

# **Using SPSS Scripting to Create Flat Files for Mandated Reporting**

Viktor Brenner, Ph.D.  
Waukesha County Technical College

## **The Problem**

External data systems often require data be submitted as fixed-format text files, also known as "flat files." Flat files are often large and created on a regular basis, making them good candidates for an automated process. However, every variable must line up exactly for every case or the flat file will be rejected.

## **The Solution**

SPSS syntax was written that takes a data file and creates an output file that conforms to the required file layout. Syntax is a command language for SPSS, a direct descendant of the original SPSSx mainframe user interface.

The sample that follows creates an Enrollment Search file for use with the National Student Clearinghouse. Colleges can submit lists of its students and determine whether those students have transferred to other institutions.

## **Business Results Achieved**

Students that transfer to other institutions are not counted when calculating the student graduation rates. WCTC increased its reported graduation rate by 25% the first year that it used the Clearinghouse to identify transfers.

# Source Data File

	PIDM	First	Last	Middle	dob	var	var	var	var	var	var
1	4	George	Ruth Jr	Herman	06-FEB-1895						
2	99	Wayne	Gretzky	Douglas	26-JAN-1961						
3	12345	Sammy	Davis Jr		12-AUG-1925						
4	234550	Chan Ho	Park		30-JUN-1973						
5	234567	Thurston	Howell III		10-OCT-1910						
6	300100	Kerri	Walsh		15-AUG-1978						
7	300101	Misty	May-Treanor		30-JUL-1977						
8	398239	John	Mellor	G	21-AUG-1952						
9	456789	Johnny	Vander Meer		02-NOV-1914						
10	488600	Gwen	Stefani	Renee	03-OCT-1969						
11	555555	Samuel L	Jackson		21-DEC-1948						
12	974339	Manfred	von Richthofen	A	02-MAY-1892						
13	968733	Loreena	McKennitt		17-FEB-1957						
14	3498477	Sarah	McLachlan	Ann	28-JAN-1968						
15	4986402	Peter	Murphy		11-JUL-1957						
16	6978400	Andrea	Corr	Jane	17-MAY-1974						
17	7995799	Gwyneth	Paltrow	Kate	28-SEP-1972						
18	8675309	William	Clinton	Jefferson	19-AUG-1946						
19	8996004	Lindsey	Lohan	Morgan	02-JUL-1986						
20	9483489	Nicole	Kidman	M	20-JUN-1967						
21											
22											
23											
24											

**NAME NORMALIZATION**  
 Names may not be in standard format in the source data file; the syntax normalizes them (see output file below). For instance, Samuel L Jackson is parsed into a middle name but Chan Ho Park is not; a suffix is created for Thurston Howell III but not for Johnny Vander Meer.

## Final Product

The submission file has a header identifying the college, a number of data records in fixed format, and a footer confirming how many records should have been read.

```
H100529400Waukesha County Technical College      2005 928SEI
D1      George      HRuth      Jr      0206189520021015 005294004
D1      Wayne      DGretzky      0126196120021015 0052940099
D1      Sammy      Davis      Jr      0812192520021015 0052940012345
D1      Chan Ho      Park      0630197320021015 00529400234550
D1      Thurston      Howell      III      1010191020021015 00529400234567
D1      Kerri      Walsh      0815197820021015 00529400300100
D1      Misty      May-Treanor      0730197720021015 00529400300101
D1      John      GMellor      0821195220021015 00529400398239
D1      Johnny      Vander Meer      1102191420021015 00529400456789
D1      Gwen      RStefani      1003196920021015 00529400488600
D1      Samuel      LJackson      1221194820021015 00529400555555
D1      Manfred      Avon Richthofen      0502189220021015 00529400974339
D1      Loreena      McKennitt      0217195720021015 00529400988733
D1      Sarah      AMcLachlan      0128196820021015 005294003498477
D1      Peter      Murphy      0711195720021015 005294004986402
D1      Andrea      JCorr      0517197420021015 005294006978400
D1      Gwyneth      KPaltrow      0928197220021015 005294007995799
D1      William      JClinton      0819194620021015 005294008675309
D1      Lindsey      MLohan      0702198620021015 005294008996004
D1      Nicole      MKidman      0620196720021015 005294009483489
T100000022
```

```
*=====
* SPSS Code for creating a submission file for the
* National Student Clearinghouse
*=====
```

\* **===== Key to reading this output =====.**

Lines that begin with a star "\*" are comments and are not read by the command interpreter

\* **===== User input needed looks like this.**

\* **===== Input/output commands look like this.**

\* **===== Comments annotating how the script works look like this.**

\* **===== Program control commands look like this.**

\* **===== Commands for normalizing names look like this.**

\* **===== Commands for creating the birthdate string look like this, this, or this.**

\* **===== Main commands look like this.**

\* **===== Begin Clearinghouse Script Main =====.**

\* **===== Start by importing the data file.**

\* **===== Name the variables in data file PIDM, term, first, middle, last, dob.**

\* **===== You will have to hard-code last day of attendance (ldate) below.**

string ldate (A8).

**compute ldate='20021015'.**

exe.

\* **===== There will be several read and write operations in this script.**

\* **===== We will turn off results so we don't have to see all the details.**

**Set results=off.**

\* **===== Initialize other variables, including transformed names.**

String type (A2). /\*record type.

compute type='D1'.

string SSN (A9).

compute SSN=' '.

String f\_name (a20).

SSN is no longer used by the Clearinghouse, so a string of 9 spaces is defined to fill the space.

```

string m_init (A1).
string l_name (A20).
string suffix (a5).
string fil_2 (A1).
string schcode (A6).
string brchcode (A2).
string reqrtrn (A50).
string fil_3 (A255).
string fil_4 (a113).
exe.

```

\* ===== Assign names, set fixed values.

```

compute f_name=rpad(ltrim(first),20).
compute m_init=rpad(substr(middle,1,1),1).
compute l_name=rpad(ltrim(last),20).
compute schcode='005294'.
compute brchcode='00'.
compute reqrtrn=ltrim(string(pidm,F9.0)).
exe.

```

Ltrim removes extraneous spaces from the left of the name, rpad fills the string with spaces so that it is exactly 20 characters.

\* ===== NORMALIZING NAMES.

\* ===== Extracts suffixes & moves middle initial if in the "first" field.

```

compute L1stblnk = index(l_name, ' ').
compute F1stblnk = index(f_name, ' ').
string jrorIII (A3).
if (substr(f_name,F1stblnk+1) ne "") jrorIII=substr(f_name,(F1stblnk+1),3).
execute.

```

Finds the first space character in the name strings.

```

do if jrorIII = 'Jr' or jrorIII = 'II' or jrorIII= 'III' or jrorIII= 'IV'.
Compute Suffix = rpad(ltrim(jrorIII),5) .
compute f_name=rpad (substr(f_name,1,f1stblnk-1),20).
else if M_init="" and substr(jrorIII,2,1)= ".
Compute M_init=substr(jrorIII,1,1).
compute f_name=rpad(substr(f_name,1,f1stblnk-1),20).
end if.
execute.

```

If the character after the first space is not also a space, checks if the rest of the string is a name suffix. If it is, the suffix is moved to a separate field.

If middle name is blank and a single letter follows the first name, that letter is moved to middle name.

```

if substr(l_name,l1stblnk+1) ne " jrорlll=substr(l_name,(L1stblnk+1),3).
execute.
do if jrорlll = 'Jr' or jrорlll = 'll' or jrорlll= 'lll' or jrорlll= 'IV'.
Compute Suffix = rpad(ltrim(jrорlll),5) .
compute l_name=rpadd(substr(l_name,1,l1stblnk-1),20).
end if.
execute.

```

Repeats suffix  
check on last  
name.

```

* ===== This section defines BDAY one of 3 ways.
* ===== depending the format of the source variable. At least one will.
* ===== result in type mismatch error, so we turn error messages off.

```

set errors = off.

```

* ===== Creates BDAY when dob is a number, like 3121997.
* ===== One possible problem is that a date is read as a number.
* ===== The largest number that a date can generate is 12312005.
* ===== SPSS encodes a date by the number of seconds since.
* ===== midnight Oct 14 1582, thus dates tend to exceed 1.1 x 1010.
* ===== If the source data comes from Excel, dates will be encoded.
* ===== as the days since Jan 1 1900, producing values under 40000.
* ===== Testing for values in-between will catch dates read as numbers.

```

```

do if ((dob lt 13000000) and (dob gt 1000000)).
string bday (a8).
compute bday=string(dob,F8.0).
if missing(dob) bday='      '.
compute bday=lpad(ltrim(rtrim(bday)),8,'0').
end if.
exe.

```

The logic behind  
this do if statement  
is explained here.

\* ===== Alternate coding if DOB is in date format.

```
do if (dob lt 1000000) or (dob gt 13000000).
string datetemp (a4).
compute temp = xdate.month(dob).
formats temp (f4.0).
compute datetemp = ltrim(string(temp,F4.0)).
if temp < 10 datetemp = concat ('0',datetemp).
compute bday=(datetemp).
compute temp = xdate.mday(dob).
compute datetemp = ltrim(string(temp,F4.0)).
if temp < 10 datetemp = concat ('0',datetemp).
compute bday = concat (rtrim(bday), datetemp).
compute temp=xdate.year(dob).
compute datetemp=string(temp,F4.0).
compute bday = concat (rtrim(bday),datetemp).
end if.
exe.
```

Numbers (such as the date in this case) are almost always reported as strings in flat files because number formats don't include the leading zeros that flat files require. When converting from a number format, a leading zero must be appended using *concat* (short for concatenate).

\* ===== Alternate coding: creates BDAY when DOB is a string.

```
compute bday=dob.
exe.
set errors = on.
```

We no longer expect any errors, so errors are turned back on

\*===== Invert data birthday string to year-first format required by NSC.

\*===== This has been added since originally presented.

```
compute bday = concat (substr(bday,5,4), substr(bday,1,4)).
exe.
```

\* ===== Define variable students to count the number of records in the file.

```
compute student=1.
aggregate outfile='c:\temp\numcases.sav'
/break=student /students=n.
exe.
```

Aggregate summarizes a data file by grouping variables defined on the break subcommand. "=n" is a build-in function to count how many groups there are. This command aggregates every case individually, but exports no variables, in order to count the cases in the file

\* ===== Save data file, dropping temporary variables.

```
save outfile='c:\temp\ensearch.sav'  
/DROP=L1stblnk F1stblnk jrorll datetemp temp student .
```

\* ===== Write the formatted NSC data to fixed width file.

```
write outfile='c:\temp\data.txt' /type SSN f_name m_init l_name suffix  
bday ldate fil_2 schcode brchcode reqrtrn fil_3 fil_4.  
exe.
```

\* ===== Create the header and footer data files (separate scripts below).

\* ===== Check the Reason variable: SE is for subsequent enrollment.

\* ===== (forward in time), PA is for prior admission (backwards in time).

```
include file='c:\temp\header.sps'.
```

```
include file='c:\temp\trailer.sps'.
```

\* ===== Reread the header, data, and footer files as a long text variable.

\* ===== Then save as an SPSS variable with a common name.

\* ===== This will only work with SPSS 12.0 and above.

\* ===== Earlier versions don't support strings of more than 255 characters.

\* ===== However, the script can be rewritten to use two 250-char strings.

```
Data List File='c:\temp\header.txt' /var1 1-500 (a).
```

```
exe.
```

```
save outfile='c:\temp\header.sav' /keep=var1.
```

```
exe.
```

The information needed for the header is created in a separate script, saved as a text file, reread as a long string, then saved in SPSS format

```
Data List File='c:\temp\data.txt' /var1 1-500 (a).
```

```
exe.
```

```
save outfile='c:\temp\data.sav' /keep=var1.
```

```
exe.
```

The above process is repeated for the data and the footer, so that all of components of the submission file are now stored as long strings named var1

```
Data List File='c:\temp\trailer.txt' /var1 1-500 (a).
```

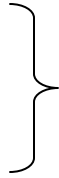
```
exe.
```

```
save outfile='c:\temp\trailer.sav' /keep=var1.
```

```
exe.
```

\* ===== Concatenate header, data, and footer files.

```
add files file='c:\temp\header.sav'  
  /file='c:\temp\data.sav'  
  /file='c:\temp\trailer.sav'.  
exe.
```



Now that all three files contain the single long string var1, we can concatenate the files

\* ===== Save the concatenated file in SPSS format as a backup.

```
save outfile='c:\temp\submissionfile.sav'.  
exe.
```

\* ===== Export output as a single fixed format flat file.

```
write outfile='c:\temp\NSC_submission.txt'  /var1.  
exe.
```

\* ===== Finally we turn messages and results back on.

Set results=on.

\* ===== End Clearinghouse Script Main =====.

**\* ===== Begin Header Script =====.**

- \* ===== This file creates the header for a Clearinghouse file.
- \* ===== The default type is SE (subsequent enrollment).
- \* ===== PA (prior attendance) is also a valid type.

Data list / rtype 1-2 (A) schcode 3-8 (A) branch 9-10 (A) reason 11-12 (A)  
entity 13 (A) school 14-53 (A).

Begin data

H100529400 IWaukesha County Technical College

end data.

```
compute reason='SE'.
compute month=xdate.month($time).
compute date=xdate.mday($time).
compute year=xdate.year($time).
execute.
formats month, date (F2.0).
formats year (F4.0).
execute.
```

This would be changed to 'PA' if a prior attendance file was needed

\$time is a system variable that stores the current date and time

\*===== Note that this has been updated since the original presentation.

\*===== to ensure that the header file submission date is zero-padded.

```
string today (A8).
string temp_$ (A2).
do if month < 10.
compute temp_$ = concat ('0', string(month,F1.0)).
else.
compute temp_$ = string(month, F2.0).
end if.
exe.
compute today=concat(string(year,F4.0),temp_$).
do if date < 10.
compute temp_$ = concat ('0', string(date,F1.0)).
else.
compute temp_$ = string (date,F2.0).
end if.
exe.
compute today = concat(ltrim(rtrim(today)),temp_$).
execute.
```

write outfile='c:\temp\header.txt' /rtype schcode branch school today  
reason entity.  
exe.

**\* ===== End Header Script =====.**

**\* ===== Begin Footer Script =====.**

get file='c:\temp\numcases.sav'.

string rtype (A2).

compute rtype='T1'.

string cases (A8).

compute cases=lpad(ltrim(string((students+2),f7.0)),8,'0').

exe.

write outfile='c:\temp\trailer.txt' /rtype cases.  
exe.

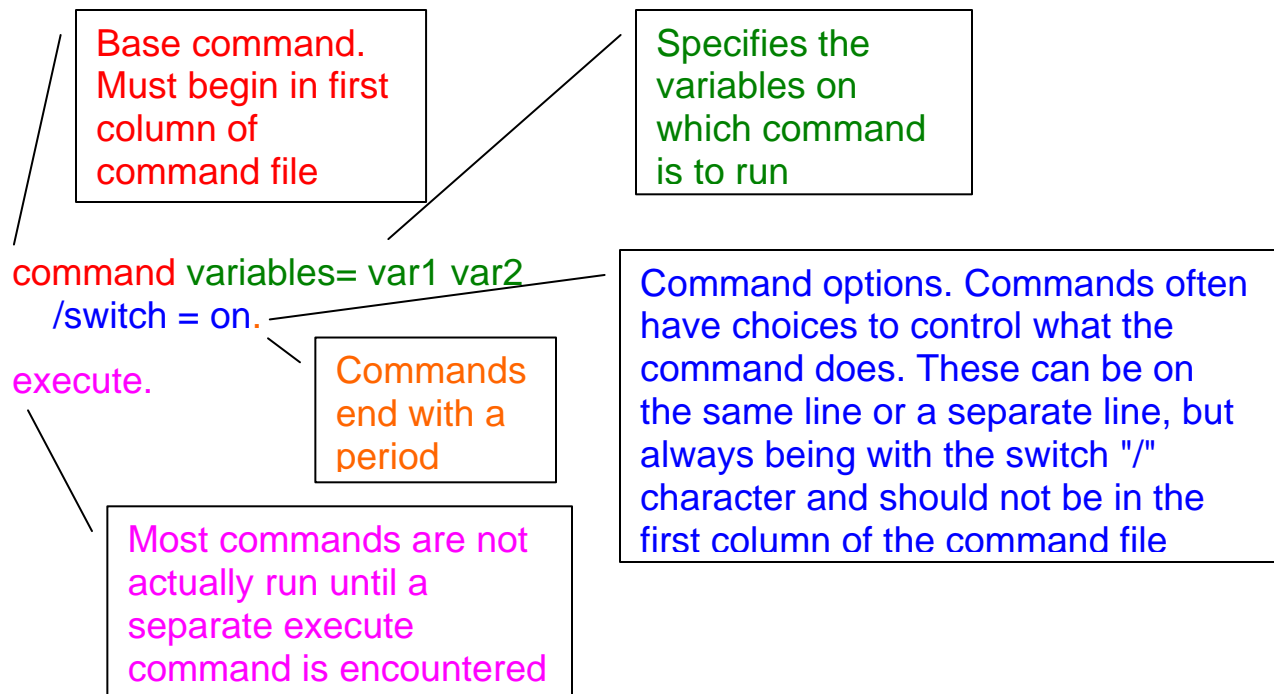
**===== End Footer Script =====.**

This is the file with the count of the number of cases created by the aggregate command in the main syntax

Two is added to the number of cases to get the record count because the header and footer are also records

## Appendix: Introduction to SPSS Syntax

Anyone who ever used SPSSx for mainframes will recognize SPSS syntax, but users who learned SPSS on a PC-based platform may have never seen syntax before. Syntax commands all follow a standard form:



### SPSS Syntax Resources

If you are unfamiliar with SPSS syntax, the best resource is the *Command Syntax Reference* that comes on the SPSS CD-ROM and is accessed from the Help menu of SPSS. Note that the command reference may not have been installed if you selected the standard or quick installation option when you initially set up SPSS; running a custom installation will ensure that you are able to install the command reference.