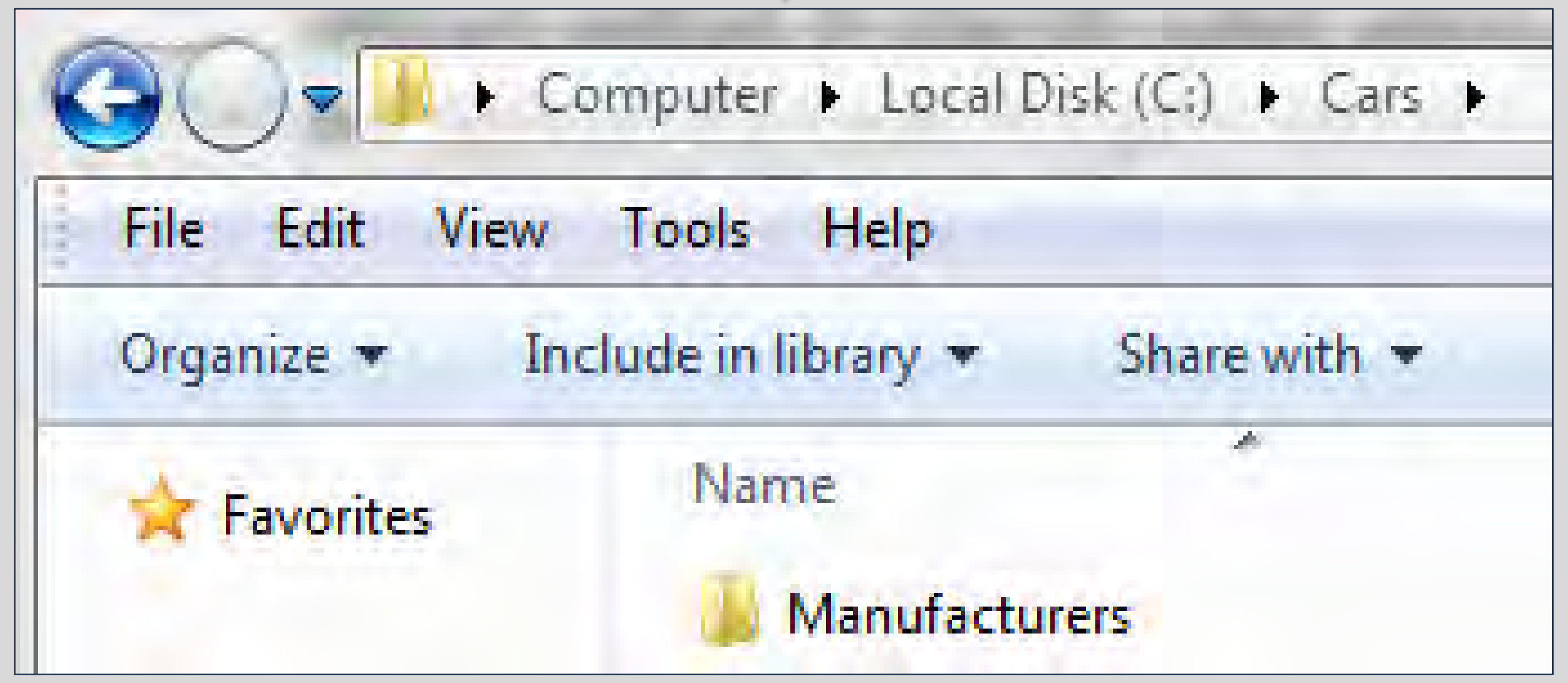
Nicole Ciaccia
Educational Testing Service

Creating a Single Folder or Directory

- **X md (an X Statement)**
  - X statements give the operating system a command directly from SAS.
  - The SAS session is exited and the processor is temporarily accessed.
  - Syntax: X '*Command*' where command is the desired command to give the operating system. X 'md' will **M**ake **D**irectories.

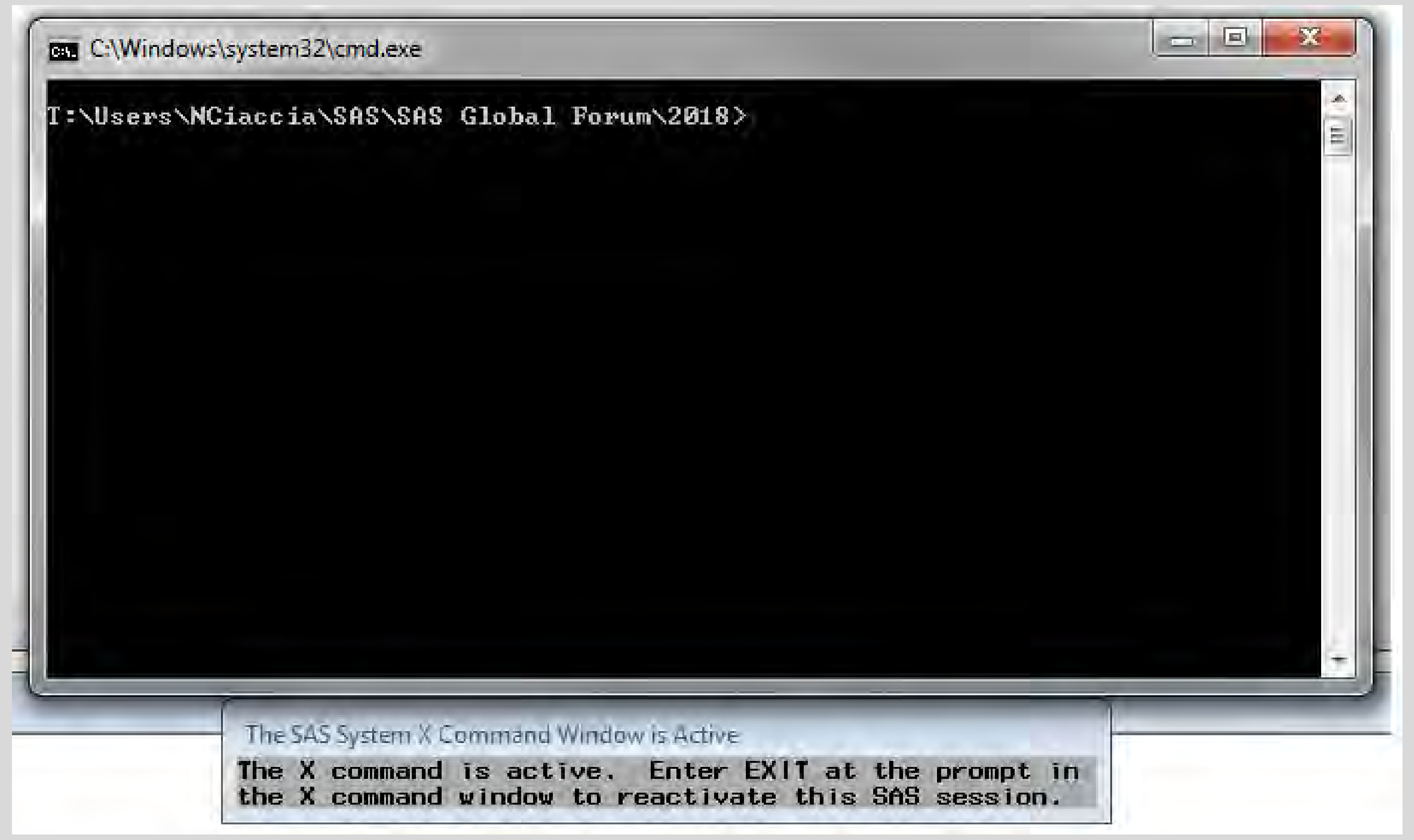
    X 'md *new-directory-file-path*';
  - Instructs Windows to leave SAS temporarily to create a folder, 'Cars' and a subfolder, 'Manufacturers' on the C: drive:

```
Data _null_;
     X 'md C:\Cars\Manufacturers';
Run;
```



**Different from using DLCREATEDIR in that both the Cars and Manufacturers folders can be created at the same time with one X statement.

- When issuing an X statement, a DOS Window automatically opens and submits the command. In order to close the DOS Window, you must type 'exit.'



```
T:\Users\NCiaccia\SAS\SAS Global Forum\2018>
```

The SAS System X Command Window is Active

```
The X command is active.  Enter EXIT at the prompt in
the X command window to reactivate this SAS session.
```

- Adding the *noxwait* option before using an X statement allows the DOS window to close automatically and return to the SAS session automatically without the user having to do anything else:

```
options noxwait;
Data _null_;
     X 'md C:\Cars\Manufacturers';
Run;
```

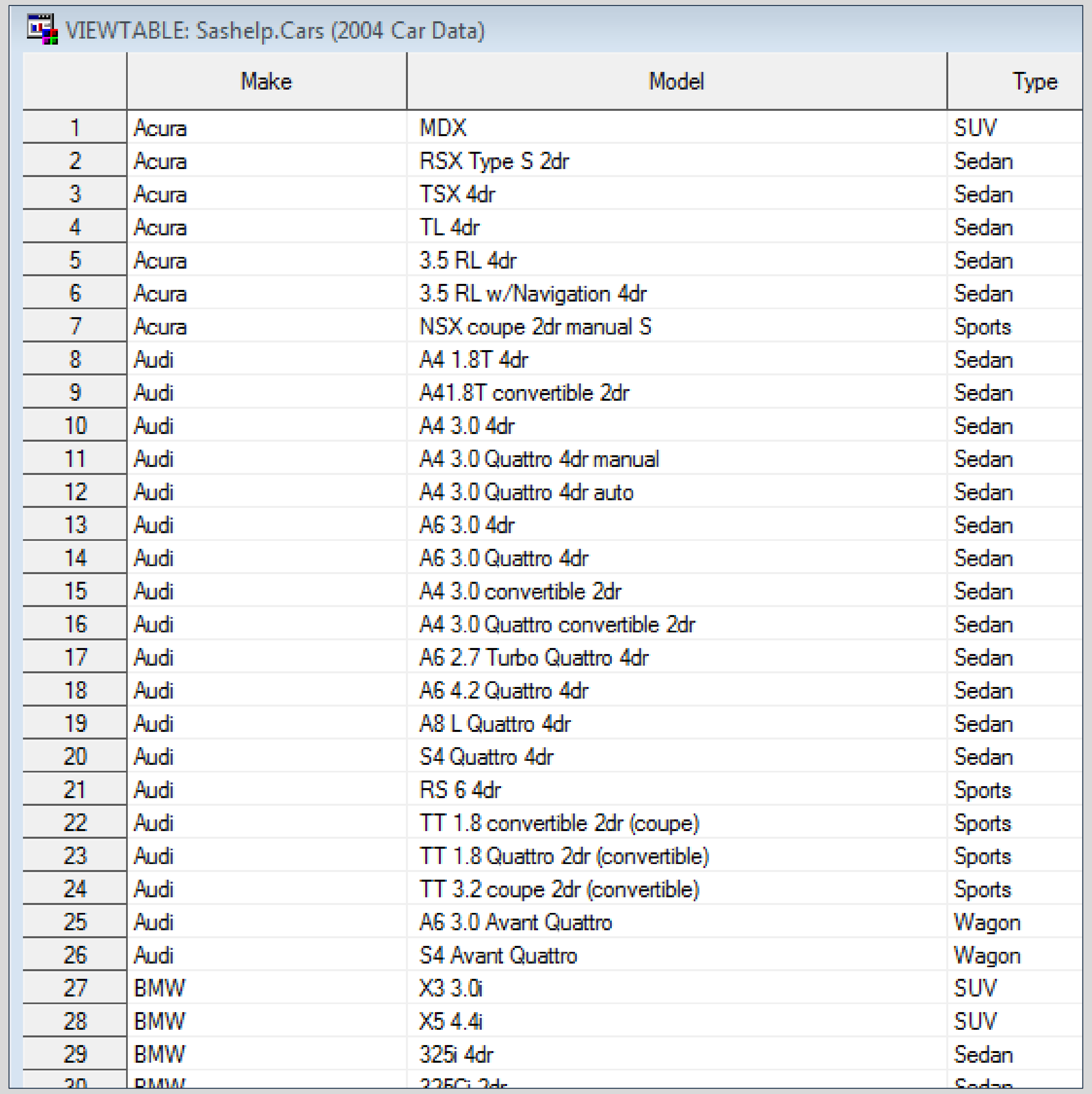
Nicole Ciaccia
Educational Testing Service

Creating Multiple Folders or Directories

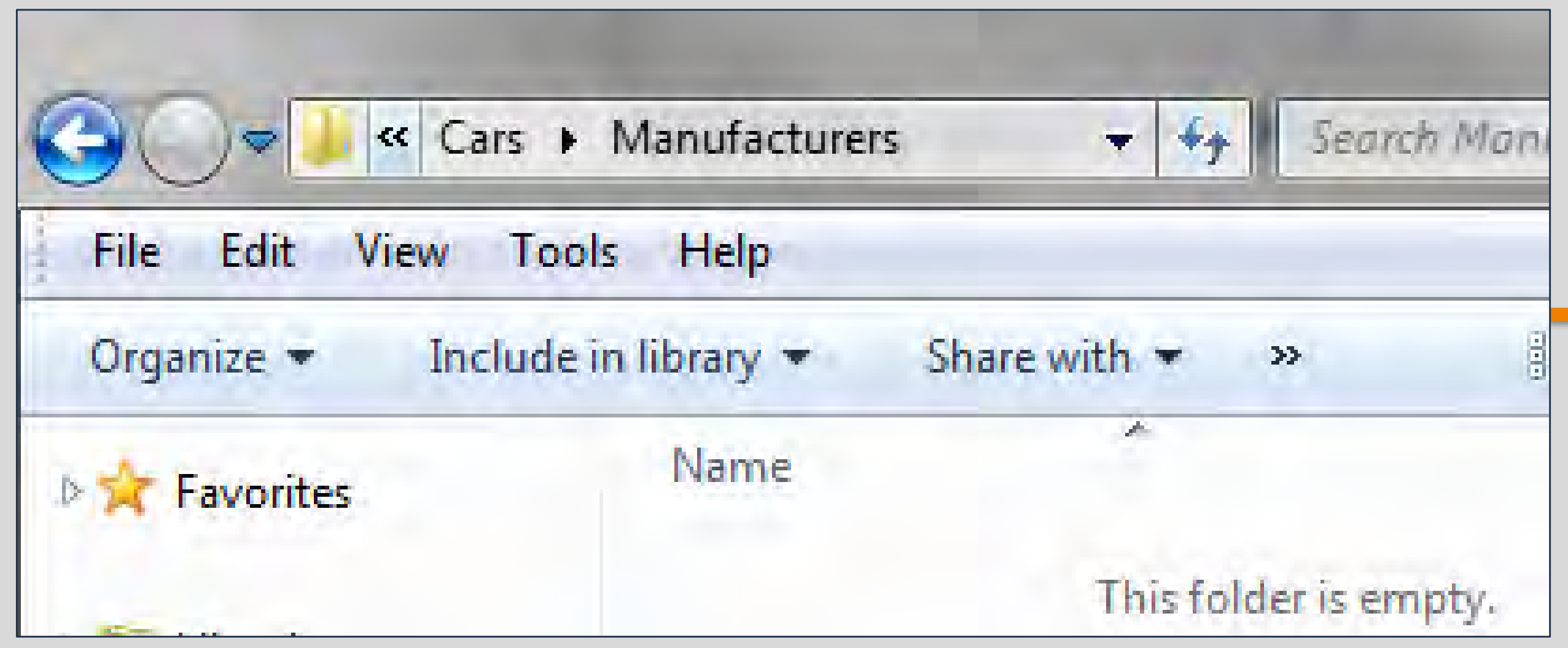
- **Business Application: Create a folder for each of the car manufacturers listed in Sashelp.Cars**
  - Can be accomplished using DLCREATEDIR with a LIBNAME statement and the CALL EXECUTE routine.

```
options dlcreatedir;
Data _null_;
Set Sashelp.Cars;
    Call Execute("libname Folder 'C:\Cars\Manufacturers\" || Make || "';");
Run;
```

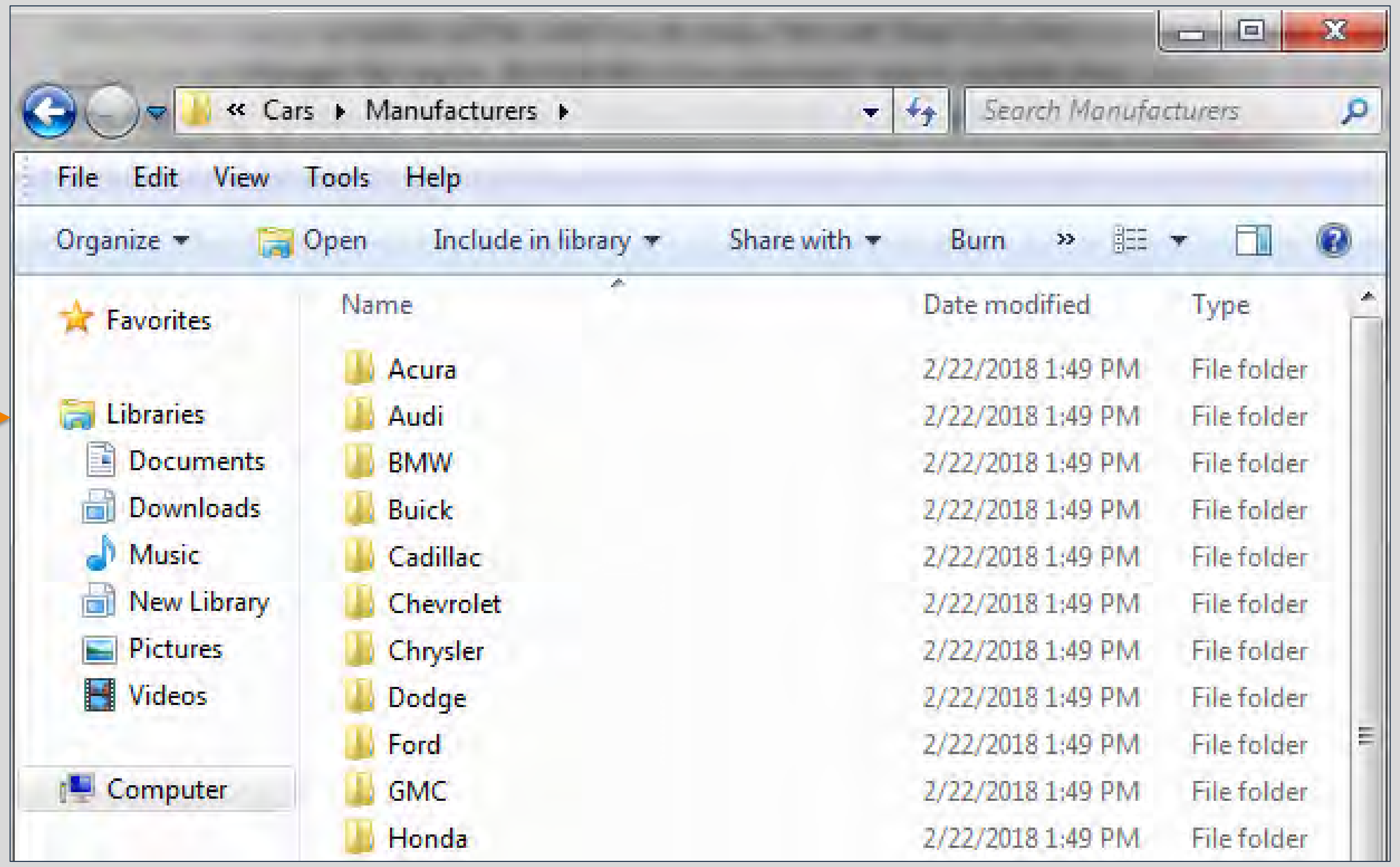
- SAS cycles through the data set and executes a LIBNAME statement for each observation.
- The corresponding value of Make for each observation is inserted into the LIBNAME statement as the name of the folder to be created.


VIEWTABLE: Sashelp.Cars (2004 Car Data)

	Make	Model	Type
1	Acura	MDX	SUV
2	Acura	RSX Type S 2dr	Sedan
3	Acura	TSX 4dr	Sedan
4	Acura	TL 4dr	Sedan
5	Acura	3.5 RL 4dr	Sedan
6	Acura	3.5 RL w/Navigation 4dr	Sedan
7	Acura	NSX coupe 2dr manual S	Sports
8	Audi	A4 1.8T 4dr	Sedan
9	Audi	A41.8T convertible 2dr	Sedan
10	Audi	A4 3.0 4dr	Sedan
11	Audi	A4 3.0 Quattro 4dr manual	Sedan
12	Audi	A4 3.0 Quattro 4dr auto	Sedan
13	Audi	A6 3.0 4dr	Sedan
14	Audi	A6 3.0 Quattro 4dr	Sedan
15	Audi	A4 3.0 convertible 2dr	Sedan
16	Audi	A4 3.0 Quattro convertible 2dr	Sedan
17	Audi	A6 2.7 Turbo Quattro 4dr	Sedan
18	Audi	A6 4.2 Quattro 4dr	Sedan
19	Audi	A8 L Quattro 4dr	Sedan
20	Audi	S4 Quattro 4dr	Sedan
21	Audi	RS 6 4dr	Sports
22	Audi	TT 1.8 convertible 2dr (coupe)	Sports
23	Audi	TT 1.8 Quattro 2dr (convertible)	Sports
24	Audi	TT 3.2 coupe 2dr (convertible)	Sports
25	Audi	A6 3.0 Avant Quattro	Wagon
26	Audi	S4 Avant Quattro	Wagon
27	BMW	X3 3.0i	SUV
28	BMW	X5 4.4i	SUV
29	BMW	325i 4dr	Sedan
30	BMW	325Ci 2dr	Sedan



This folder is empty.



Name	Date modified	Type
Acura	2/22/2018 1:49 PM	File folder
Audi	2/22/2018 1:49 PM	File folder
BMW	2/22/2018 1:49 PM	File folder
Buick	2/22/2018 1:49 PM	File folder
Cadillac	2/22/2018 1:49 PM	File folder
Chevrolet	2/22/2018 1:49 PM	File folder
Chrysler	2/22/2018 1:49 PM	File folder
Dodge	2/22/2018 1:49 PM	File folder
Ford	2/22/2018 1:49 PM	File folder
GMC	2/22/2018 1:49 PM	File folder
Honda	2/22/2018 1:49 PM	File folder

While only the DLCREATDIR option was shown here, a similar method could be applied substituting in DCREATE or X md.

Copy That! - Using SAS® to Create Directories and Duplicate Files

Nicole Ciaccia
Educational Testing Service

Duplicating Files

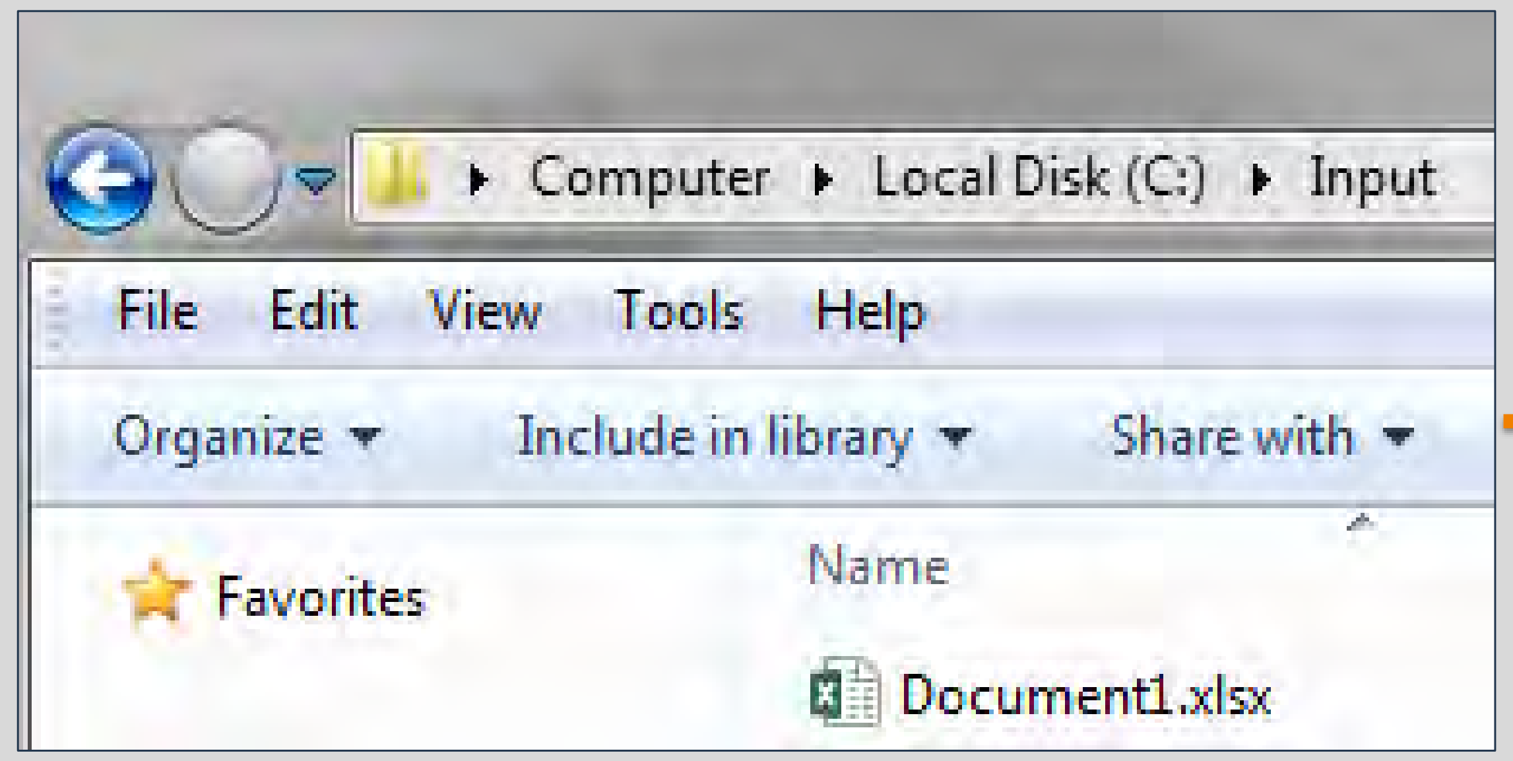
- **X Copy**
  - Instead of putting 'md' in the command argument to make directories, the Copy command can be used in order to duplicate files.
  - Syntax: **X 'Copy *original-file output-file*'**

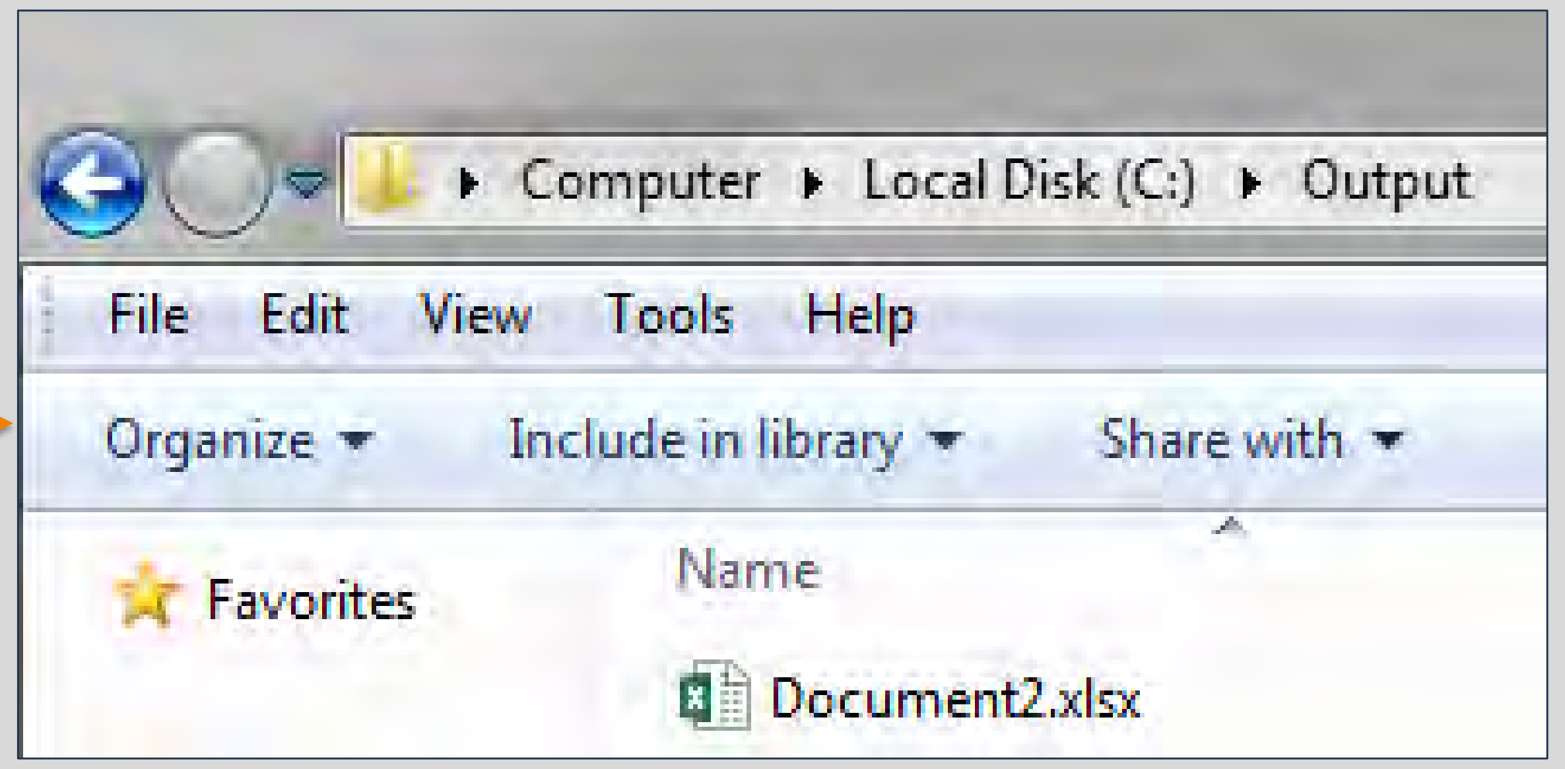
File path of document to be duplicated

File path where duplicated file should be placed

```
options noxwait;
Data _null_;
   X 'Copy C:\Input\Document1.xlsx C:\Output\Document2.xlsx';
Run;
```

  - Document1 located in C:\Input is specified as the file to duplicate, and it should be copied to C:\Output, this time named Document2.





Note: X commands are global statements so they cannot be executed conditionally.

- **System Copy**
  - The SYSTEM function can also be utilized to instruct the operating system to perform a desired command.
  - Syntax: **System ('Copy *original-file output-file*');**
  - Generates a return code that denotes if the command executed or not.
    - For Windows 7, by default, it will return 0 if the command executes properly and 1 if it does not.

```
options noxwait;
Data _null_;
   Check= system('Copy C:\Input\Document1.xlsx C:\Output\Document2.xlsx');
   put Check=;
Run;
```

  - Document1 will again be duplicated, but this time a variable, *check*, is created that stores the return code and states whether or not the command executed.

```
options noxwait;
Data _null_;
  Check= system('Copy C:\Input\Document1.xlsx C:\Output\Document2.xlsx');
  put Check=;
Run;

Check=0
```

Note: Unlike X commands, the SYSTEM function can be used conditionally if desired.

Copy That! - Using SAS® to Create Directories and Duplicate Files

Nicole Ciaccia
Educational Testing Service

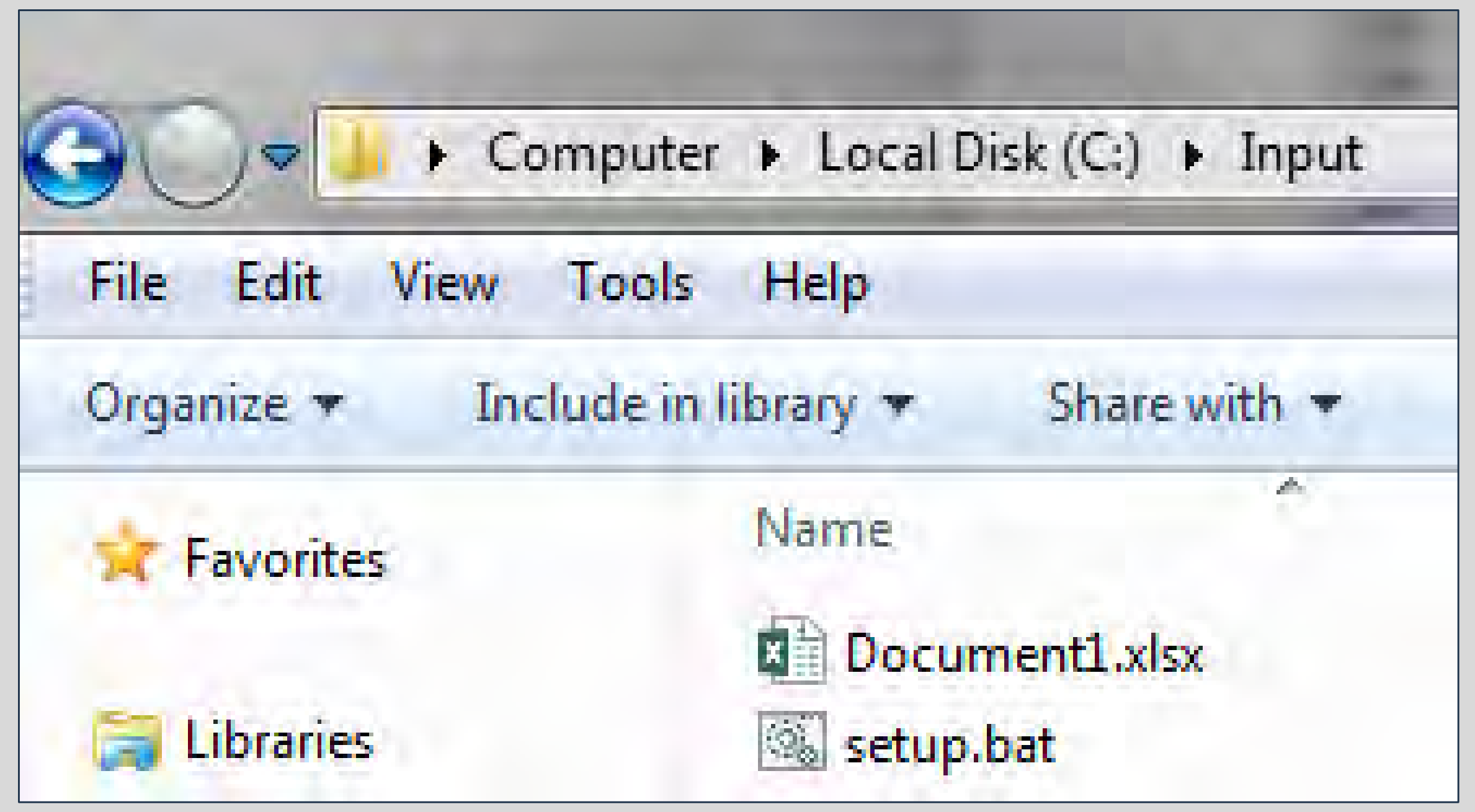
Duplicating Files

- **Create a Batch File**
  - A batch file is a script that contains a series of commands to be performed by the processor
  - For Windows, the extension for this plain text file is .bat.
  - When batch file is double clicked, commands contained within are carried out.
  - File can be created by typing the appropriate commands in a text editor, such as Notepad or Textpad, and saving it with the .bat extension, or SAS can be used to create this file automatically.

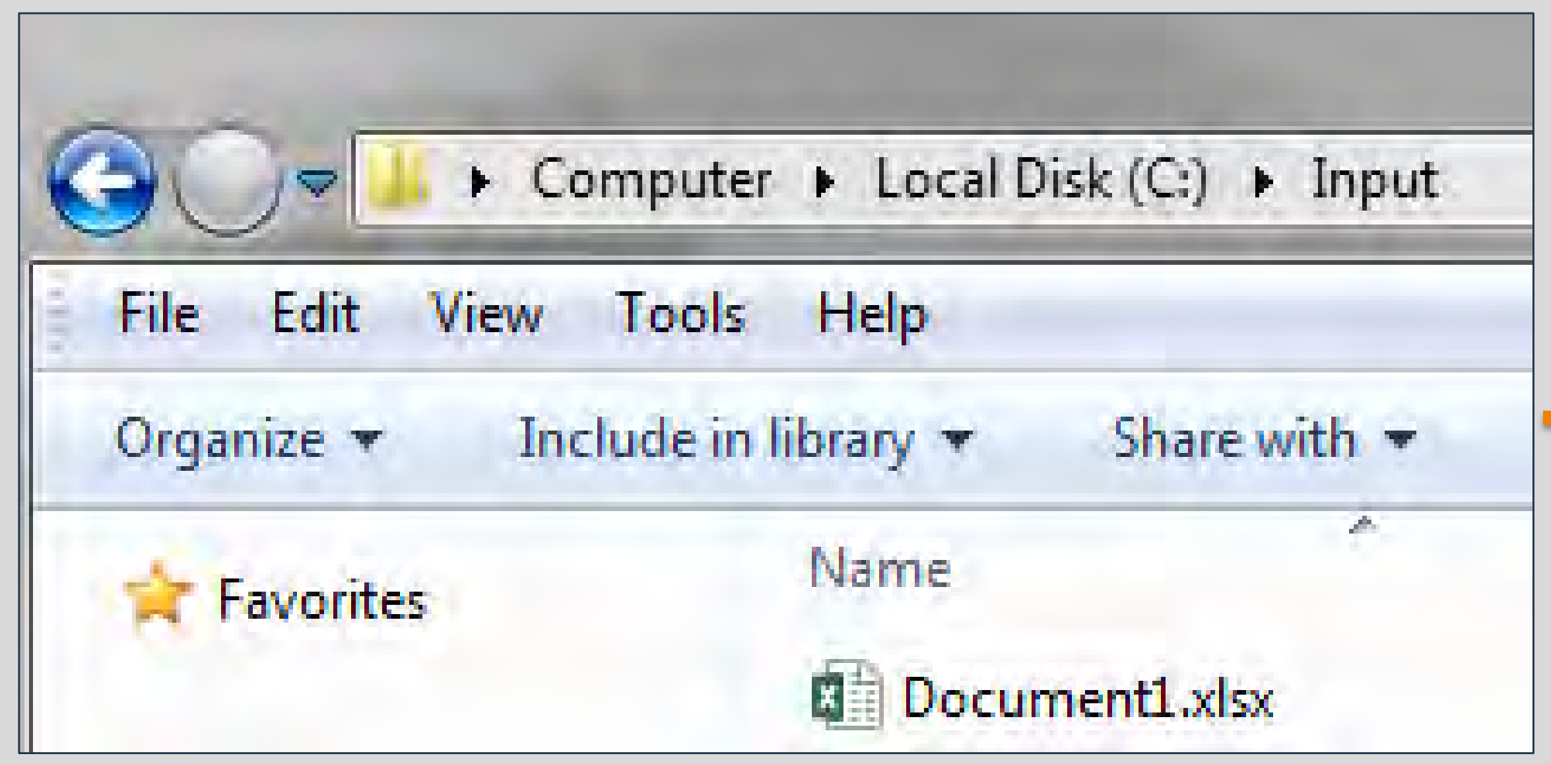
```
Data _null_;
    final='Copy C:\Input\Document1.xlsx C:\Output\Document2.xlsx';
    file "C:\Input\setup.bat";
        put @1 final;
Run;
```

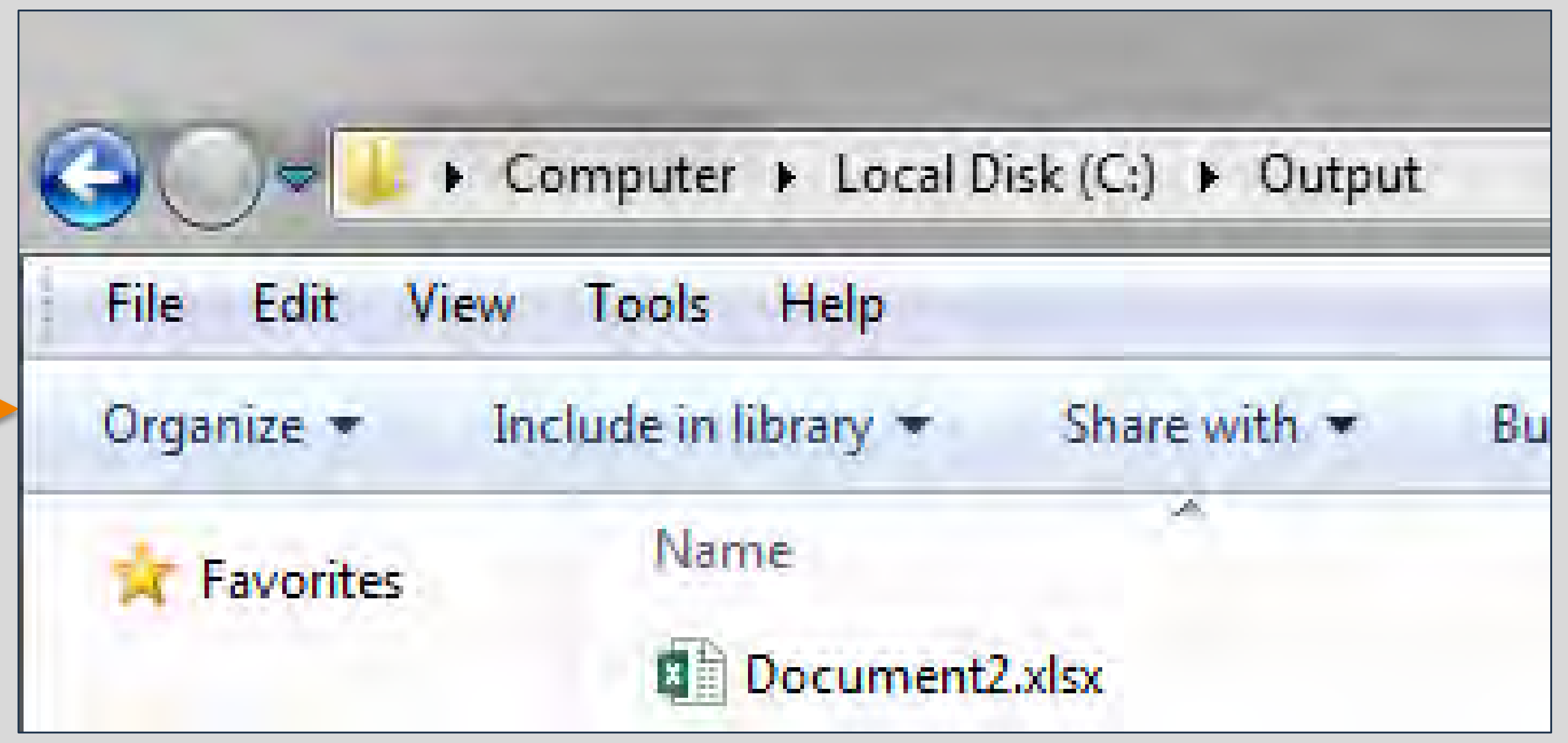
- A variable, *final*, is defined with the desired command. The syntax is the same as System Copy: 'Copy *original-file output-file*'.
- The output location for the batch file is defined with a file statement, and the put statement specifies that the 'final' variable should be written out to this file.

- The folder C:\Input now contains a batch file named setup.


Computer ▸ Local Disk (C:) ▸ Input
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾
Favorites
Libraries
Name
Document1.xlsx
setup.bat

- Once the file is double clicked on, Document1.xlsx will be copied into C:\Output as Document2.xlsx.


Computer ▸ Local Disk (C:) ▸ Input
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾
Favorites
Name
Document1.xlsx


Computer ▸ Local Disk (C:) ▸ Output
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾ Bu
Favorites
Name
Document2.xlsx

- Note: the batch file remains in C:\Input and should be deleted after it has been run. (If desired, you *can* code to have the .bat automatically delete after being run.)

Copy That! - Using SAS® to Create Directories and Duplicate Files

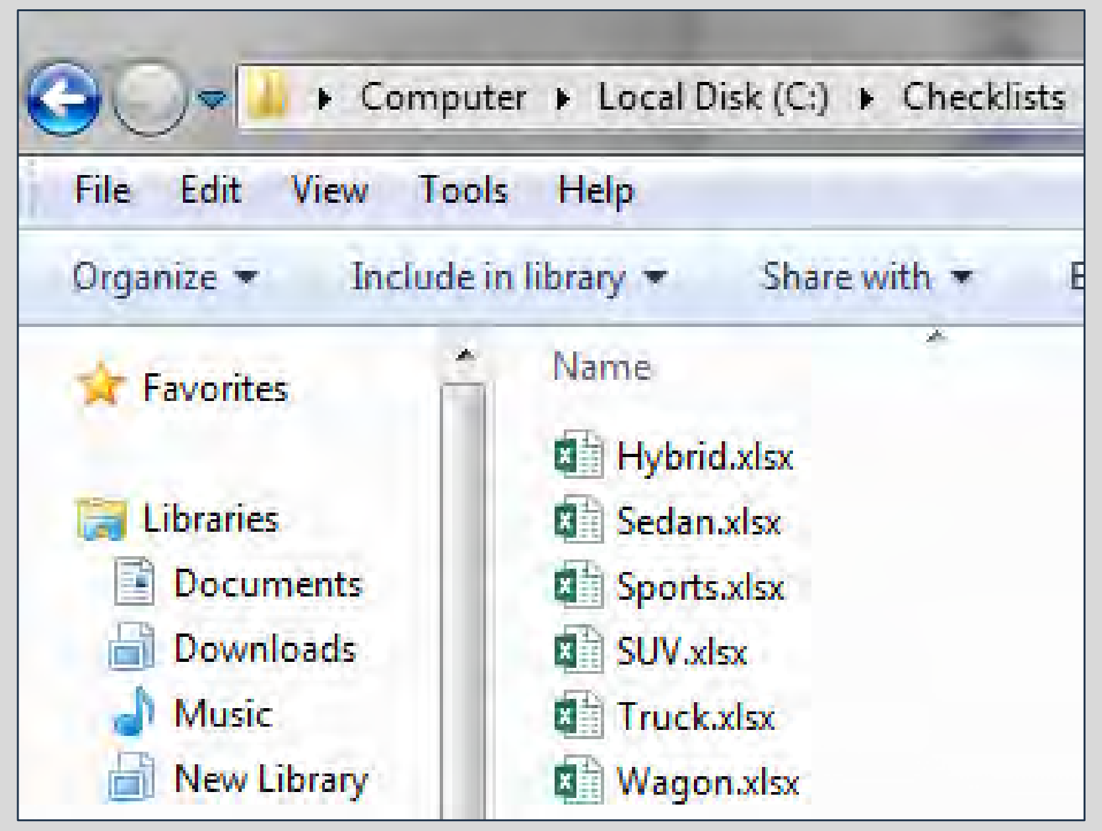
Nicole Ciaccia
Educational Testing Service

Duplicating Multiple Files

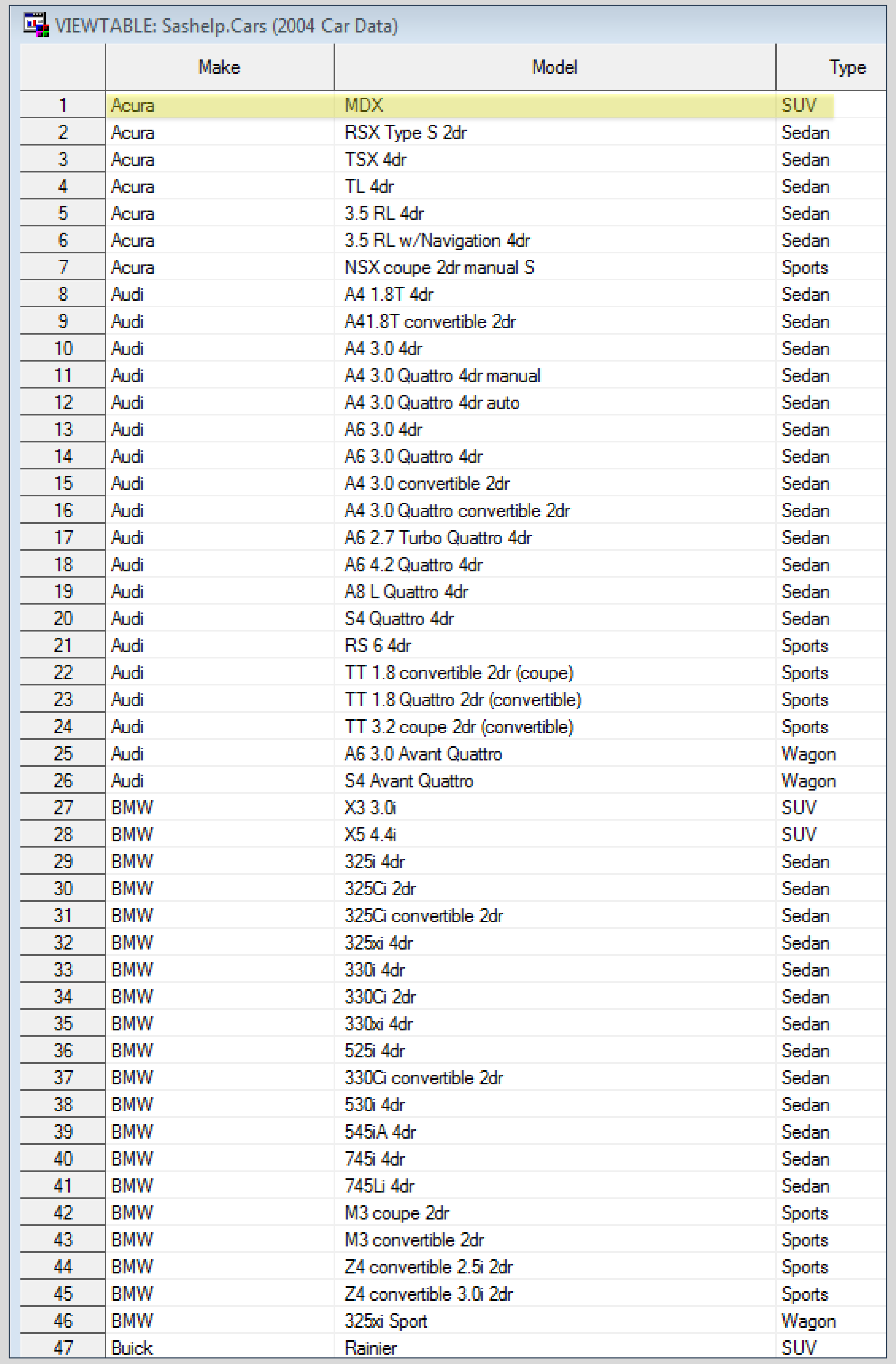
- **Business Application: Create a checklist for each car model in the appropriate manufacturer's folder, based on car type.**
  - 6 types of cars listed under the Type variable: SUV, Sedan, Sports, Wagon, Hybrid and Truck.
  - Suppose a company has separate checklists for each vehicle type.
- **Example:** Output an SUV checklist in the Acura folder with the name MDX.

	Make	Model	Type
1	Acura	MDX	SUV

- Assume that a copy of each of these checklists exists as an .xlsx document in C:\Checklists.



Hybrid.xlsx
Sedan.xlsx
Sports.xlsx
SUV.xlsx
Truck.xlsx
Wagon.xlsx


VIEWTABLE: Sashelp.Cars (2004 Car Data)

	Make	Model	Type
1	Acura	MDX	SUV
2	Acura	RSX Type S 2dr	Sedan
3	Acura	TSX 4dr	Sedan
4	Acura	TL 4dr	Sedan
5	Acura	3.5 RL 4dr	Sedan
6	Acura	3.5 RL w/Navigation 4dr	Sedan
7	Acura	NSX coupe 2dr manual S	Sports
8	Audi	A4 1.8T 4dr	Sedan
9	Audi	A41.8T convertible 2dr	Sedan
10	Audi	A4 3.0 4dr	Sedan
11	Audi	A4 3.0 Quattro 4dr manual	Sedan
12	Audi	A4 3.0 Quattro 4dr auto	Sedan
13	Audi	A6 3.0 4dr	Sedan
14	Audi	A6 3.0 Quattro 4dr	Sedan
15	Audi	A4 3.0 convertible 2dr	Sedan
16	Audi	A4 3.0 Quattro convertible 2dr	Sedan
17	Audi	A6 2.7 Turbo Quattro 4dr	Sedan
18	Audi	A6 4.2 Quattro 4dr	Sedan
19	Audi	A8 L Quattro 4dr	Sedan
20	Audi	S4 Quattro 4dr	Sedan
21	Audi	RS 6 4dr	Sports
22	Audi	TT 1.8 convertible 2dr (coupe)	Sports
23	Audi	TT 1.8 Quattro 2dr (convertible)	Sports
24	Audi	TT 3.2 coupe 2dr (convertible)	Sports
25	Audi	A6 3.0 Avant Quattro	Wagon
26	Audi	S4 Avant Quattro	Wagon
27	BMW	X3 3.0i	SUV
28	BMW	X5 4.4i	SUV
29	BMW	325i 4dr	Sedan
30	BMW	325Ci 2dr	Sedan
31	BMW	325Ci convertible 2dr	Sedan
32	BMW	325xi 4dr	Sedan
33	BMW	330i 4dr	Sedan
34	BMW	330Ci 2dr	Sedan
35	BMW	330xi 4dr	Sedan
36	BMW	525i 4dr	Sedan
37	BMW	330Ci convertible 2dr	Sedan
38	BMW	530i 4dr	Sedan
39	BMW	545iA 4dr	Sedan
40	BMW	745i 4dr	Sedan
41	BMW	745Li 4dr	Sedan
42	BMW	M3 coupe 2dr	Sports
43	BMW	M3 convertible 2dr	Sports
44	BMW	Z4 convertible 2.5i 2dr	Sports
45	BMW	Z4 convertible 3.0i 2dr	Sports
46	BMW	325xi Sport	Wagon
47	Buick	Rainier	SUV

How to efficiently create 428 checklists and output them all in the correct place?

Two Methods for Completing the Task:

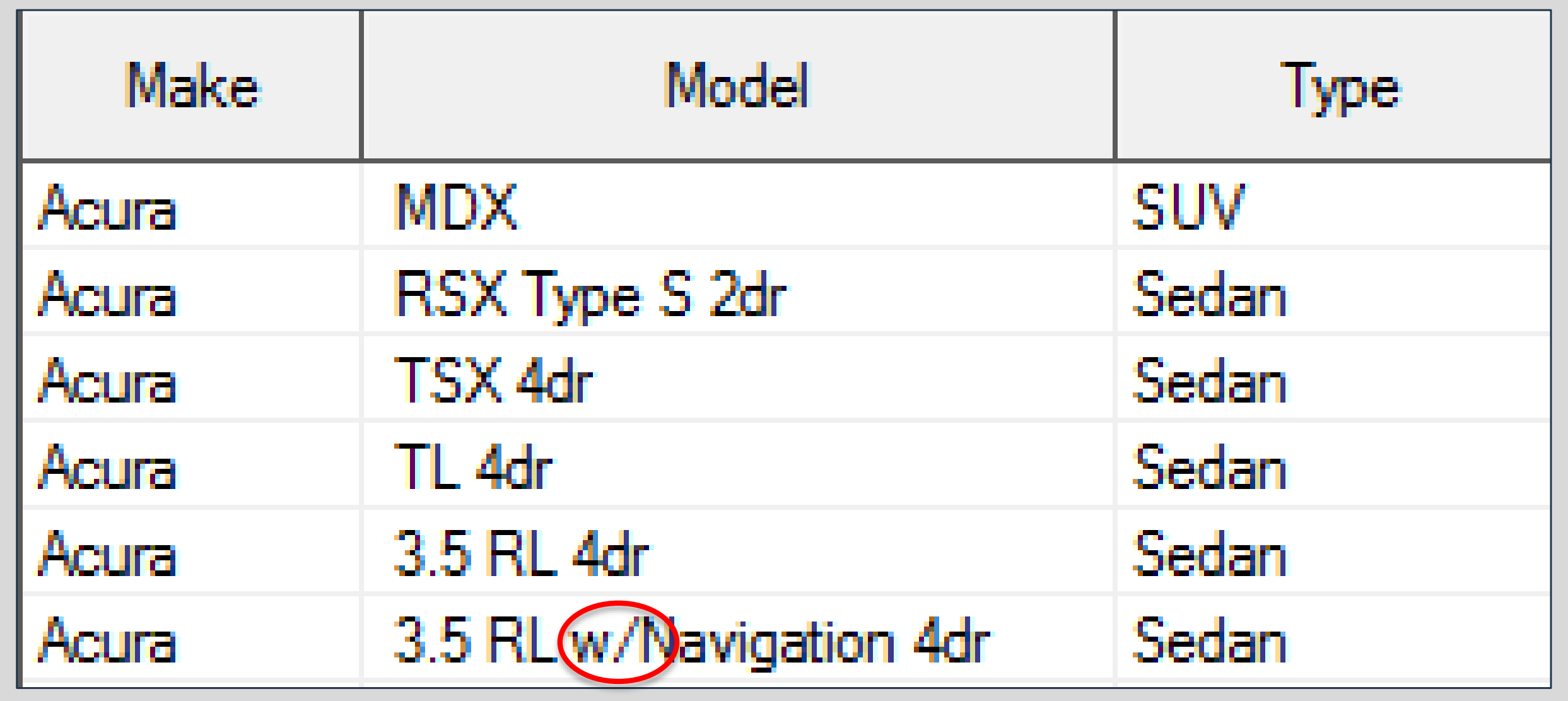
1. Creating a Batch file
   - ✚ Coding is simple, runs faster.
   - ✖ SAS Log only tells you if batch file was created successfully, not if the files were successfully duplicated.
2. Using System Copy
   - ✚ SAS Log tells you directly if files were created.
   - ✖ Coding is more complex, takes longer to run.

Copy That! - Using SAS® to Create Directories and Duplicate Files

Nicole Ciaccia
Educational Testing Service

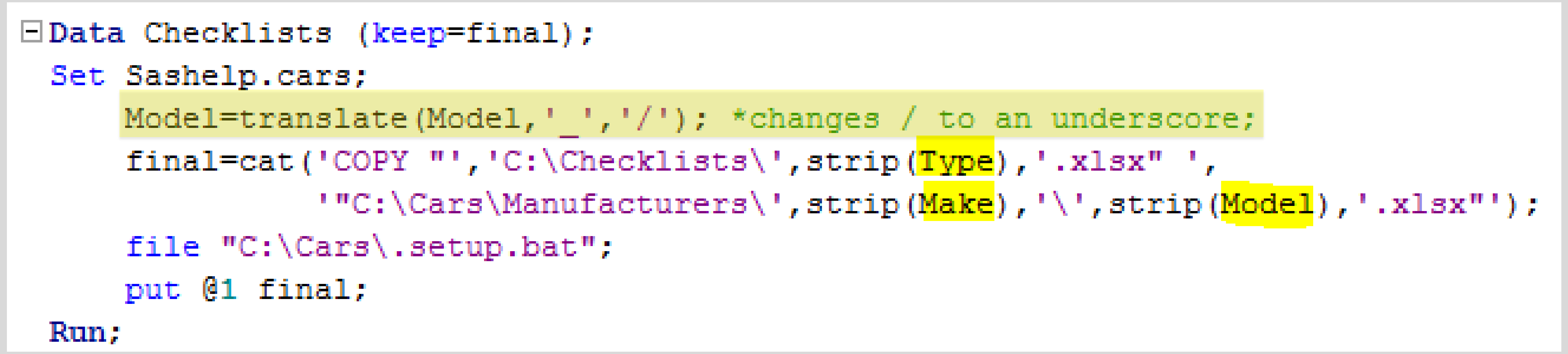
Duplicating Multiple Files

- **Batch File Method**
  - Syntax for the command contained in the batch file is: **'Copy original-file output-file'**
  - SAS needs to cycle through the Cars data set and output a line for each time a file needs to be copied.
  - Each line must contain the corresponding:
    1. File to be copied *(from Type)*
    2. Output file name *(from Model)*
    3. Output file location *(from Make)*



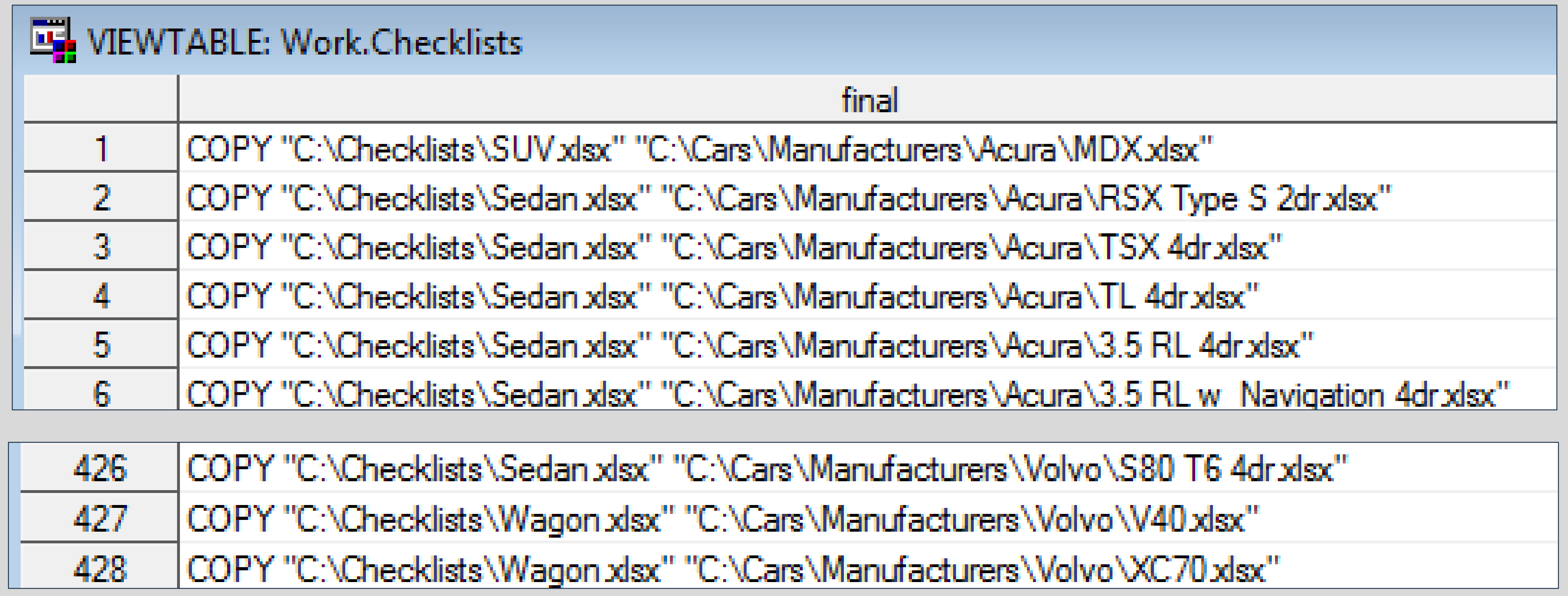
Make	Model	Type
Acura	MDX	SUV
Acura	RSX Type S 2dr	Sedan
Acura	TSX 4dr	Sedan
Acura	TL 4dr	Sedan
Acura	3.5 RL 4dr	Sedan
Acura	3.5 RL w/Navigation 4dr	Sedan

  - Defines a variable, *final*, that is equal to the desired command.
    - Final is set up as a shell, inserting the 3 variables that will change depending on each observation:



```
Data Checklists (keep=final);
 Set Sashelp.cars;
    Model=translate(Model,'_','/'); *changes / to an underscore;
    final=cat('COPY "','C:\Checklists\',strip(Type),'.xlsx" ',
              '"C:\Cars\Manufacturers\',strip(Make),'\',strip(Model),'.xlsx"');
    file "C:\Cars\.setup.bat";
    put @1 final;
Run;
```

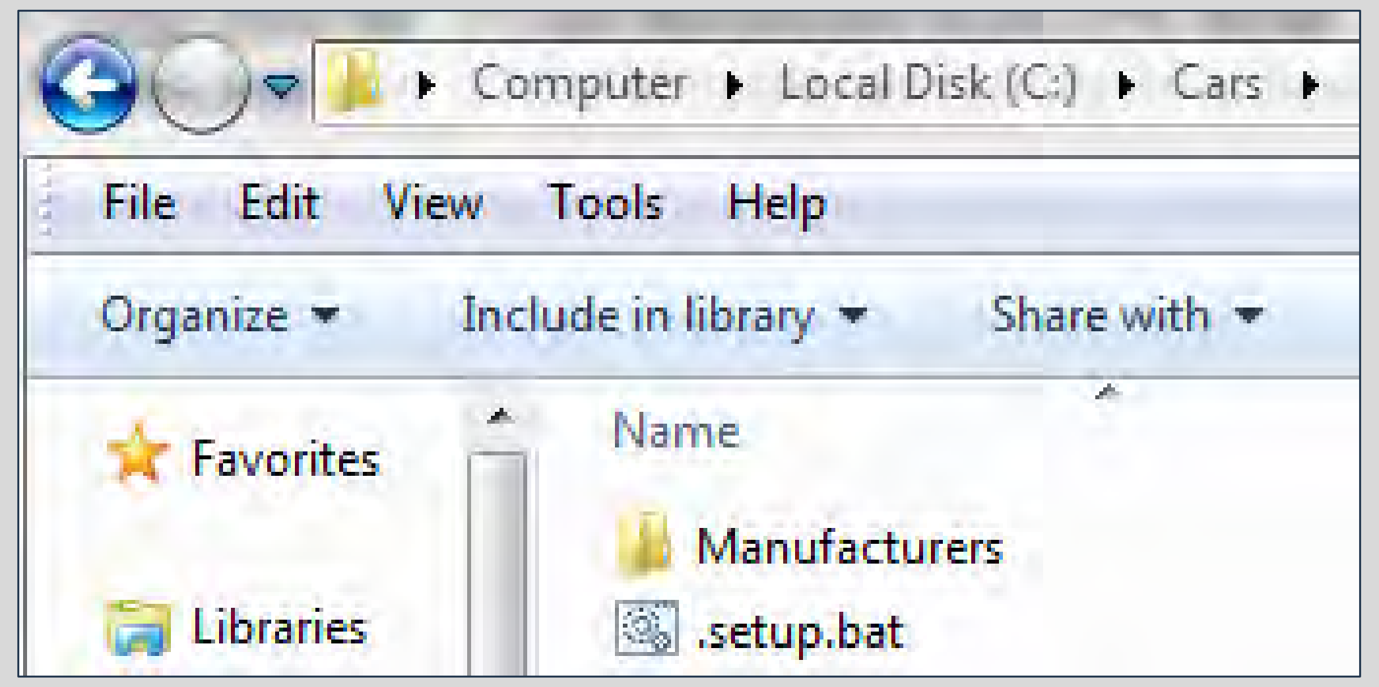
- Once this code is run, the data set Checklists looks like this:



VIEWTABLE: Work.Checklists

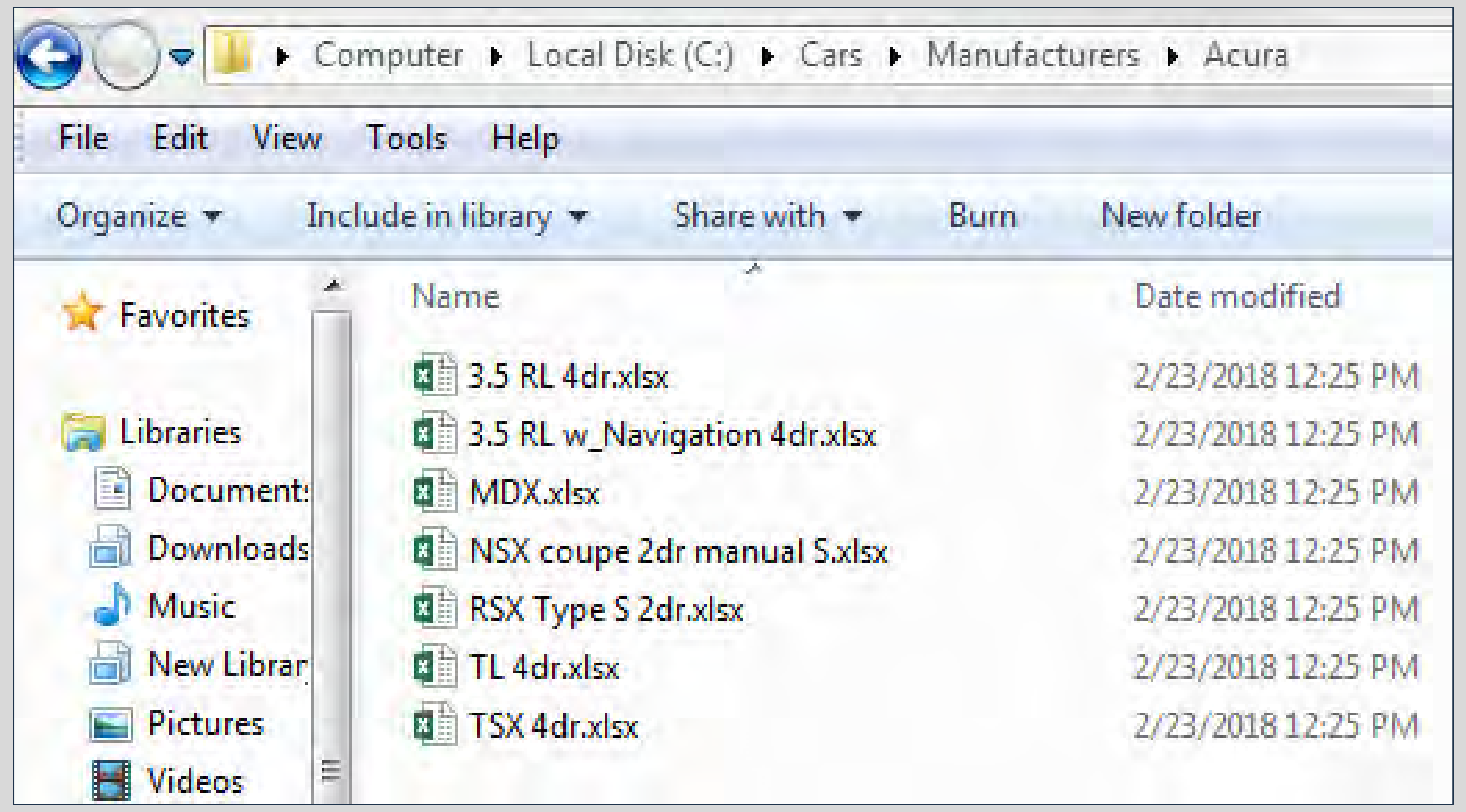
	final
1	COPY "C:\Checklists\SUV.xlsx" "C:\Cars\Manufacturers\Acura\MDX.xlsx"
2	COPY "C:\Checklists\Sedan.xlsx" "C:\Cars\Manufacturers\Acura\RSX Type S 2dr.xlsx"
3	COPY "C:\Checklists\Sedan.xlsx" "C:\Cars\Manufacturers\Acura\TSX 4dr.xlsx"
4	COPY "C:\Checklists\Sedan.xlsx" "C:\Cars\Manufacturers\Acura\TL 4dr.xlsx"
5	COPY "C:\Checklists\Sedan.xlsx" "C:\Cars\Manufacturers\Acura\3.5 RL 4dr.xlsx"
6	COPY "C:\Checklists\Sedan.xlsx" "C:\Cars\Manufacturers\Acura\3.5 RL w_Navigation 4dr.xlsx"
426	COPY "C:\Checklists\Sedan.xlsx" "C:\Cars\Manufacturers\Volvo\S80 T6 4dr.xlsx"
427	COPY "C:\Checklists\Wagon.xlsx" "C:\Cars\Manufacturers\Volvo\V40.xlsx"
428	COPY "C:\Checklists\Wagon.xlsx" "C:\Cars\Manufacturers\Volvo\XC70.xlsx"

A batch file with these commands is output to C:\Cars:


Computer ▸ Local Disk (C:) ▸ Cars ▸
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾
Favorites
Libraries
Name

Name
Manufacturers
.setup.bat

Once the batch file is double clicked:


Computer ▸ Local Disk (C:) ▸ Cars ▸ Manufacturers ▸ Acura
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾ Burn New folder
Favorites
Libraries
Documents
Downloads
Music
New Library
Pictures
Videos

Name	Date modified
3.5 RL 4dr.xlsx	2/23/2018 12:25 PM
3.5 RL w_Navigation 4dr.xlsx	2/23/2018 12:25 PM
MDX.xlsx	2/23/2018 12:25 PM
NSX coupe 2dr manual S.xlsx	2/23/2018 12:25 PM
RSX Type S 2dr.xlsx	2/23/2018 12:25 PM
TL 4dr.xlsx	2/23/2018 12:25 PM
TSX 4dr.xlsx	2/23/2018 12:25 PM

Copy That! - Using SAS® to Create Directories and Duplicate Files

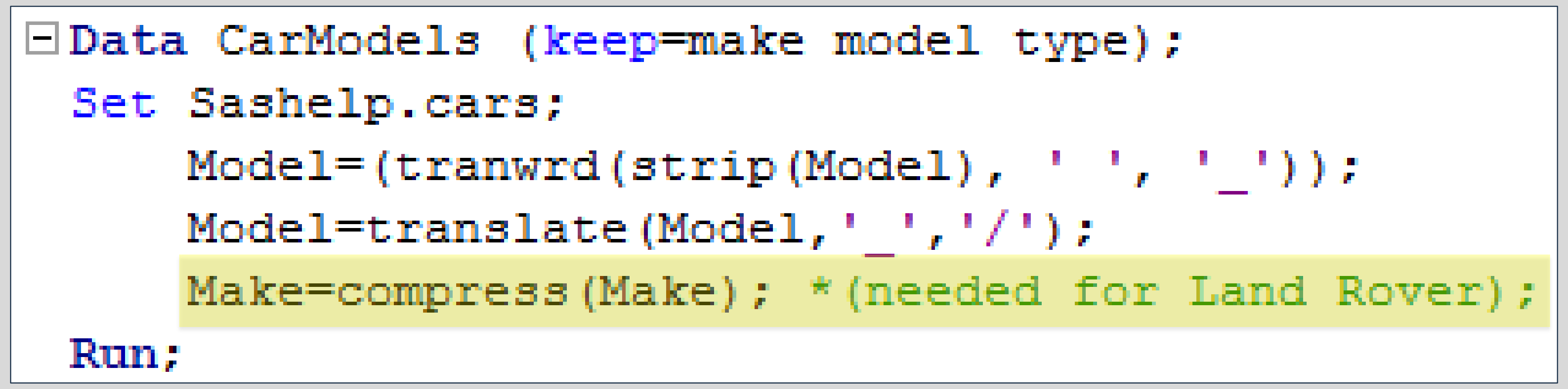
Nicole Ciaccia
Educational Testing Service

Duplicating Multiple Files

- **System Copy Method**

  Step 1: Because the values of Type, Model and Make will be used as variables within SYSTEM, some data manipulation needs to occur.

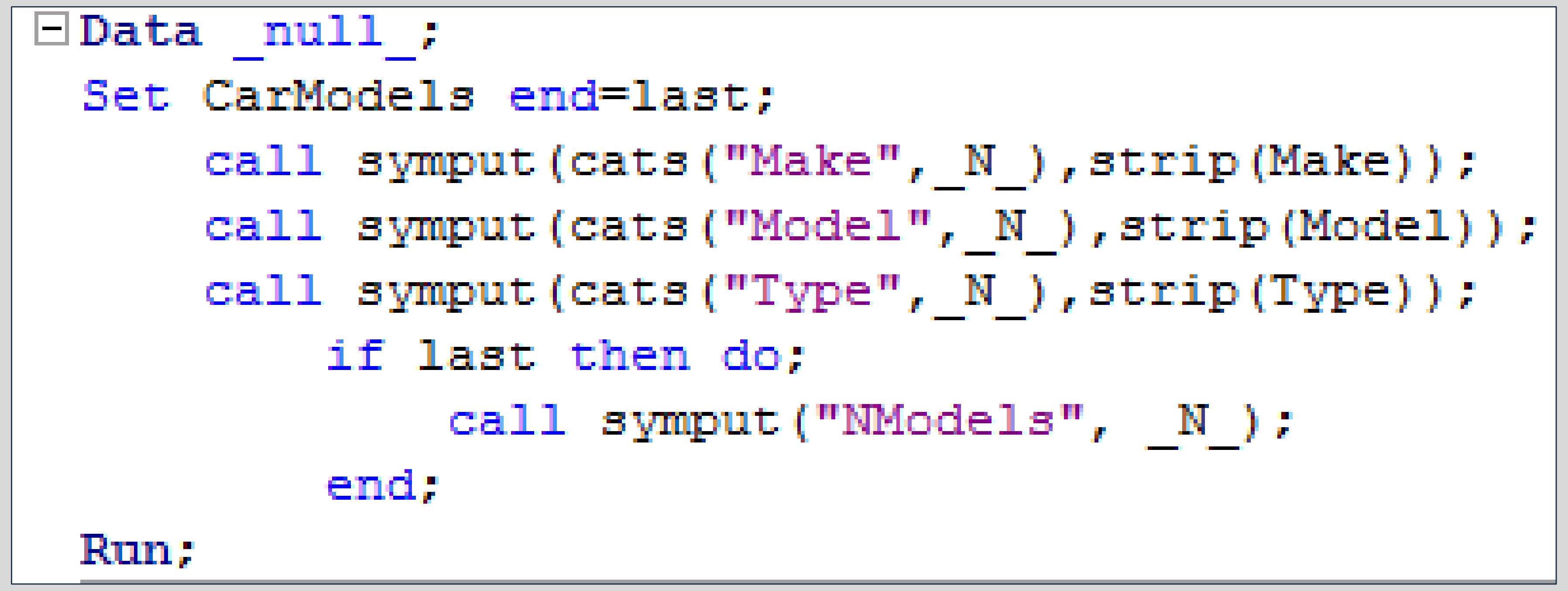
  - All spaces need to be changed to underscores or compressed.
  - Forward slashes will need to be replaced with underscores.



```
Data CarModels (keep=make model type);
 Set Sashelp.cars;
     Model=(tranwrd(strip(Model), ' ', '_'));
     Model=translate(Model,'_','/');
     Make=compress(Make); *(needed for Land Rover);
 Run;
```

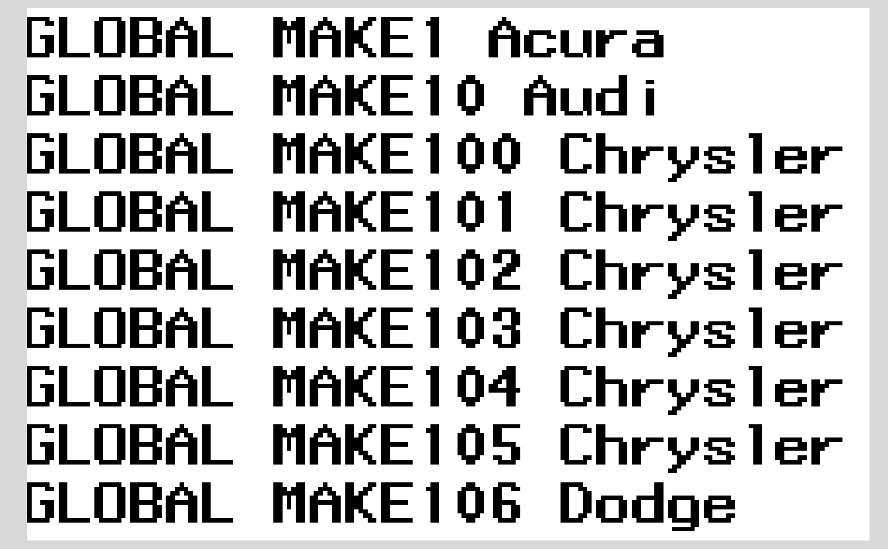
**When the folders were created, compress was not used. The name of this folder can be manually changed to LandRover since it is the only manufacturer with this issue. If this was an issue for multiple manufacturers, the Make variable could have been compressed in the CALL SYMPUT statement when creating the folders initially.)

Step 2: Assign each Make, Model and Type as a macro variable value using the CALL SYMPUT routine:

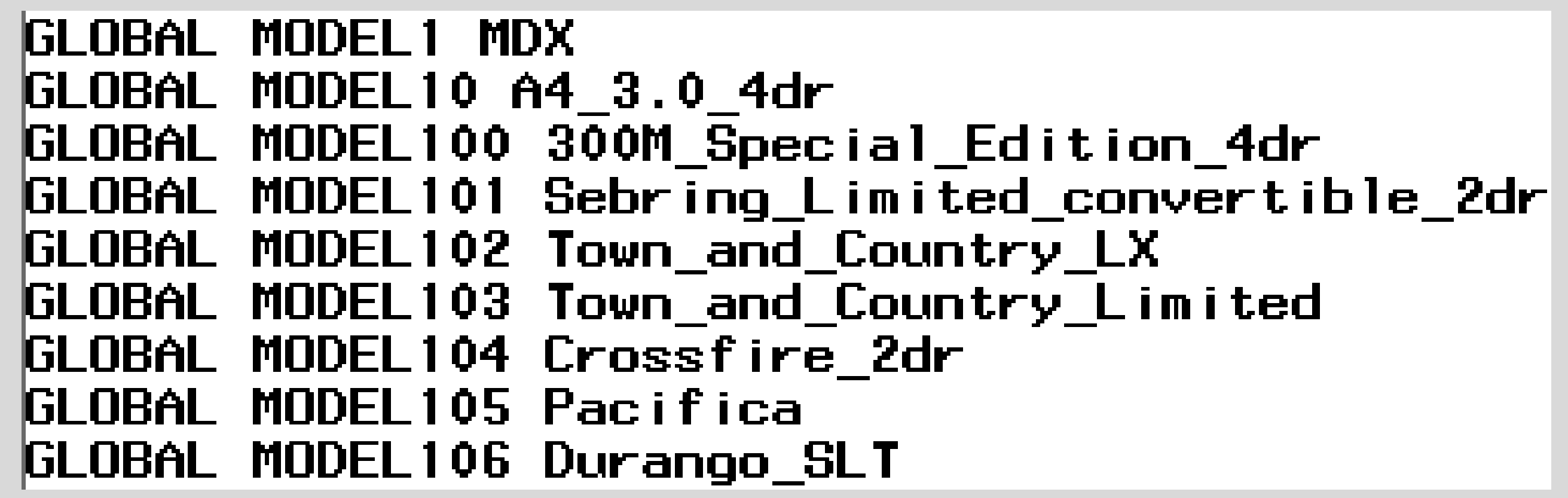


```
Data _null_;
 Set CarModels end=last;
     call symput(cats("Make",_N_),strip(Make));
     call symput(cats("Model",_N_),strip(Model));
     call symput(cats("Type",_N_),strip(Type));
        if last then do;
           call symput("NModels", _N_);
        end;
 Run;
```

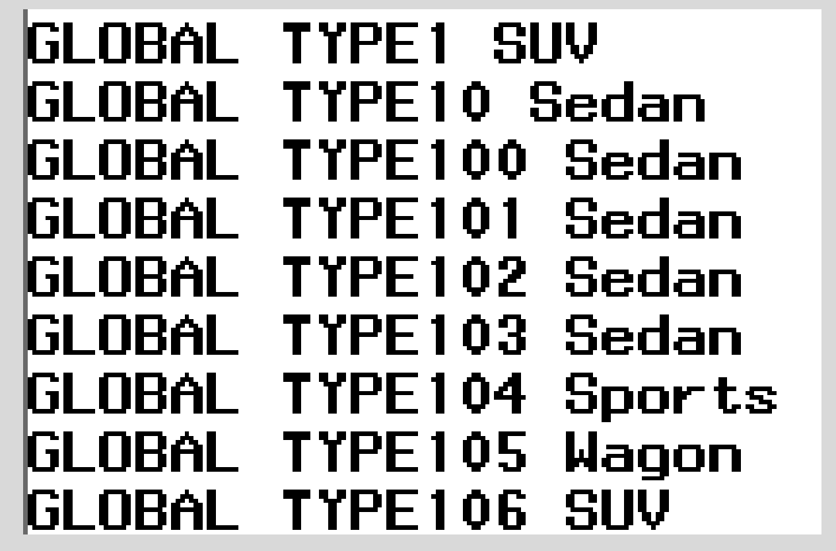
- Three variables will be changing, so three CALL SYMPUT statements are needed.
- STRIP is used to ensure there are no extra blank spaces.
- The last observation in the data set is identified and then that observation number is stored in the macro variable, NModels so that SAS knows how many times to cycle through the do loop.



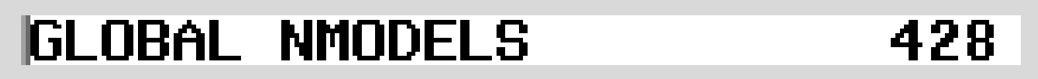
```
GLOBAL MAKE1 Acura
GLOBAL MAKE10 Audi
GLOBAL MAKE100 Chrysler
GLOBAL MAKE101 Chrysler
GLOBAL MAKE102 Chrysler
GLOBAL MAKE103 Chrysler
GLOBAL MAKE104 Chrysler
GLOBAL MAKE105 Chrysler
GLOBAL MAKE106 Dodge
```



```
GLOBAL MODEL1 MDX
GLOBAL MODEL10 A4_3.0_4dr
GLOBAL MODEL100 300M_Special_Edition_4dr
GLOBAL MODEL101 Sebring_Limited_convertible_2dr
GLOBAL MODEL102 Town_and_Country_LX
GLOBAL MODEL103 Town_and_Country_Limited
GLOBAL MODEL104 Crossfire_2dr
GLOBAL MODEL105 Pacifica
GLOBAL MODEL106 Durango_SLT
```



```
GLOBAL TYPE1 SUV
GLOBAL TYPE10 Sedan
GLOBAL TYPE100 Sedan
GLOBAL TYPE101 Sedan
GLOBAL TYPE102 Sedan
GLOBAL TYPE103 Sedan
GLOBAL TYPE104 Sports
GLOBAL TYPE105 Wagon
GLOBAL TYPE106 SUV
```



```
GLOBAL NMODELS          428
```

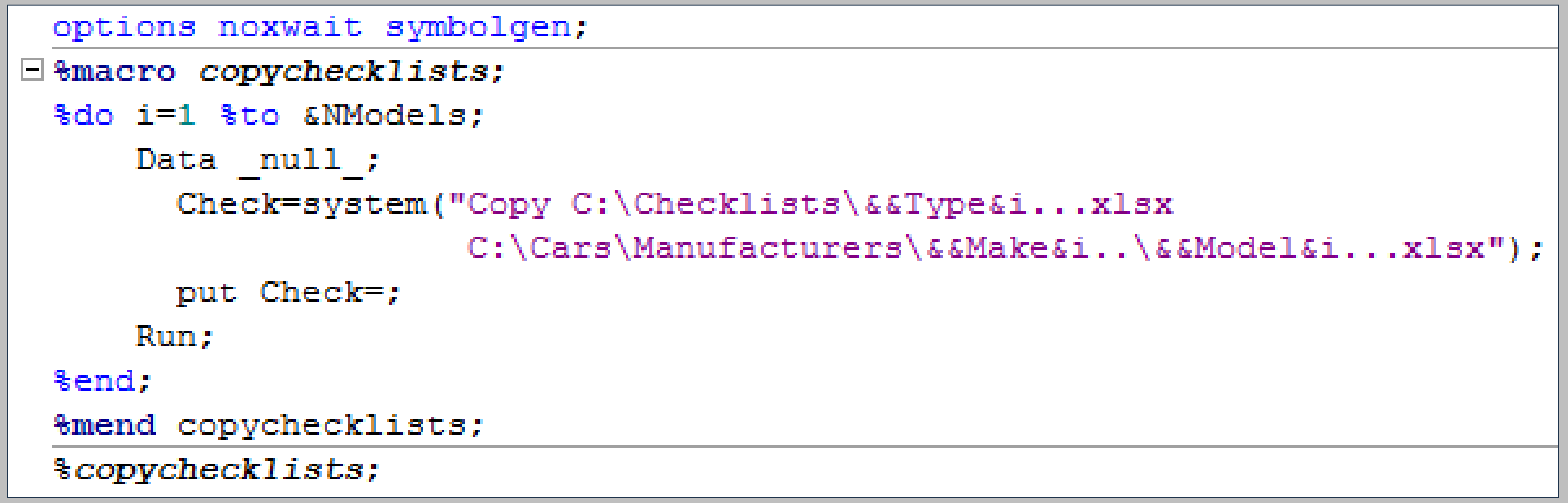
Copy That! - Using SAS® to Create Directories and Duplicate Files

Nicole Ciaccia
Educational Testing Service

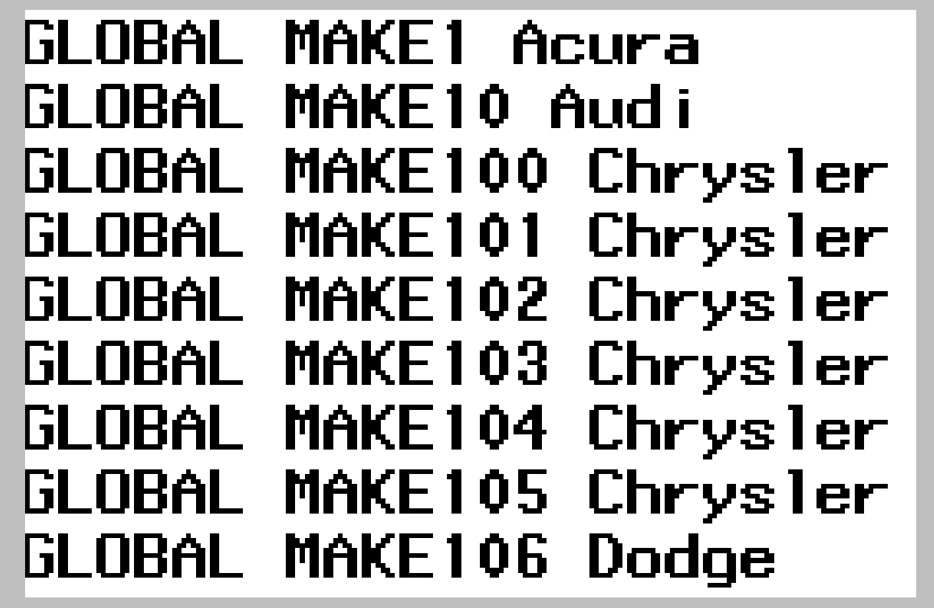
Duplicating Multiple Files

- **System Copy Method**

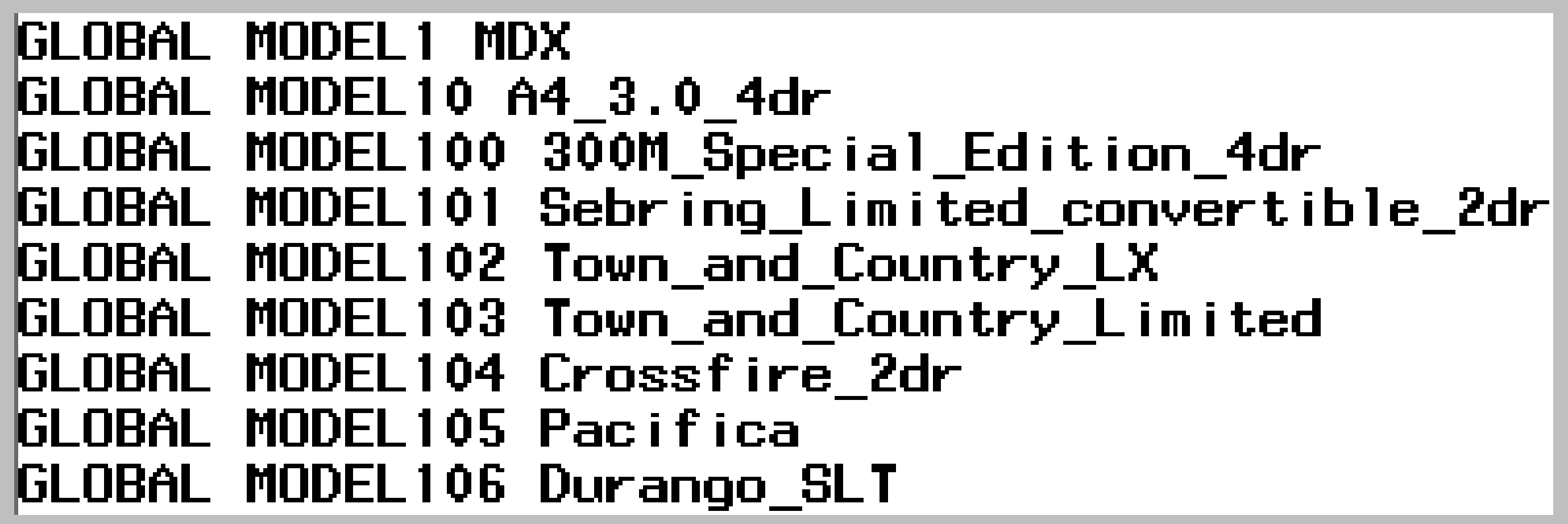
  Step 3: Now that all macro variables have been created, a SYSTEM Copy statement can be used inside a macro, and these variables are inserted each time the do loop cycles through:



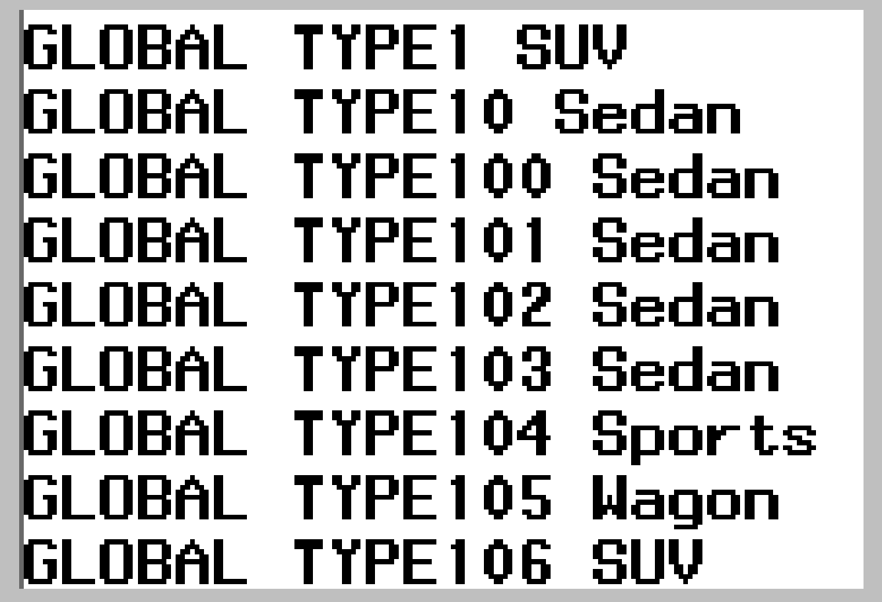
```
options noxwait symbolgen;
%macro copychecklists;
%do i=1 %to &NModels;
    Data _null_;
      Check=system("Copy C:\Checklists\&&Type&i...xlsx
                    C:\Cars\Manufacturers\&&Make&i..\&&Model&i...xlsx");
      put Check=;
    Run;
%end;
%mend copychecklists;
%copychecklists;
```



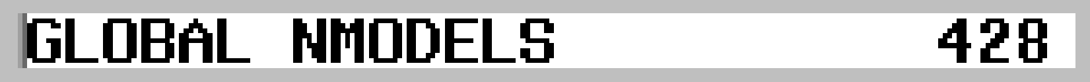
```
GLOBAL MAKE1 Acura
GLOBAL MAKE10 Audi
GLOBAL MAKE100 Chrysler
GLOBAL MAKE101 Chrysler
GLOBAL MAKE102 Chrysler
GLOBAL MAKE103 Chrysler
GLOBAL MAKE104 Chrysler
GLOBAL MAKE105 Chrysler
GLOBAL MAKE106 Dodge
```



```
GLOBAL MODEL1 MDX
GLOBAL MODEL10 A4_3.0_4dr
GLOBAL MODEL100 300M_Special_Edition_4dr
GLOBAL MODEL101 Sebring_Limited_convertible_2dr
GLOBAL MODEL102 Town_and_Country_LX
GLOBAL MODEL103 Town_and_Country_Limited
GLOBAL MODEL104 Crossfire_2dr
GLOBAL MODEL105 Pacifica
GLOBAL MODEL106 Durango_SLT
```

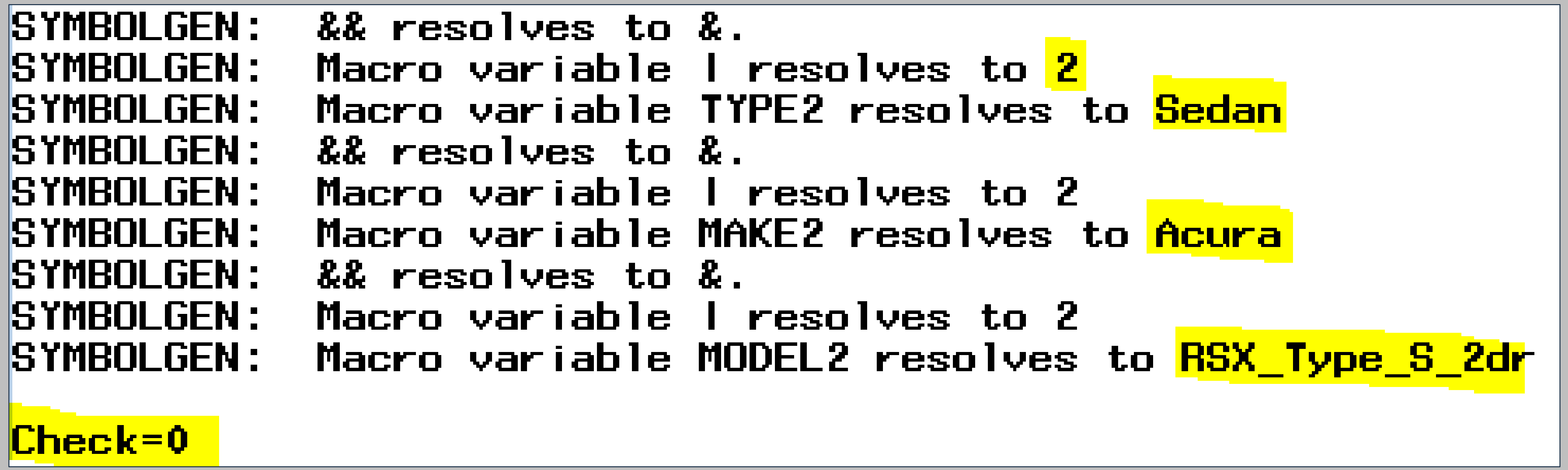


```
GLOBAL TYPE1 SUV
GLOBAL TYPE10 Sedan
GLOBAL TYPE100 Sedan
GLOBAL TYPE101 Sedan
GLOBAL TYPE102 Sedan
GLOBAL TYPE103 Sedan
GLOBAL TYPE104 Sports
GLOBAL TYPE105 Wagon
GLOBAL TYPE106 SUV
```



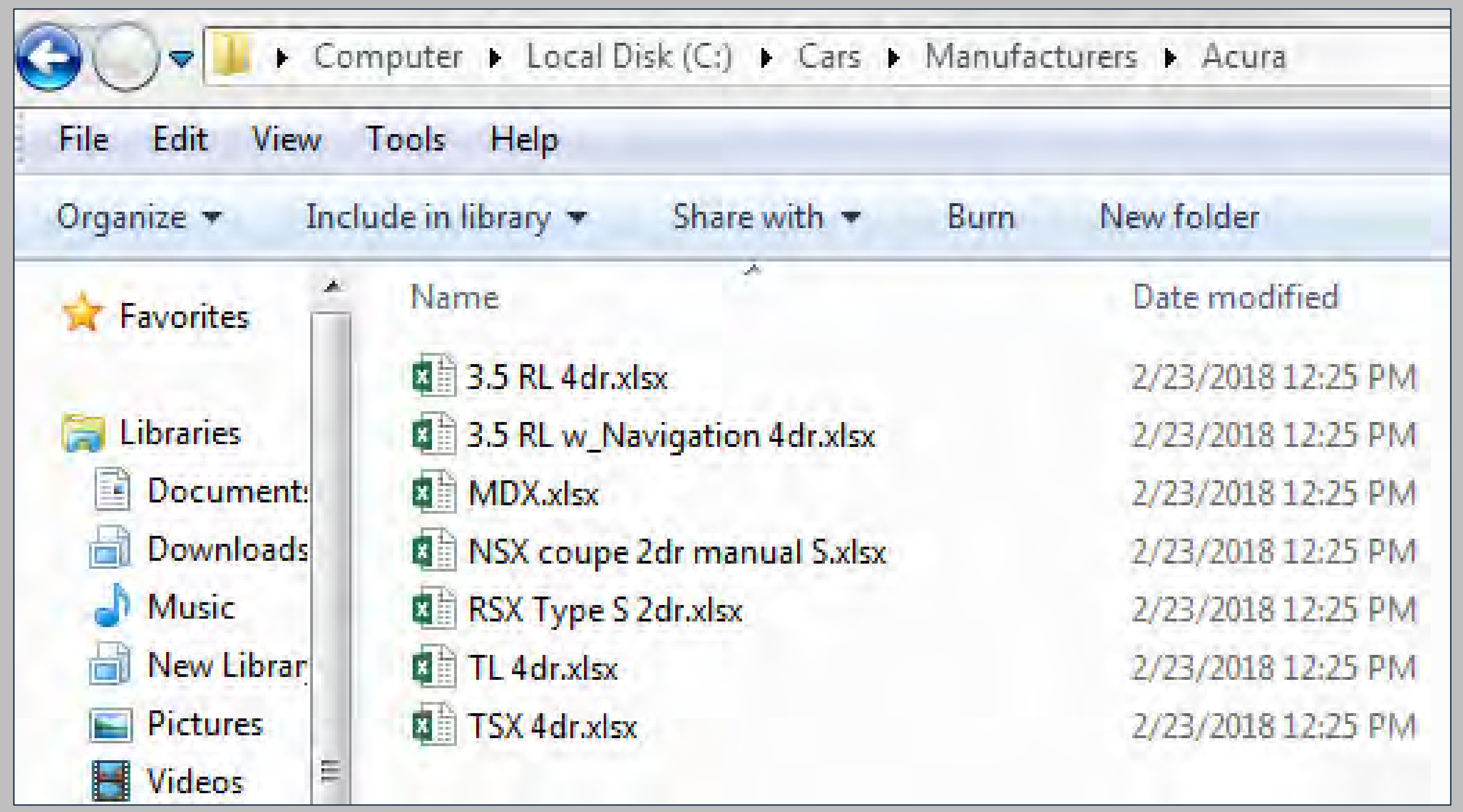
```
GLOBAL NMODELS          428
```

- After running the code, the log can be searched for Check=1 to see if all files were duplicated.
  - (*Symbolgen* option is included-to make troubleshooting easier).
  - Searching for Check=1 yields no results, and the user receives confirmation that all checklists are output.



```
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable I resolves to 2
SYMBOLGEN:  Macro variable TYPE2 resolves to Sedan
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable I resolves to 2
SYMBOLGEN:  Macro variable MAKE2 resolves to Acura
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable I resolves to 2
SYMBOLGEN:  Macro variable MODEL2 resolves to RSX_Type_S_2dr

Check=0
```



Computer ▸ Local Disk (C:) ▸ Cars ▸ Manufacturers ▸ Acura

Name	Date modified
3.5 RL 4dr.xlsx	2/23/2018 12:25 PM
3.5 RL w_Navigation 4dr.xlsx	2/23/2018 12:25 PM
MDX.xlsx	2/23/2018 12:25 PM
NSX coupe 2dr manual S.xlsx	2/23/2018 12:25 PM
RSX Type S 2dr.xlsx	2/23/2018 12:25 PM
TL 4dr.xlsx	2/23/2018 12:25 PM
TSX 4dr.xlsx	2/23/2018 12:25 PM

Conclusion

- Though the coding seen here is relatively simple, it is powerful. It can save the user a great deal of time and assist considerably in arranging files and completing organizational tasks.
- These are just a few of the methods that can be used to create directories and duplicate files, and there is even greater potential for utilizing SAS to interact with an operating system.
- The reference section of the paper associated with this ePoster gives a more in depth look at these commands and functions as well as other ways that they can be used.

SAS® GLOBAL FORUM 2018

April 8 – 11 | Denver, CO
Colorado Convention Center

#SASGF

Paper 2489-2018

Copy That! - Using SAS® to Create Directories and Duplicate Files

Nicole Ciaccia, Educational Testing Service

ABSTRACT

Whether it is part of a client's request or needed for organizational purposes, the creation of several directories is often required. Along those same lines, business requirements sometimes dictate that the same document or checklist be filled out for each individual client or product that a company manages. The monotony of creating and naming folders or doing a save-as function for files is a task that few eagerly anticipate having to do. With a little help from SAS®, you don't have to! In this paper we explore different methods of using SAS system options, functions and commands to create separate directories and duplicate files based on a given data set. First, we investigate multiple ways to create a single directory in SAS. Then, using the SAS data set Sashelp.Cars, we use these functions and commands to create directories for each of the car manufacturers listed in the data set. Finally, we discuss how to use SAS to copy existing files, and, again using the Sashelp.Cars data set as an example, generate a checklist for each car model in the data set.

INTRODUCTION

While SAS can be used to create directories and copy files on several operating systems, this paper will explore doing so using Windows. These concepts can be applied if using other systems, but the directories would be slightly different (ie instead of 'C:\Users\', if using Unix for example, use '/local/u/abcdef/'). In this paper, DLCREATEDIR, DCREATE, X Commands, System Copy and creating a Batch file will be used, but these are just a few of the many ways SAS can be leveraged to assist in file management and organization.

CREATING A SINGLE FOLDER OR DIRECTORY

DLCREATEDIR (a system option)

One of the simplest options that can be used to create a directory is the DLCREATEDIR system option. When the DLCREATEDIR option is turned on and used with a LIBNAME statement, if a libref is assigned to a folder that doesn't exist, SAS will create it. Using this option, a folder can be created with just two lines of code:

```
options dlcreatedir;
libname Folder "C:\Cars";
```

After running this code, a folder named 'Cars' now exists on the C drive. If subfolders are needed, multiple specifications in the LIBNAME statement can be used:

```
options dlcreatedir;
libname Folder ("C:\Cars", "C:\Cars\Manufacturers");
```

Note: It is important to put the individual file paths in the proper order - Manufacturers can't be created under the Cars directory if Cars has not been created first. For this same reason, "C:\Cars\Manufacturers" could not be used alone in the LIBNAME statement if C:\Cars did not yet exist.

Since these SAS librefs were only needed to create folders, they can be cleared afterward if desired:

```
libname Folder clear;
```

DCREATE (a function)

Another way to create a directory is using the DCREATE function. The syntax is as follows:

new-directory=DCREATE(*'new-directory-name', 'parent-directory');*

New-directory-name is the name of the directory that will be created. It can be either a character constant, a variable or expression. Parent-directory is the path where the new directory will be created. This argument is optional. If a parent directory is not specified in the DCREATE function, the current directory is the default and the folder will be output there.

Since DCREATE is a function, it can be used in a data step. Consider the following code:

```
Data _null_;
    NewDirectory=dcreate('Cars','C:\');
Run;
```

In this case, a new folder, Cars, is created on the C drive.

Using X Statements (X 'md')

The X statement can be used to give the operating system a command directly from SAS. After submitting an X statement, the SAS session is exited and the processor is temporarily accessed. The syntax is X '*command'* where command is the desired command to give the operating system. There are a variety of these commands that can be issued, but we will specifically be looking at X '*md'* which **m**akes **d**irectories. When using X 'md' the syntax is:

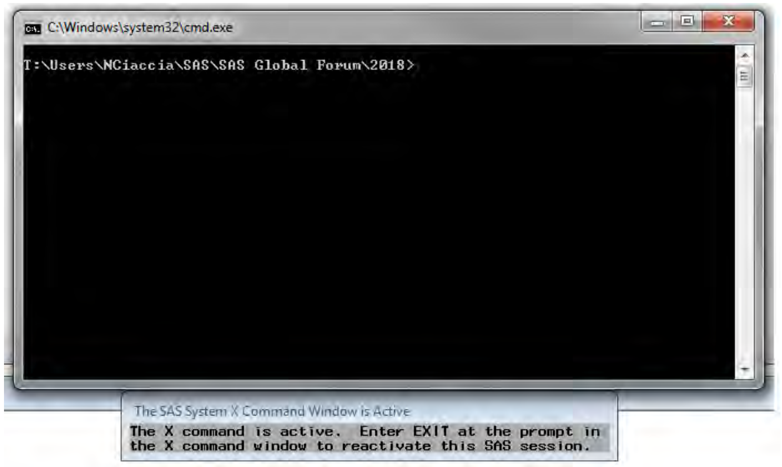
X 'md *new-directory-file-path*';

For example, the code shown below would instruct Windows to leave SAS temporarily to create a folder, 'Cars' and a subfolder, 'Manufacturers' on the C: drive:

```
Data _null_;
    X 'md C:\Cars\Manufacturers';
Run;
```

Notice this is different from using DLCREATEDIR in that both the Cars and Manufacturers folders can be created at the same time with one X statement.

When issuing an X statement, a DOS Window automatically opens and submits the command. In order to close the DOS Window, you must type 'exit.'


C:\Windows\system32\cmd.exe
The SAS System X Command Window is Active

```
T:\Users\NCiaccia\SAS\SAS Global Forum\2018>
```

```
The X command is active.  Enter EXIT at the prompt in
the X command window to reactivate this SAS session.
```

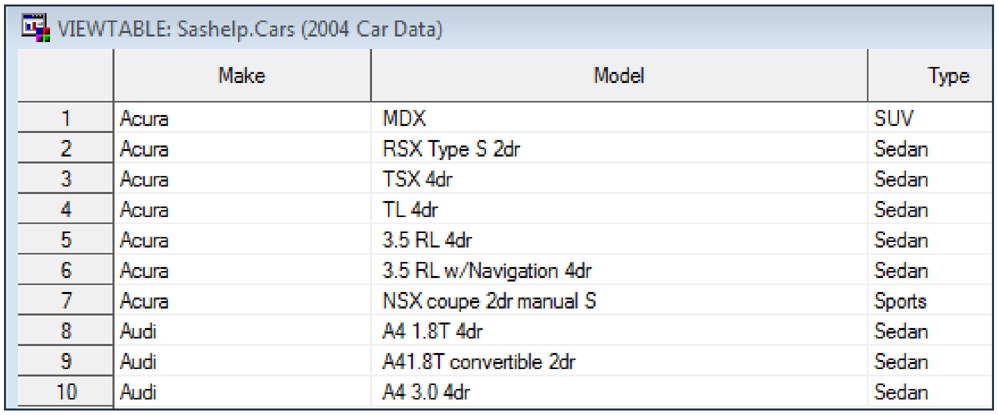
To avoid having to do this and have the DOS window close automatically, the *noxwait* option can be added. This allows the processor to go back to the SAS session automatically so that the program will continue to run without the user having to do anything else. Adding *noxwait* to the previous example:

```
options noxwait;
Data _null_;
    X 'md C:\Cars\Manufacturers';
Run;
```

CREATING MULTIPLE FOLDERS OR DIRECTORIES

Now that creating a single directory has been explored, this knowledge can be extended to create multiple directories based off of the SAS data set SAShelp.Cars. The coding below uses DLCREATEDIR and the CALL EXECUTE routine to do so. While any of the above methods could be used, DLCREATEDIR was chosen because it is the simplest method.

The objective is to create a folder for each of the car manufacturers listed in the data set under the variable Make.



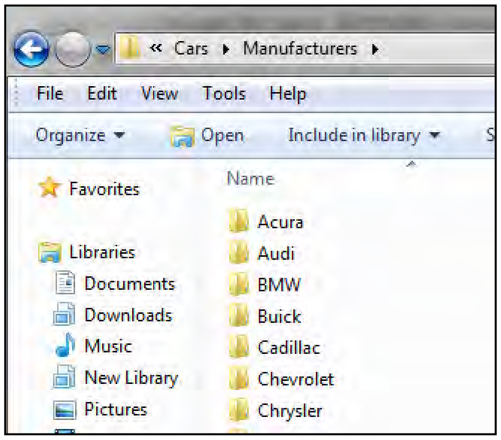
VIEWTABLE: Sashelp.Cars (2004 Car Data)

	Make	Model	Type
1	Acura	MDX	SUV
2	Acura	RSX Type S 2dr	Sedan
3	Acura	TSX 4dr	Sedan
4	Acura	TL 4dr	Sedan
5	Acura	3.5 RL 4dr	Sedan
6	Acura	3.5 RL w/Navigation 4dr	Sedan
7	Acura	NSX coupe 2dr manual S	Sports
8	Audi	A4 1.8T 4dr	Sedan
9	Audi	A41.8T convertible 2dr	Sedan
10	Audi	A4 3.0 4dr	Sedan

In the coding below, SAS cycles through the data set and executes a LIBNAME statement for each observation. The corresponding value of Make for each observation is inserted into the LIBNAME statement as the name of the folder to be created under C:\Cars\Manufacturers (which already exists from the previous coding above). As SAS cycles through the data set, each of the desired folders is created as necessary:

```
options dlcreatedir;
Data _null_;
Set Sashelp.Cars;
      Call Execute("libname Folder 'C:\Cars\Manufacturers\" || Make || "';");
Run;
```

Though there are repeated values of the Make variable in the data set, each folder is only created the first time the value is encountered during the data step. Hence, the repeated values will not add much more processing time. However if desired, the data set could be copied and sorted using the nodupkey option to eliminate any duplicates.



DUPLICATING FILES

Using X Statements (X ‘copy’)

Similar to how X ‘md’ can create directories, the X statement can also be used to duplicate files. Instead of putting md in the command argument, the Copy command can be used in order to duplicate files. The syntax in this context is:

X ‘Copy *original-file output-file*’;

Original-file is the file path of the document that is to be duplicated. Output-file is the file path where the duplicated file should be placed.

In the code below, the Document1 excel file located in C:\Input is specified as the file to duplicate, and it should be copied to C:\Output, this time named Document2. Again, the noxwait option is included so that the DOS window closes automatically:

```
options noxwait;
Data _null_;
  X 'Copy C:\Input\Document1.xlsx C:\Output\Document2.xlsx';
Run;
```

It is important to note that X commands are global statements so they cannot be executed conditionally.

System Copy

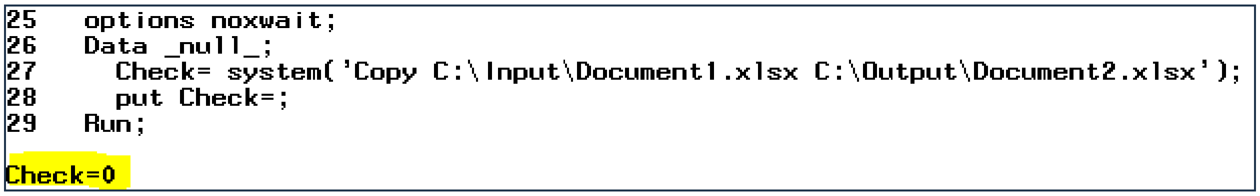
Similar to X Commands, the SYSTEM function can also be utilized to instruct the operating system to perform a desired command. The syntax for this function is System (*command*). Similar to an X statement, the command argument in the SYSTEM function can use the same format:

System (‘Copy *original-file output-file*’);

A helpful attribute of the SYSTEM function is that it generates a return code that denotes if the command executed or not. This return code is dependent on the operating environment. By default, for Windows 7, SYSTEM will return 0 if the command executes properly and 1 if it does not. This return code can then be leveraged as a future condition (ie if command processed successfully, continue. If not, stop processing the code). In the coding below, the same Document1 file will again be duplicated, but this time a variable, *check*, is created that stores the return code and states whether or not the command executed:

```
options noxwait;
Data _null_;
  Check= system('Copy C:\Input\Document1.xlsx C:\Output\Document2.xlsx');
  put Check=;
Run;
```

After running this code, the log states that Check=0, meaning the command executed properly.



```
options noxwait;
Data _null_;
  Check= system('Copy C:\Input\Document1.xlsx C:\Output\Document2.xlsx');
  put Check=;
Run;

Check=0
```

Unlike X commands, the SYSTEM function can be used conditionally if desired. The coding below would only execute successfully on Mondays. All other days, the file would not copy:

```
options noxwait;
Data _null_;
   if "&sysday"="Monday" then do;
      Check=system('Copy C:\Input\Document1.xlsx C:\Output\Document2.xlsx');
   end;
put Check=;
Run;
```

Create a Batch file

Finally, SAS can be used to output a batch file that can then be run to duplicate files. A batch file is essentially a script that tells the operating system what to do, made up of a series of commands to be performed by the processor. For Windows, the extension for this plain text file is .bat. When the batch file is double clicked, the commands contained within are carried out. This file can be created manually by typing the appropriate commands in a text editor, such as Notepad or Textpad, and saving it with the .bat extension. SAS can be used to create this file automatically.

Within a datastep, we define a variable, *final*, and enter in the desired command with the same syntax as when System Copy was used: 'Copy *original-file output-file*'. The output location for the batch file is defined with a file statement, and the put statement specifies that the 'final' variable should be written out to this file:

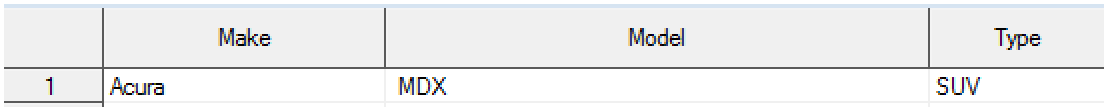
```
Data _null_;
   final='Copy C:\Input\Document1.xlsx C:\Output\Document2.xlsx';
   file "C:\Input\setup.bat";
      put @1 final;
Run;
```

The folder C:\Input now contains a batch file named setup. Once the file is double clicked on, Document1.xlsx will be copied into C:\Output as Document2.xlsx.

Note: the batch file remains in C:\Input and should be deleted after it has been run. (If desired, it is possible to add coding to have the .bat automatically delete after being run.)

Duplicating Multiple Files

While using SAS to copy a single file doesn't seem particularly useful, the true potential for this coding becomes clear when applied to a more authentic scenario. Returning to the Sashelp.Cars data set, there are 6 different types of cars listed in the Type variable – Hybrid, Sedan, Sports, SUV, Truck and Wagon. Suppose a company has separate checklists for each vehicle type and needs to output the correct checklist in the appropriate Make folder for each Model of car. Using the first observation as an example, an SUV checklist would need to be output in the Acura folder with the name MDX.

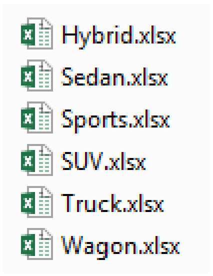


	Make	Model	Type
1	Acura	MDX	SUV

There are 428 observations in the Cars data set. How can all of these checklists be output in an efficient manner? The folders all exist from the previous exploration of using DLCREATEDIR. The task at hand then becomes copying the correct checklist and placing it in the correct folder with the correct name. The coding strategies for duplicating referenced above can be extended to copy multiple files and fulfill this business need.

First let's examine creating these checklists using a batch file. Remembering that the syntax for the command contained in the batch file is 'Copy *original-file output-file*', SAS needs to cycle through the Cars data set and output a line for each time a file needs to be copied with the corresponding file to be copied, output file name, and output file location.

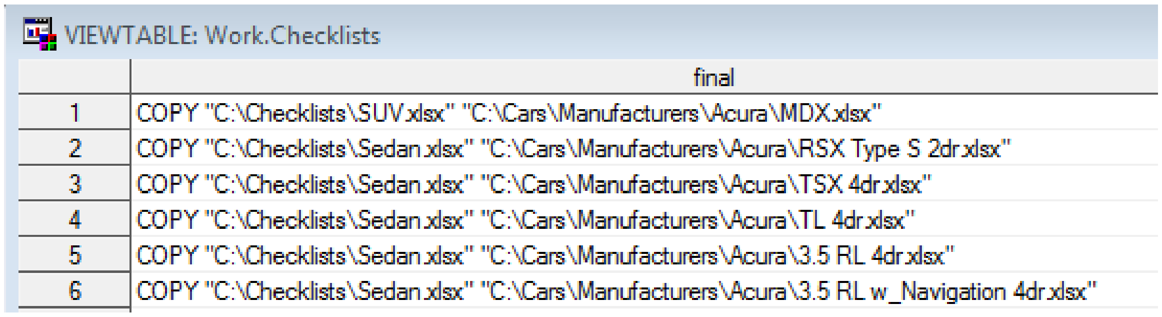
Assume a copy of each of the checklists exists as an .xlsx document in C:\Checklists.


Hybrid.xlsx
Sedan.xlsx
Sports.xlsx
SUV.xlsx
Truck.xlsx
Wagon.xlsx

The code below once again defines a variable, final, that is equal to the desired command. In this case, final is set up as a shell, concatenating certain parts of the statement that will stay the same in every line, but also inserting the 3 variables that will change depending on each observation:

```
Data Checklists (keep=final);
Set Sashelp.cars;
    Model=translate(Model,'_','/');
    final=cat('COPY "','C:\Checklists\',strip(Type),'.xlsx" ',
              '"C:\Cars\Manufacturers\',strip(Make),'\',strip(Model),'.xlsx"');
Run;
```

Notice that the CAT version of the concatenate function is used instead of CATS because any leading or trailing spaces are included intentionally and we do not want them stripped before concatenating. The STRIP function is used with each of the variables so that any unintentional leading or trailing spaces within these values are stripped. Also, there are car models in the data with forward slashes (ie. Observation 6, 3.5 RL w/Navigation 4dr). Since Model will be used as the file name and Windows doesn't allow forward slashes in file names, these are replaced with underscores using the TRANSLATE function. Once this code is run, the data set Checklists looks like this:


VIEWTABLE: Work.Checklists

	final
1	COPY "C:\Checklists\SUV.xlsx" "C:\Cars\Manufacturers\Acura\MDX.xlsx"
2	COPY "C:\Checklists\Sedan.xlsx" "C:\Cars\Manufacturers\Acura\RSX Type S 2dr.xlsx"
3	COPY "C:\Checklists\Sedan.xlsx" "C:\Cars\Manufacturers\Acura\TSX 4dr.xlsx"
4	COPY "C:\Checklists\Sedan.xlsx" "C:\Cars\Manufacturers\Acura\TL 4dr.xlsx"
5	COPY "C:\Checklists\Sedan.xlsx" "C:\Cars\Manufacturers\Acura\3.5 RL 4dr.xlsx"
6	COPY "C:\Checklists\Sedan.xlsx" "C:\Cars\Manufacturers\Acura\3.5 RL w_Navigation 4dr.xlsx"

A line is output for each observation in the Cars data set with directions to copy the correct checklist and output it to the appropriate folder with the appropriate file name.

Adding in a file and a put statement to the coding above results in the batch file being output:

```
Data Checklists (keep=final);
Set Sashelp.cars;
    Model=translate(Model,'_','/');
    final=cat('COPY "','C:\Checklists\',strip(Type),'.xlsx" ',
              '"C:\Cars\Manufacturers\',strip(Make),'\',strip(Model),'.xlsx"');
    file "C:\Cars\.setup.bat";
    put @1 final;
Run;
```

Once the batch file is run, the checklists will appear in each of the Manufacturers folders.

While creating a batch file is relatively simplistic in terms of coding, one drawback of using this method is that the SAS log only displays if the batch file was created correctly. It does not state if the commands in the batch file performed as expected, since that is done outside of SAS. If it is important to the user to be able to check that the files were duplicated successfully without going into the folders and checking manually, the SYSTEM Copy method might be better suited.

To accomplish this, we will use the CALL SYMPUT routine to make variables out of the Make, Model, Type, and observation number for each observation in the data set. Then, these variables will be inserted into a SYSTEM COPY statement that is contained within a do loop macro. Using a do loop macro allows SAS to cycle through and execute the SYSTEM COPY function for each observation in the data set.

Because the values of Type, Model and Make will be used within SYSTEM, some data manipulation needs to occur before using them to create variables in order for the code to run successfully. The duplication will not complete if there are spaces in the values of variables, so all spaces need to be changed to underscores or compressed. Additionally, just like when creating the batch file, no forward slashes can be included in the final file names, so these will once again be replaced using the TRANSLATE function:

```
Data CarModels (keep=make model type);
Set Sashelp.cars;
    Model=(tranwrd(strip(Model), ' ', '_'));
    Model=translate(Model,'_','/');
    Make=compress(Make);
Run;
```

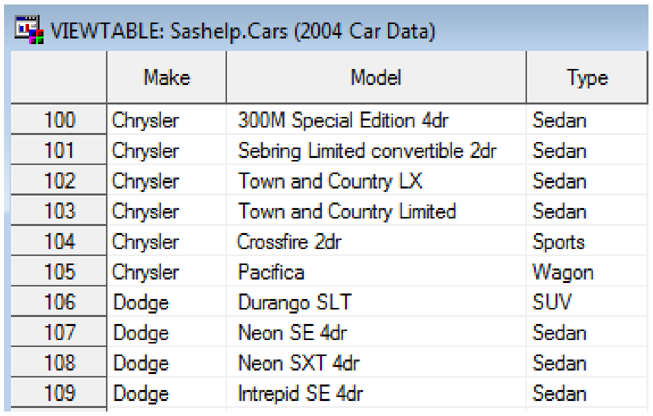
The Cars data set is read in and the Model variable is stripped of all trailing and leading blanks. Then all remaining spaces, (such as those in between words), are changed to underscores using the TRANWRD function. All forward slashes are changed to underscores using the TRANSLATE function, and then all spaces in the Make variable are eliminated using the COMPRESS function. (This statement is needed because the car manufacturer Land Rover has a space and if the space is left, the checklists will not output. When the folders were created, compress was not used. The name of this folder can be manually changed to LandRover since it is only one manufacturer with this issue, or if this was an issue for multiple manufacturers, the Make variable could have been compressed when creating the folders initially.)

Now that the necessary manipulations to the data have been made, they can each be turned into macro variables using CALL SYMPUT:

```
Data _null_;
Set CarModels end=last;
    call symput(cats("Make",_N_),strip(Make));
    call symput(cats("Model",_N_),strip(Model));
    call symput(cats("Type",_N_),strip(Type));
         if last then do;
            call symput("NModels", _N_);
         end;
Run;
```

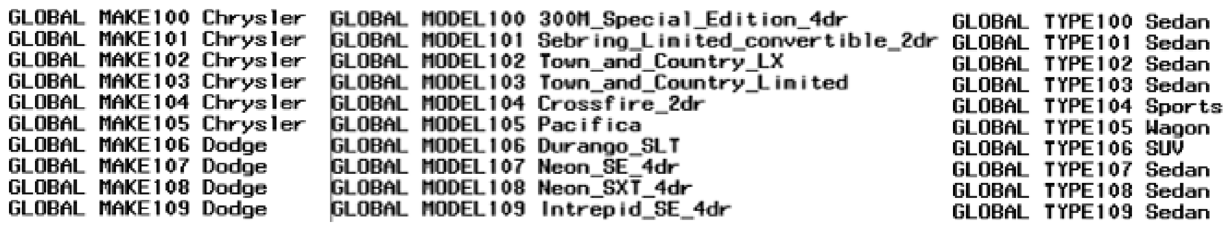
In the first Call SYMPUT statement, a macro variable is created for each observation in the data set based off of the variable Make. The name of the macro variable is the concatenation of the word "Make" with the observation number. The value each macro variable is defined as is the corresponding value of Make for that observation. STRIP is used to ensure there are no extra blank spaces. Since Model and Type will also be used, two more CALL SYMPUT statements are needed to repeat the process for these variables as well.

Using the screen shots below as an example, the macro variable Make100 resolves as Chrysler because Chrysler is the 100th observation in the data set. The same is true for Model100 and Type100 - they correspond to the values of Model and Type for the 100th observation.



VIEWTABLE: Sashelp.Cars (2004 Car Data)

	Make	Model	Type
100	Chrysler	300M Special Edition 4dr	Sedan
101	Chrysler	Sebring Limited convertible 2dr	Sedan
102	Chrysler	Town and Country LX	Sedan
103	Chrysler	Town and Country Limited	Sedan
104	Chrysler	Crossfire 2dr	Sports
105	Chrysler	Pacifica	Wagon
106	Dodge	Durango SLT	SUV
107	Dodge	Neon SE 4dr	Sedan
108	Dodge	Neon SXT 4dr	Sedan
109	Dodge	Intrepid SE 4dr	Sedan



```
GLOBAL MAKE100 Chrysler   GLOBAL MODEL100 300M_Special_Edition_4dr          GLOBAL TYPE100 Sedan
GLOBAL MAKE101 Chrysler   GLOBAL MODEL101 Sebring_Limited_convertible_2dr   GLOBAL TYPE101 Sedan
GLOBAL MAKE102 Chrysler   GLOBAL MODEL102 Town_and_Country_LX               GLOBAL TYPE102 Sedan
GLOBAL MAKE103 Chrysler   GLOBAL MODEL103 Town_and_Country_Limited          GLOBAL TYPE103 Sedan
GLOBAL MAKE104 Chrysler   GLOBAL MODEL104 Crossfire_2dr                     GLOBAL TYPE104 Sports
GLOBAL MAKE105 Chrysler   GLOBAL MODEL105 Pacifica                          GLOBAL TYPE105 Wagon
GLOBAL MAKE106 Dodge      GLOBAL MODEL106 Durango_SLT                       GLOBAL TYPE106 SUV
GLOBAL MAKE107 Dodge      GLOBAL MODEL107 Neon_SE_4dr                       GLOBAL TYPE107 Sedan
GLOBAL MAKE108 Dodge      GLOBAL MODEL108 Neon_SXT_4dr                      GLOBAL TYPE108 Sedan
GLOBAL MAKE109 Dodge      GLOBAL MODEL109 Intrepid_SE_4dr                   GLOBAL TYPE109 Sedan
```

In addition to defining the values of the variables that will be inserted into the SYSTEM COPY statement, SAS will also need to know how many times to cycle through the do loop. The coding above identifies the last observation in the data set and then that observation number is stored in the macro variable, NModels.

Now that all of the appropriate macro variables have been created, a SYSTEM Copy statement can be used inside a macro, and these variables are inserted each time the do loop cycles through:

```
options noxwait symbolgen;
%macro copychecklists;
%do i=1 %to &NModels;
    Data _null_;
      Check=system("Copy C:\Checklists\&&Type&i...xlsx
                   C:\Cars\Manufacturers\&&Make&i..\&&Model&i...xlsx");
      put Check=;
    Run;
%end;
%mend copychecklists;
%copychecklists;
```

This code takes a few minutes to run, but once it is complete, all checklists are present in the correct folders. While the code takes slightly longer and the file names don't look exactly the same as in the data set (they do with batch file), the log can easily be scanned for Check=1 to see if all files were duplicated as expected. (This is why the SYMBOLGEN option is included-to make troubleshooting easier). Searching for Check=1 yields no results, and the user receives confirmation that all checklists are output.

CONCLUSION

The coding included in this paper is relatively simple, and often the desired task can be accomplished in just a few lines of code. Though short, it is powerful code that can save the user a great deal of time and assist considerably in arranging files and completing organizational tasks. These are just a few of the methods that can be used to create directories and duplicate files, and there is even greater potential for utilizing SAS to interact with an operating system. The references included at the end of this paper give a more in depth look at these commands and functions as well as other ways that they can be used.

REFERENCES

Hemedinger, C. (2013, July 2). SAS Trick: get the LIBNAME statement to create folders for you [Blog post]. Retrieved from http://blogs.sas.com/content/sasdummy/2013/07/02/use-dlcreatedir-to-create-folders/

Hurley, G. (2006). Xamining the X Statement (and Some Other Xciting Code) [Paper no. 036-31]. In *SUGI 31 Proceedings, San Francisco, CA.* Retrieved from http://www2.sas.com/proceedings/sugi31/036-31.pdf

Jia, J., & Lin, A. (2015). Yes, SAS® can do! ---Manage external files with SAS programming [Paper no. 3262-2015]. In *Proceedings of the SAS Global Forum, Dallas TX.* Retrieved from http://support.sas.com/resources/papers/proceedings15/3262-2015.pdf

Patrick. (2011, December 20). *Copying and renaming a file* [Blog post]. Retrieved from https://communities.sas.com/t5/SAS-Procedures/Copying-and-renaming-a-file/td-p/17715

SAS Institute Inc. (2009). DCREATE. In *SAS® Component Language 9.2: Reference.* Retrieved from http://support.sas.com/documentation/cdl/en/sclref/59578/HTML/default/viewer.htm#a000308115.htm

SAS Institute Inc. (2010). Running Windows or MS-DOS commands from within SAS*.* In *SAS® 9.2 Companion for Windows*(2nd ed.). Retrieved from http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#exittemp.htm

SAS Institute Inc. (2010). X Command: Windows*.* In *SAS® 9.2 Companion for Windows*(2nd ed.). Retrieved from http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#win-cmd-x.htm

SAS Institute Inc. (2011). SYSTEM Function. In In *SAS® 9.2 Language Reference: Dictionary*(4th ed.). Retrieved from http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000245956.htm

ACKNOWLEDGEMENTS

Special thanks go out to Stephanie Fournier-Zajac, Matthew Duchnowski and Christopher Kelbaugh of Educational Testing Service for their inspiration, guidance and feedback on this paper. I am extremely grateful for all of the advice you have provided along the way.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Nicole Ciaccia
Educational Testing Service
660 Rosedale Rd, MS 14P
Princeton, NJ 08541
(609) 734-1612
nciaccia@ets.org