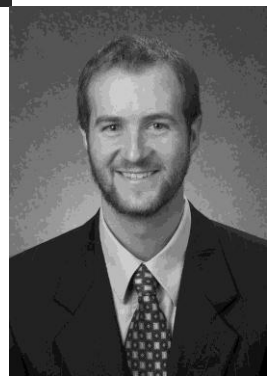
Tips, Tricks, and Techniques from the Experts





SYSTEMS SEMINAR CONSULTANTS, INC.

Presented by Systems Seminar Consultants
2997 Yarmouth Greenway Drive, Madison, WI 53711
Phone: (608) 278-9964 • Web: **www.sys-seminar.com**

Contact Us

SYSTEMS SEMINAR CONSULTANTS, INC.

SAS® Training, Consulting, & Help Desk Services
2997 Yarmouth Greenway Drive • Madison, WI 53711
(608) 278-9964 x304 • Fax (608) 278-0065
www.sys-seminar.com

sas | SAS Alliance *Silver Member*

Greg Herker
Business Developer
gherker@sys-seminar.com

Tips, Tricks, and Techniques from the Experts

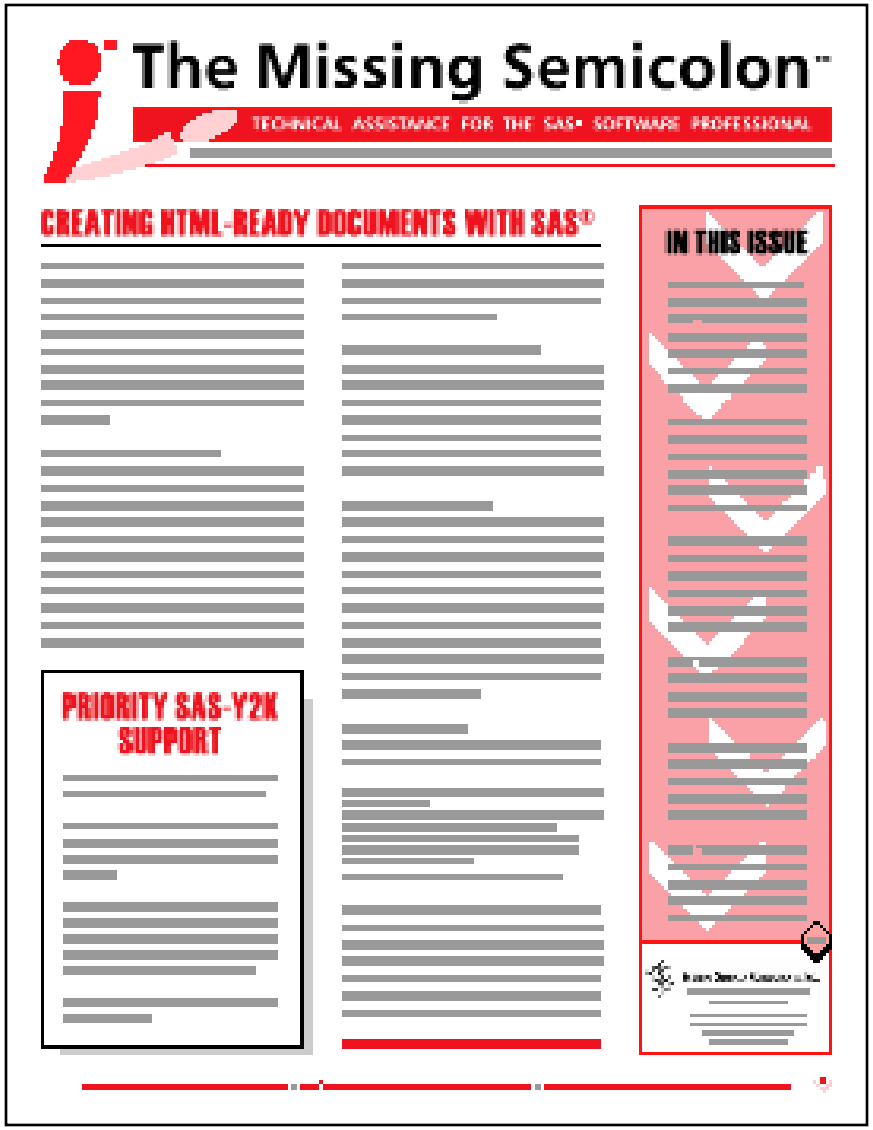
This paper was written by Systems Seminar Consultants, Inc.

SSC specializes in SAS software and offers:

- Training Services
- Consulting Services
- Help Desk Plans
- Newsletter Subscriptions to ***The Missing Semicolon™***

Tips, Tricks, and Techniques from the Experts

- **TIPS**
- TRICKS
- TECHNIQUES


The Missing Semicolon™

TECHNICAL ASSISTANCE FOR THE SAS® SOFTWARE PROFESSIONAL

CREATING HTML-READY DOCUMENTS WITH SAS®

IN THIS ISSUE

PRIORITY SAS-Y2K SUPPORT

Back issues on our website: **www.sys-seminar.com**

Tips, Tricks, and Techniques from the Experts

- General Topics
- Graphing
- DATA Step
- PROC Step
- Report Formatting

General Tips

1. Run Cancel
2. Dataset Options
3. Data Set Labels
4. Wildcards – Colon
5. Grouped – not sorted
6. Space
7. Deleting Macro Variables
8. Quoting Problems
9. Commenting
10. Graphing
11. Excel Output

General Processing: CANCEL option (Oct '00)

You can use the RUN CANCEL; statement to have a step compiled but not executed.

Example:

```
PROC PRINT DATA=SOFTSALE;
   VAR NAME DIVISION;
RUN CANCEL; /* step compiles but does not execute */
```

General Processing: Dataset Options (Jan '02)

Many dataset options can be used in PROC steps as well as DATA steps. The most common data step options are RENAME, WHERE, DROP and KEEP.

Example:

```
PROC SUMMARY DATA=customer(rename=(id=custid));
   CLASS custid;
   VAR sales;
   OUTPUT OUT=nosales(where =(totsales <= 0 )
                      rename=(_freq_=numbobs)
                      drop  =_type_)
   SUM(SALES)=TOTSALES;
RUN;
```

Data Set Processing: Label Option (Jun '98)*

Data sets have labels too! The data set label is a good place to store critical information about the dataset.

```
%Let pdate=Oct 31, 2009;
DATA EMPLOYEES
(LABEL="Employees of Systems Seminar Consultants as of &pdate");
     ...
RUN;
```

```
                    The CONTENTS Procedure

Data Set Name: WORK.EMPLOYEES           Observations:          40
Member Type:   DATA                     Variables:             6
Engine:        V8                       Indexes:               0
Created:       15:10 Tuesday, November  Observation Length:    40
Last Modified: 15:10 Tuesday, November  Deleted Observations: 0
Protection:                             Compressed:            NO
Data Set Type:                          Sorted:                NO
Label:  Employees of Systems Seminar Consultants as of Oct 31, 2009
```

Variable Processing: Colon Wildcard (Jun '98)

The colon can be used as a wildcard in a variable list. For example, BAL: refers to all variables beginning with BAL.

```
PROC PRINT DATA=EMPLOYEES;
   VAR NAME BAL: ;
RUN;
```

Resulting in:

Obs	NAME	BAL1	BALAMT	BAL20
1	Katie	400	331	45

Grouping: BY - NOTSORTED option (Feb '99)

When data is grouped by logical sections, but remains unsorted alphabetically or numerically (i.e. JANUARY, FEBRUARY, MARCH, etc.), a BY statement can still be used with the NOTSORTED option.

Example:

```
PROC PRINT DATA=MYDATA;
   BY MONTH NOTSORTED;
   VAR MONTH AMOUNT;
RUN;
```

Grouping: BY - NOTSORTED option (Feb '99)

Resulting in (Sample Output):

```
------ MONTH=JANUARY -----------------

  Obs      MONTH      AMOUNT

   1      JANUARY       300


------ MONTH=FEBRUARY ----------------

 Obs      MONTH       AMOUNT

  2      FEBRUARY       300
```

Space Issues: Drop Unneeded Datasets (Jan '00)

To clear up space in the middle of your programs, delete any unneeded data sets as soon as they are no longer needed.

Example:

```
PROC DATASETS;
   DELETE TEST;
QUIT;

     or

PROC SQL;
   DROP TABLE TEST;
QUIT;
```

Data Step: Delete User Macro Variables (Jan '02)

To delete a user macro variable from the global symbol table, you can use the %symdel macro function.

Example:

```
%symdel mlastyr;
```

To delete all global user macro variables, a pair of data steps can be used to delete all variables found:

```
data macvars;              /*build dataset with name of  mac vars*/
   set sashelp.vmacro;  /*dictionary table eith mac var names */
   where scope='GLOBAL';/*keep global macro variables only    */
run;

data _null_;               /* do not create a SAS dataset */
   set macvars;            /* read SAS dataset from above */
   call execute('%symdel '!!trim(left(name))!!';'); /*delete */
run;
```

PC SAS: Open Quote Problem (Jul '00)

Submitting code with missing quote marks (single ' or double ") in Display Manager or Windows/SAS can be frustrating to fix.

Try submitting the following to close off the mis-quoted string:

```
*'; *"; run;
```

Data Step: Comment Out Code (Oct '00)

A simple way to block a section of SAS code from being processed is to make it look like it is a piece of macro code but never invoke it:

```
%MACRO COMMENT;     /* Starts a MACRO definition */
   DATA WHATEVER;
     .........
   RUN;
   PROC PRINT DATA=WHATEVER;
     .........
   DATA WHATEVER2;
     .........
   RUN;
   PROC FREQ DATA=WHATEVER2;
     .........
%MEND COMMENT;      /* End of block to ignore */
```

Data Step: Comment Out Code (Aug '06)

If you are working in the SAS enhanced editor, here is a quick way to comment out a section of code.

To enclose code or text within a block comment,
highlight the selected code and press Ctrl+/

To remove a block comment,
highlight the selected code and press Ctrl+Shift+/

Graphing Tips

1. Descending
2. Annotate

Graphing: Descending Option (Jun '99)

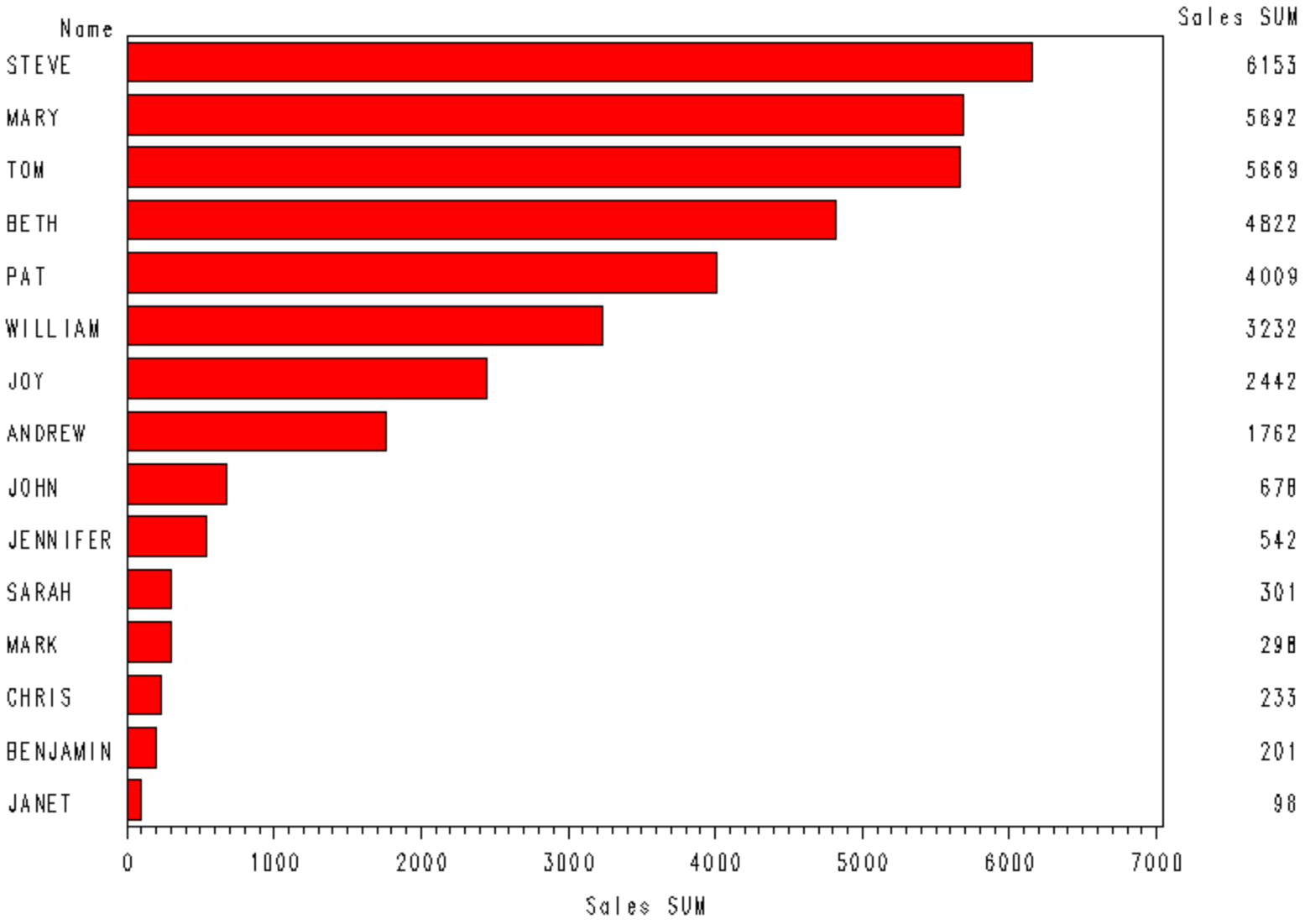
Use the descending option within the VBAR or HBAR statement to produce bar charts in descending order of magnitude.

Example:

```
PROC GCHART DATA=SASUSER.SOFTSALE;
   HBAR NAME/DESCENDING SUMVAR=SALES;
RUN;
QUIT;
```

Graphing: Descending Option (Jun '99)

Use the descending option within the VBAR or HBAR



Graphing: Annotate Datasets (Jun '99)

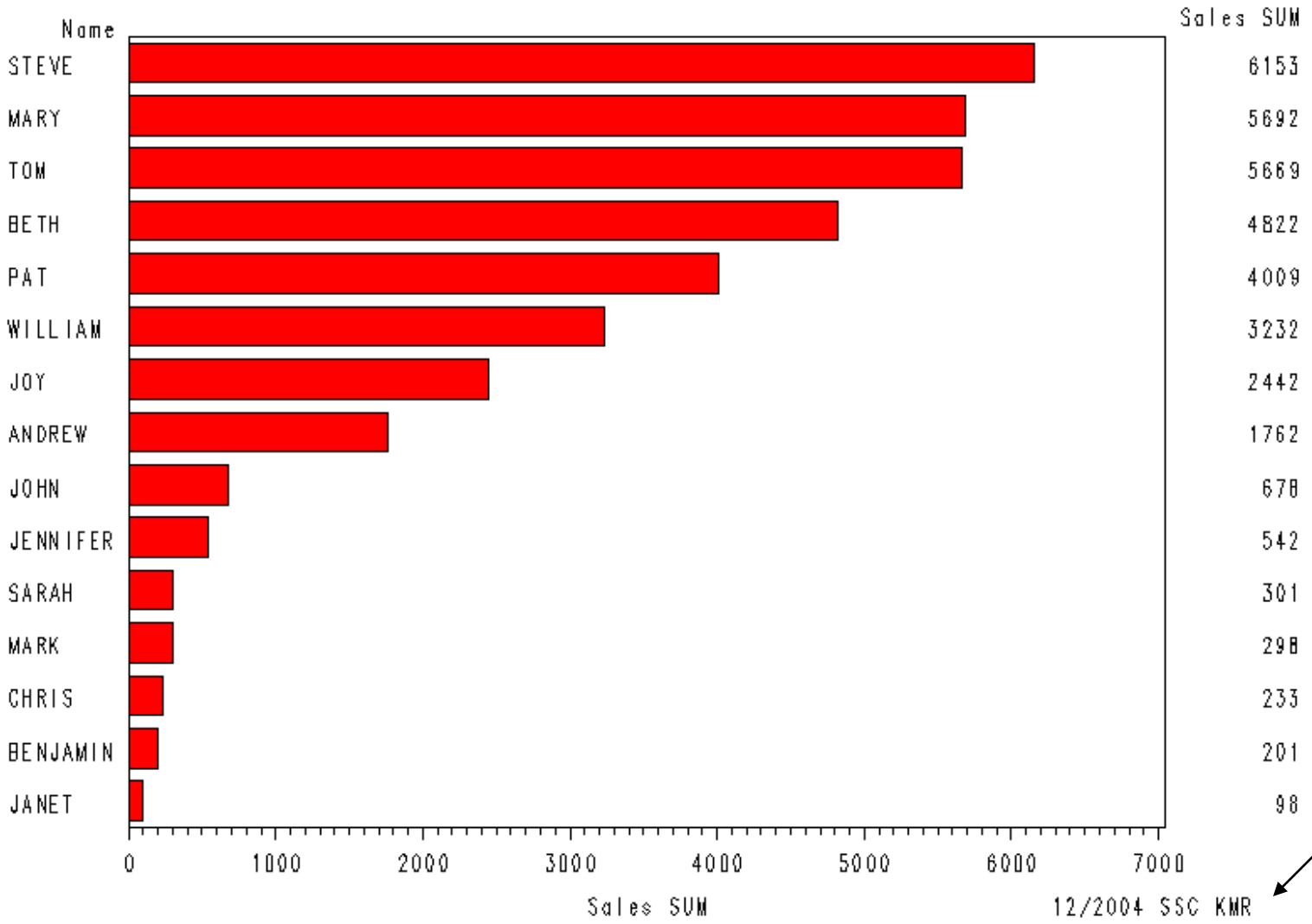
Utilize an annotate dataset to customize your graphs:

Example:

```
DATA MyAnnotate;
   FUNCTION='LABEL';
   TEXT='12/2004 SSC KMR';  /*DATE, COMPANY, INITIALS */
   SIZE=1; X=85; Y=1;
   OUTPUT;
RUN;

TITLE;
PROC GCHART DATA=SASUSER.SOFTSALE ANNOTATE=MYANNOTATE;
   HBAR NAME/DESCENDING SUMVAR=SALES;
RUN;
QUIT;
```

Graphing: Annotate Datasets (Jun '99)



ODS: Creating Excel Output (Winter '06)

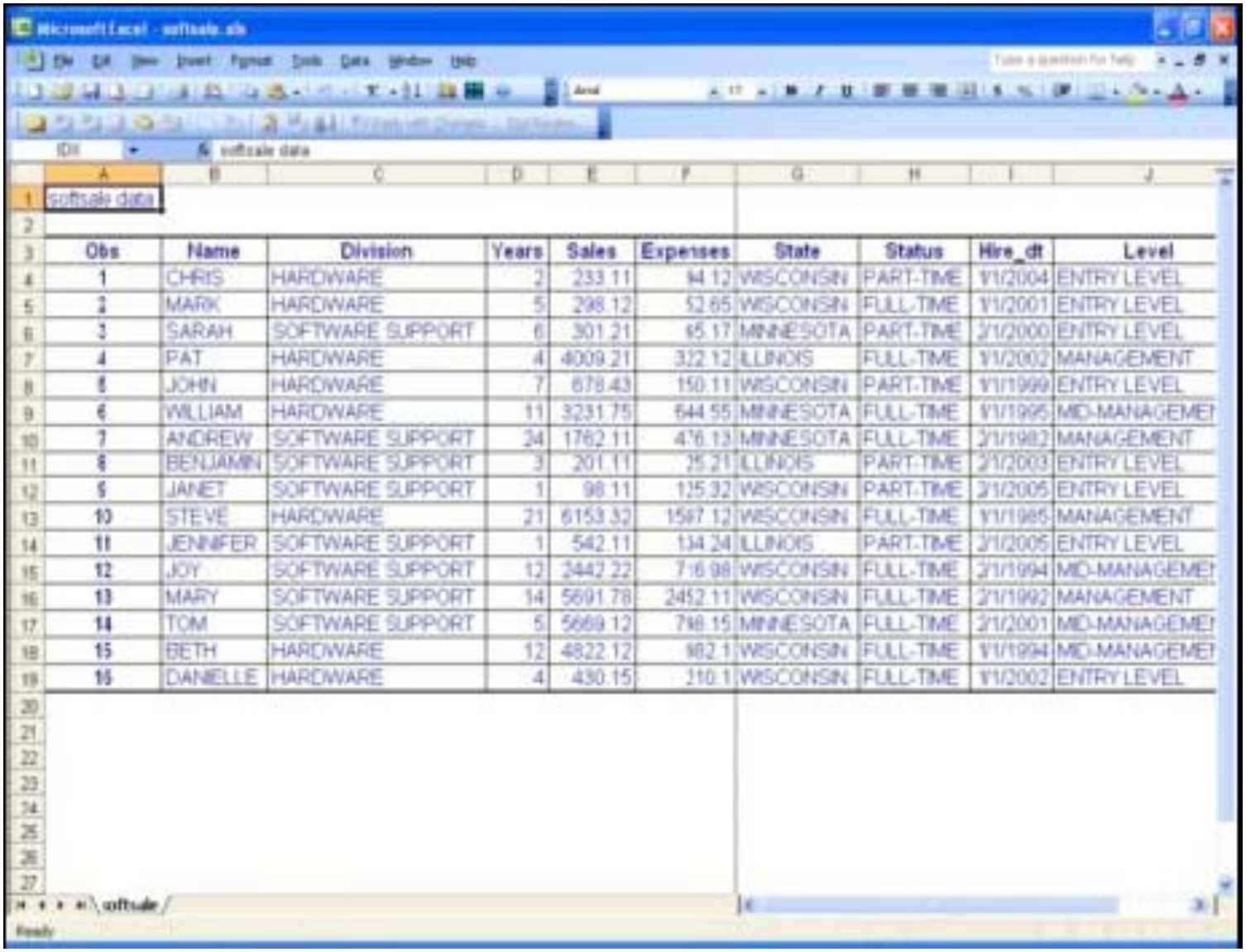
ODS makes it easy to take the reports we create in SAS and move them to Excel. With just a few lines of code, the SAS report is ready to open in Excel.

Example:

```
ODS HTML FILE ='C:\TEMP\SOFTSALE.XLS';
   PROC PRINT DATA = SOFTSALE;
      TITLE 'SOFTSALE DATA';
   RUN;
   TITLE;
ODS HTML CLOSE;
```

Double clicking on this file created with a .xls extension opens the HTML file in Excel.

ODS: Creating Excel Output (Winter '06)



softsale data

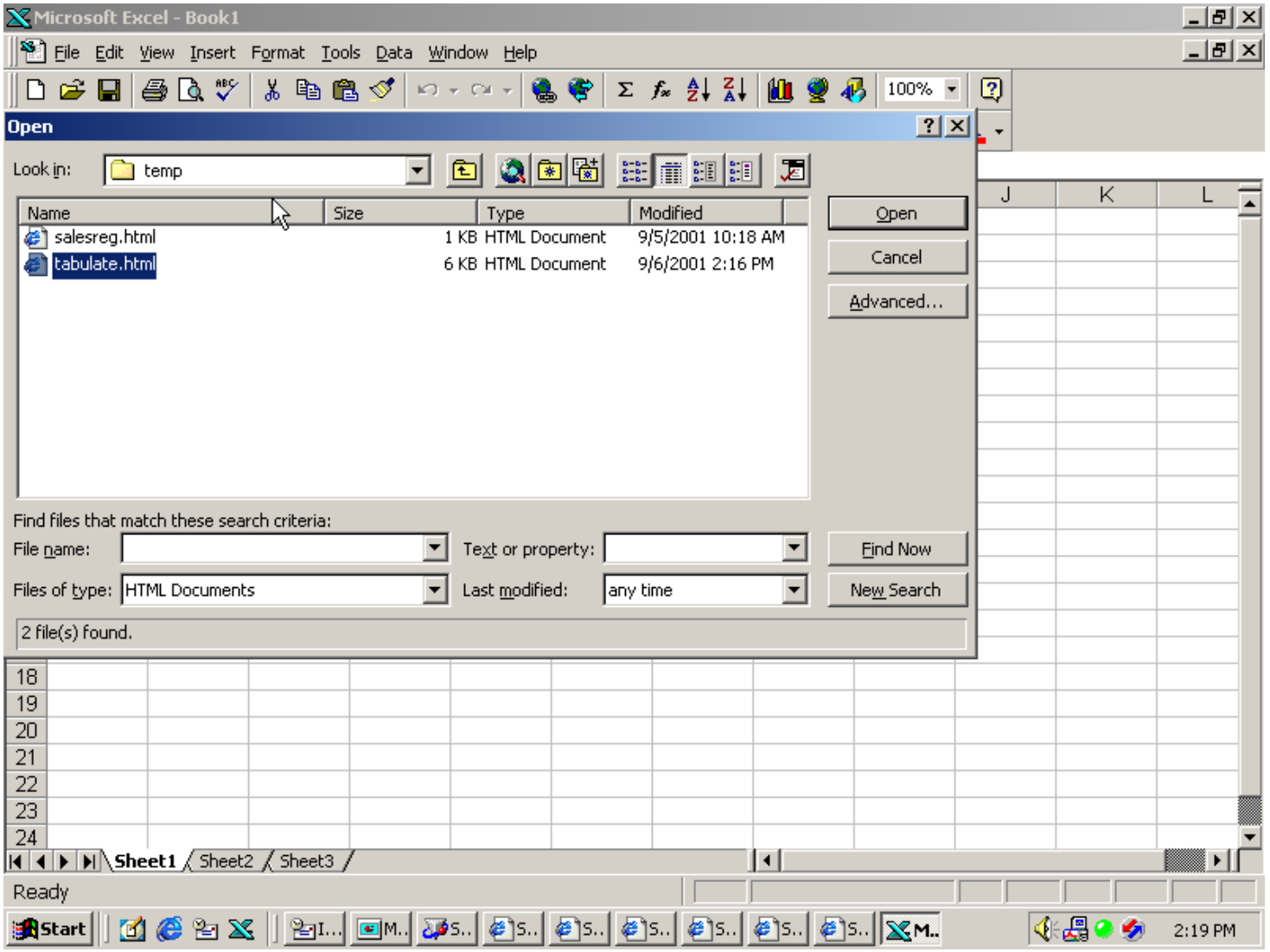
Obs	Name	Division	Years	Sales	Expenses	State	Status	Hire_dt	Level
1	CHRIS	HARDWARE	2	233.11	94.12	WISCONSIN	PART-TIME	1/1/2004	ENTRY LEVEL
2	MARK	HARDWARE	5	298.12	52.65	WISCONSIN	FULL-TIME	1/1/2001	ENTRY LEVEL
3	SARAH	SOFTWARE SUPPORT	6	301.21	65.17	MINNESOTA	PART-TIME	2/1/2000	ENTRY LEVEL
4	PAT	HARDWARE	4	4009.21	322.12	ILLINOIS	FULL-TIME	1/1/2002	MANAGEMENT
5	JOHN	HARDWARE	7	678.43	150.11	WISCONSIN	PART-TIME	1/1/1999	ENTRY LEVEL
6	WILLIAM	HARDWARE	11	3231.75	644.55	MINNESOTA	FULL-TIME	1/1/1995	MID-MANAGEMEI
7	ANDREW	SOFTWARE SUPPORT	24	1762.11	476.13	MINNESOTA	FULL-TIME	2/1/1982	MANAGEMENT
8	BENJAMIN	SOFTWARE SUPPORT	3	201.11	25.21	ILLINOIS	PART-TIME	2/1/2003	ENTRY LEVEL
9	JANET	SOFTWARE SUPPORT	1	98.11	125.32	WISCONSIN	PART-TIME	2/1/2005	ENTRY LEVEL
10	STEVE	HARDWARE	21	6153.32	1597.12	WISCONSIN	FULL-TIME	1/1/1985	MANAGEMENT
11	JENNIFER	SOFTWARE SUPPORT	1	542.11	134.24	ILLINOIS	PART-TIME	2/1/2005	ENTRY LEVEL
12	JOY	SOFTWARE SUPPORT	12	2442.22	716.98	WISCONSIN	FULL-TIME	2/1/1994	MID-MANAGEMEI
13	MARY	SOFTWARE SUPPORT	14	5691.78	2452.11	WISCONSIN	FULL-TIME	2/1/1992	MANAGEMENT
14	TOM	SOFTWARE SUPPORT	5	5669.12	798.15	MINNESOTA	FULL-TIME	2/1/2001	MID-MANAGEMEI
15	BETH	HARDWARE	12	4822.12	982.1	WISCONSIN	FULL-TIME	1/1/1994	MID-MANAGEMEI
16	DANIELLE	HARDWARE	4	430.15	210.1	WISCONSIN	FULL-TIME	1/1/2002	ENTRY LEVEL

ODS: Creating Excel Output (Winter '06)

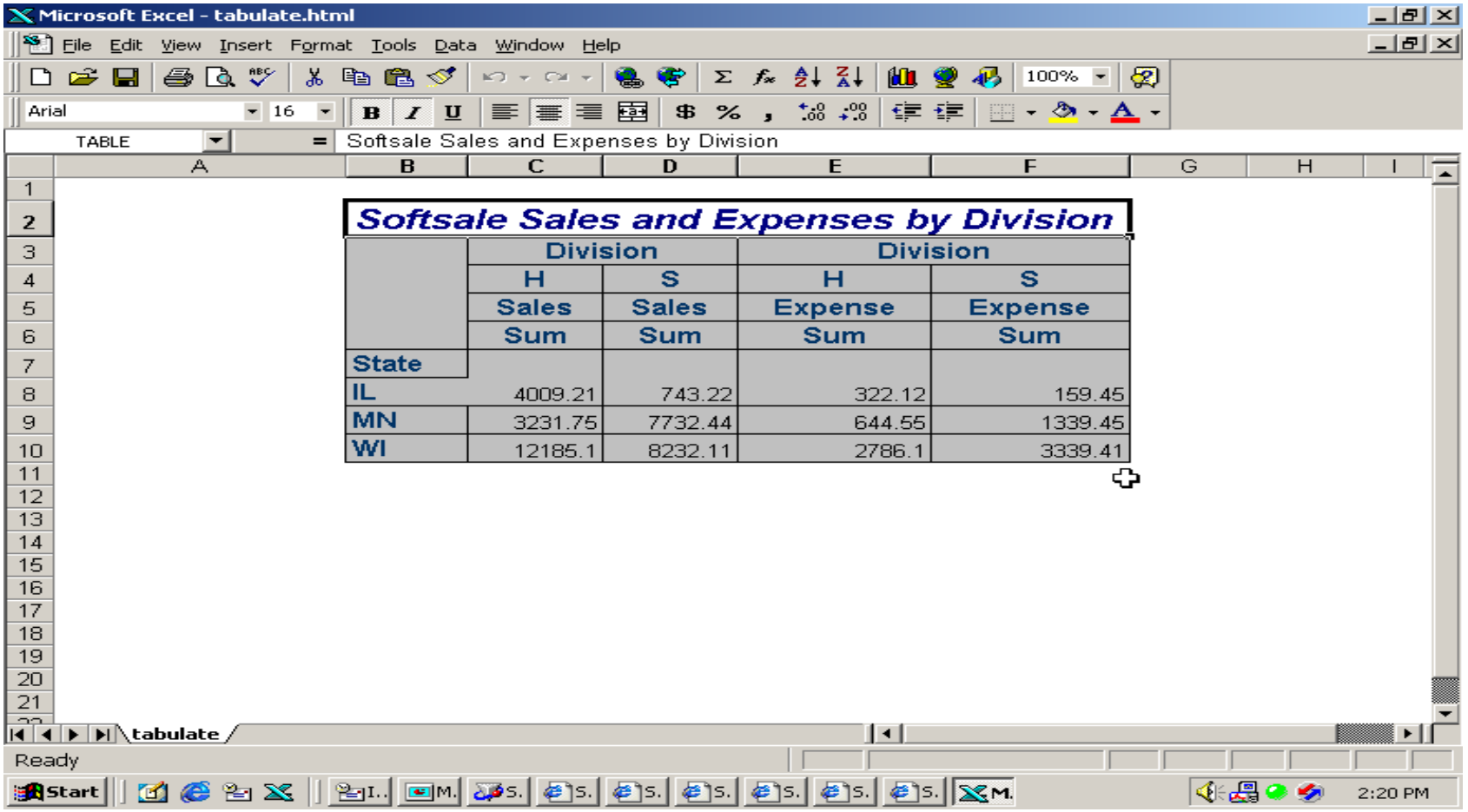
Use 'File Open' in Excel to read this file created with a .html extension.

```
ODS HTML body='c:\temp\tabulate.html';
   PROC TABULATE DATA=SOFTSALE;
      TITLE 'SOFTSALE SALES AND EXPENSES BY DIVISION';
      CLASS STATE DIVISION ;
      VAR SALES EXPENSE;
      TABLE STATE,DIVISION*SALES*SUM
                  DIVISION*EXPENSE*SUM;
   RUN;
ODS HTML CLOSE;
```

ODS: Creating Excel Output (Winter '06)



ODS: Creating Excel Output (Winter '06)



Softsale Sales and Expenses by Division

	Division		Division	
	H	S	H	S
	Sales	Sales	Expense	Expense
	Sum	Sum	Sum	Sum
State				
IL	4009.21	743.22	322.12	159.45
MN	3231.75	7732.44	644.55	1339.45
WI	12185.1	8232.11	2786.1	3339.41

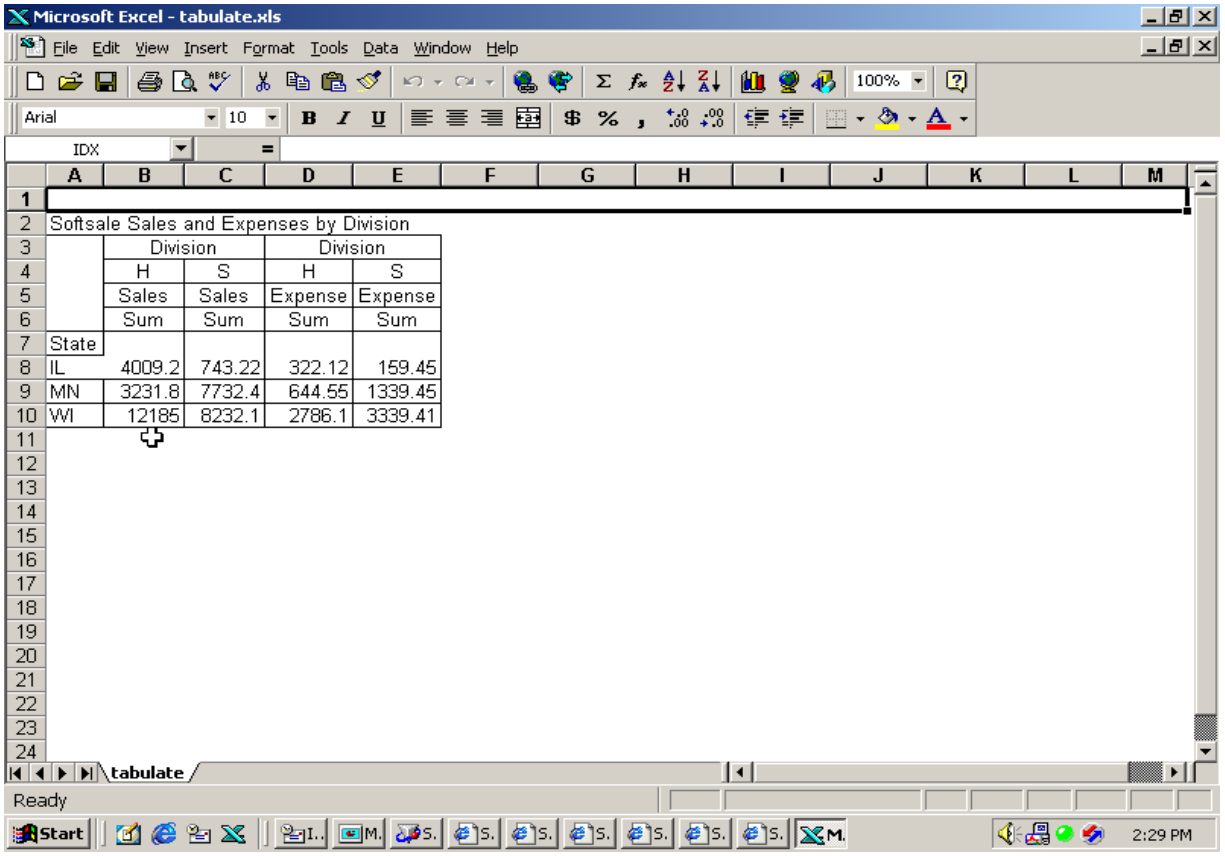
ODS: Creating Excel Output (Winter '06)

Reproducing the PROC TABULATE Output using the option STYLE=MINIMAL greatly reduces file size.

```
OPTIONS NOCENTER;
ODS HTML BODY='C:\TEMP\TABULATE.XLS' STYLE=MINIMAL;
PROC TABULATE DATA=SOFTSALE;
   TITLE 'SOFTSALE SALES AND EXPENSES BY DIVISION';
   CLASS STATE DIVISION ;
   VAR SALES EXPENSE;
   TABLE STATE,DIVISION*SALES*SUM
               DIVISION*EXPENSE*SUM;
RUN;
ODS HTML CLOSE;
```

ODS: Creating Excel Output (Winter '06)

The resulting worksheet in Excel.



You may want to do some slight formatting (see article)

Data Step Tips

1. INFILE
2. INPUT - Informats
3. ATTRIBUTE
4. LENGTH
5. WHERE ALSO
6. Errors
7. Formats
8. Functions

Data Step: INFILE - FIRSTOBS Option (Oct '99)

When reading raw files use the FIRSTOBS=record-number to begin reading the input data at the record number specified.

This is also helpful when you want to bypass a header record, usually stored in the first record of the input file.

Example:

```
INFILE RAWFILE FIRSTOBS=2;
```

Data Step: INFILE- OBS Options (Oct '99)

To read a range of records from a raw file you can use the FIRSTOBS option combined with OBS=record-number, where record-number specifies the last record that you want to read from an input file.

Example:

```
INFILE RAWFILE FIRSTOBS=10 OBS=30;
```

- Results in 21 records being read, 10 through 30.

Data Step: INPUT Informats - $UPCASEw (Spring '05)

Use the informat $UPCASEw. to read in character data which may be in upper, lower, or mixed case and store the value in the SAS data set as an upper case value.

Input file:

```
William h WI
```

Data Step:

```
INPUT @01 Name     $upcase10.
      @12 Division $upcase1.
      @14 State    $upcase2.;
```

SAS data set:

```
Name    Division State
WILLIAM    H       WI
```

Data Step: Date Informat - YYMMDD**N**8. (Apr '00)

To read a SAS date with a four-digit year, two-digit month, and two-digit day without slashes, use the YYMMDD**N**8. informat.

Example:

Input file:

```
20041120
```

Input statement:

```
INPUT Start_dt YYMMDDN8.;
```

Data Step: ATTRIBUTE statement (Feb '99)

Use the ATTRIB statement in the Data Step to associate a format, informat, label and length with a variable all at once.

Example:

```
ATTRIB ENAME LENGTH=$20 FORMAT=$20. LABEL= "EMPLOYEE NAME";
```

Data Step: LENGTH statement (Oct '00)

Numeric variables default to 8 bytes in SAS. If you have a numeric date, you can save space and safely store it in a 4 byte numeric variable (5 bytes for WINDOWS/UNIX), such as:

```
DATA _NULL_;
   LENGTH  TODAY TODAY2  4;
   TODAY=TODAY();
   TODAY2=TODAY();
   FORMAT TODAY2 DATE9.;
   PUT _ALL_;
RUN;

Resulting in:

TODAY=16391 TODAY2=16NOV2004 _ERROR_=0 _N_=1
```

Data Step: LENGTH statement (Oct '00)

The chart below shows the maximum integer values that can be stored in floating point numeric variables of the respective sizes:

Length in Bytes	Windows	MVS
2	NA	256
3	8,192	65,536
4	2,097,152	16,777,216
5	536,870,912	4,294,967,296
6	137,438,953,472	1,099,511,627,776
7	35,184,372,088,832	281,474,946,710,656
8	9,007,199,254,740,990	72,057,594,037,927,900

Data Step: WHERE ALSO operator (Apr '00)

If you have a long WHERE clause which contains an AND, you can break it into two clauses.

The ALSO operator adds requirements to your first WHERE clause.

Example:

```
WHERE SALES>=500 AND EXPENSE<=100;

WHERE SALES>=500;
WHERE ALSO EXPENSE<=100;
```

Data Step: Stopping Data Errors (Jul '00)

Check denominator to avoid errors that result from division by missing values or zero.

Possible Data Errors:

```
   AVGBAL=TOTBAL/NUMB;
RUN;
```

```
NOTE: Missing values were generated as a result of performing an
   operation on missing values.
Each place is given by: (Number of times) at (Line):(Column)

1 at 360:16
```

Better:

```
IF NUMB NOT IN (0,.) THEN AVGBAL=TOTBAL/NUMB;
ELSE AVGBAL=0;
```

Data Step: Creating Variables Using Formats(Jan '00)

A format can be used to add a variable instead of a joining multiple dataset.

Example:

Enrollment Dataset:

Obs	NAME	ENRDATE	PLAN
1	KATIE	17SEP2002	A
2	TERESA	17OCT2000	A
3	STEVE	01MAY1998	B
4	ANN	13AUG2001	B

City Dataset:

Obs	NAME	CITY
1	KATIE	MADISON
2	TERESA	MCFARLAND
3	STEVE	MONONA
4	ANN	WAUNAKEE

Data Step: Creating Variables Using Formats(Jan '00)

Create a User Defined Format from the CITY dataset:

```
DATA CITYFMT;
   SET CITY;
   RETAIN FMTNAME '$CITYFMT';
   RENAME NAME=START
          CITY=LABEL;
RUN;

PROC FORMAT CNTLIN=CITYFMT; RUN;
PROC PRINT DATA=CITYFMT;    RUN;
```

Obs	START	LABEL	FMTNAME
1	KATIE	MADISON	$CITYFMT
2	TERESA	MCFARLAND	$CITYFMT
3	STEVE	MONONA	$CITYFMT
4	ANN	WAUNAKEE	$CITYFMT

Data Step: Creating Variables Using Formats(Jan '00)

Create a User Defined Format from the data:

Obs	START	LABEL	FMTNAME
1	KATIE	MADISON	$CITYFMT
2	TERESA	MCFARLAND	$CITYFMT
3	STEVE	MONONA	$CITYFMT
4	ANN	WAUNAKEE	$CITYFMT

Equivalent to:

```
PROC FORMAT;
    VALUE $CITYFMT
        'KATIE' ='MADISON'
        'TERESA'='MCFARLAND'
        'STEVE' ='MONONA'
        'ANN'   ='WAUNAKEE';
RUN;
```

Data Step: Creating Variables Using Formats(Jan '00)

Apply the User Defined Format to the Name variable in the Enrollment dataset:

```
DATA EMP;
    SET ENROLLMENT;
    CITY=PUT(NAME,$CITYFMT.);
RUN;
PROC PRINT DATA=EMP;
RUN;
```

Obs	NAME	ENRDATE	PLAN	CITY
1	KATIE	17SEP2002	A	MADISON
2	TERESA	17OCT2000	A	MCFARLAND
3	STEVE	01MAY1998	B	MONONA
4	ANN	13AUG2001	B	WAUNAKEE

Function Tips

1. Getoption
2. Compbl
3. Compress
4. Fileexist
5. Datepart
6. Intnx
7. Ranuni
8. Mort

Data Step: COMPBL Function (Jul '00)

To take extra blanks out of a variable (character), use the COMPBL function.

Example:

```
BIGNAME="MARY JANE       SMITH"
SMALLER=COMPBL(BIGNAME);
```

Resulting in:

```
"MARY JANE SMITH"
```

Data Step: COMPRESS Function (Jan '06)

Use the COMPRESS function's third parameter to specify characters to be removed. 'A' specifies that all alpha characters, both upper and lower case be removed. The 'S' removes all spaces, the 'P' removes all punctuation.

Example: `find the zip code in the address.`

```
DATA GETZIP;
   INPUT THIRDLINE $25.;
   LENGTH ZIPCODE $5;
   ZIPCODE = COMPRESS(THIRDLINE,',.- ','A');
   ZIPCODE2= COMPRESS(THIRDLINE,'','ASP');   /* alternate */
DATALINES;
NEW ORLEANS, LA 70160
MADISON, WI 53713
ST. PAUL, MN 55191
;
RUN;
```

Remember to add a LENGTH statement so the new variable being created is as small as possible!

Data Step: FILEEXIST Function (Jan '01)

The "FILEEXIST( )" function can tell you if a file exists on the system.

It returns 0 if the file does NOT exist and 1 if it does.

Examples:

SAS Code	file exist?	X value
`X=FILEEXIST('ABC.XYZ.TEST');`	Yes	1
`X=FILEEXIST('C:\MYDATA\XYZ.DAT');`	No	0
`X=FILEEXIST('ABC.XYZ.PDS(GOOD)');`	Yes	1

Data Step: DATEPART Function (Jul '00)

To change a date-time variable to a date only variable, use the datepart function.

Example:

```
MYDATE=DATEPART(MYDATE) ;
```

Data Step: TRANWRD Function (Spring '05)

To change variable values, use the tranwrd function instead of 'if' 'then' statements.

Example:

```
IF   DEPT ='California Dept of Transportation'
THEN DEPT ='CA Dept of Transportation ';
```

Make the change with tranwrd:

```
DEPT =TRANWRD(DEPT,"California","CA");
```

Remember to use caution with this function so only the strings you intend to change are altered.

Data Step:INTNX Function–DateTime Intervals(Mar '09)

SAS provides date, time, and datetime intervals for determining a date in the future or past based on a given date – ie. give me the date 6 weeks from today.

If you need different date or time intervals than SAS's defaults, use the INTNX function with multipliers and shift indexes. This allows you to create interval terms that reflect the way you do business.

The syntax of an interval name is:

```
name<multiplier><.shift-index>
```

The *multiplier* and the *shift-index* arguments default to 1.

Data Step: INTNX function - Time Intervals (Mar '09)

Example: Find the Monday of the week for a given date.

```
DATA  _NULL_;
    JAN14TH = '14JAN09'D; /* WEDNESDAY */
    SUNDAY_DEFAULT   = INTNX( 'WEEK'    , JAN14TH, 0 );
    SUNDAY2          = INTNX( 'WEEK2'   , JAN14TH, 0 );
    MONDAY_2         = INTNX( 'WEEK.2'  , JAN14TH, 0 );
    PUT SUNDAY_DEFAULT=  WEEKDATE30.;
    PUT SUNDAY2=         WEEKDATE30.;
    PUT MONDAY_2=        WEEKDATE30.;
RUN;

LOG:
SUNDAY_DEFAULT=SUNDAY, JANUARY 11, 2009
SUNDAY2=SUNDAY, JANUARY 4, 2009
MONDAY_2=MONDAY, JANUARY 12, 2009
```

The "week2" designates a 2-week interval
The "week.2" designates the interval as week starting on the second day – Monday.
"0" indicates beginning of the interval.

More Examples - Interval Description

- DAY3 : Three-day intervals starting on Sundays
- TENDAY4.2 : Four ten-day periods starting at the second TENDAY period
- MONTH2 : January-February, March-April, May-June, July-August, September-October, November-December
- QTR3.2 : Three-month intervals starting on April 1, July 1, October 1, and January 1
- SEMIYEAR.3: Six-month intervals, March-August and September-February
- YEAR.10 : Fiscal years starting in October
- HOUR8.7 : Eight-hour intervals starting at 6 a.m., 2 p.m., and 10 p.m. (might be used for work shifts).

Data Step: RANUNI function-Random Sample(Oct '00)

To read a random sample of roughly 15% of the data from any file, use the RANUNI(0) function:

```
DATA TESTING;
   INFILE  RAWIN;
   INPUT    @;                  /* read a record and hold it */
   IF  RANUNI(0)  LT  .15;      /* allows about 15% of
                                          rows to be used  */
   INPUT    rest of program.....

RUN;
```

Data Step: MORT – Mortgage Function (Jan '02)

Use the MORT function to calculate payments on loans.

```
MORT(AMOUNT,PAYMENT,RATE,NUMBER)
```

Should you refinance your home loan? Compare two different interest rates:

```
DATA _NULL_;
   OLDPAY=MORT(100000,.,.08/12,30*12);
   NEWPAY=MORT(100000,.,.06/12,30*12);
   SAVINGS=OLDPAY-NEWPAY;
   PUT _ALL_;
   FORMAT OLDPAY NEWPAY SAVINGS DOLLAR10.2;
RUN;

LOG:
OLDPAY=$733.76 NEWPAY=$599.55 SAVINGS=$134.21 _ERROR_=0 _N_=1
```

Proc Step Tips

- Proc Summary
- Proc Import
- Proc SQL

Proc Step: PROC SUMMARY: ID statement (Jan '00)

To take along an extra value in the output dataset of a PROC SUMMARY, use the ID statement.

- Default is the maximum value
- Option for minimum value.

Example:

```
PROC SUMMARY DATA=SALES NWAY IDMIN;
   VAR SALEAMT;
   ID SALEDATE;
   CLASS CUSTNUMB;
   OUTPUT OUT=SALESUM
   N(SALEAMT)=NUMSALES
   SUM(SALEAMT)=TOTSALES;
RUN;
```

Proc Step:PROC IMPORT-Getnames Usedate(Aug '06)

When importing an Excel spreadsheet with a libname statement*, you may want SAS to use the first row of data as variable names and use the DATE9. format for date/time values. Use the Getnames and Usedate statements.

```
LIBNAME MYXLS 'C:\TEMP\SALES.XLS'; /* Point to workbook with
                                      Excel engine*/
PROC IMPORT OUT=MYDATAIN DATAFILE='MYXLS' DBMS=EXCEL;
   GETNAMES=YES;
   USEDATE=YES;
RUN;
```

GETNAMES = YES|NO

YES – use the first row of data or range as column names - default

NO – do not use the first row of data as column names.

USEDATE = YES|NO

YES – date/time values assigned DATE9. format - this is the default .

NO – date/time values assigned DATETIME. format.

(*you must have SAS/ACCESS Interface to PC Files to use the Microsoft Excel libname engine)

Proc Step: PROC IMPORT –Guessingrows (Jun '08)

Proc IMPORT is useful for converting delimited text files (CSV, TAB, and DLM) to SAS data sets. By default, SAS will only scan the first 20 rows to determine variable attributes such as field length and data type.

```
PROC IMPORT OUT=MYDATAIN DATAFILE='/MYDIR/MYDATA1.CSV' DBMS=CSV;
   GETNAMES=YES;
   DATAROW=2;
RUN;
```

If the first 20 rows are numeric data and the next rows are mixed data, the default type will be numeric. If any subsequent rows were not numeric, the results will be missing values.

If a character value has a length of 3 in the first 20 rows the default length would be 3. If any subsequent rows were longer than 3, they would be truncated to 3.

Proc Step: PROC IMPORT (Jun '08)

Example:

If we were to input a column for AnimalID and Description from the following file:

```
1005  cat
2     rat
345   dog
. . . . . through row 20 or beyond then

100A  elephant
G222  giraffe
3R5T  coyote
```

The result would be:

```
1005 cat
2    rat
345  dog
. . . . . through row 20
 .   ele
 .   gir
 .   coy
```

Proc Step: PROC IMPORT (Jun '08)

Version 9.1 introduced the GUESSINGROWS statement to Proc Import. This indicates to SAS the number of rows to scan in order to determine the variable attributes.

The value ranges from 1 to 32767.

Example:

Scan the first 12,000 records to determine the variable attributes for the CSV file:

```
PROC IMPORT OUT=MYDATAIN DATAFILE='/MYDIR/MYDATA1.CSV' DBMS=CSV;
    GETNAMES=YES;
    GUESSINGROWS=12000;
    DATAROW=2;
RUN;
```

Proc Step: Proc SQL - Counting (Jan '06)

PROC SQL can be used to produce a quick report showing all of the observations in a file without a unique key. This code rolls up the data by account number and then prints all observations where the account number was on the file more than once. This can be helpful for data cleaning.

Example:

```
PROC SQL;
   SELECT *
   FROM MEMBERS
   GROUP BY ACCT_NUM
   HAVING COUNT(*) > 1;
QUIT;
```

Proc Step: Proc SQL - Coalesce Function (Fall '05)

Need to replace a missing value on the dataset with another value? Use the COALESCE function in SQL.

This will allow you to return a default value if the field is missing, or choose the order of looking for a value from tables that have variables with the same name. The COALESCE function scans the list from left to right and returns the first non-null value.

Syntax: `COALESCE(<value expression>,...,<value expression>)`

```
PROC SQL;                    /* SELECT PAYMENTS                          */
   CREATE TABLE TEST1 AS     /* CREATE TEST1 DSN                         */
   SELECT CUSTOMERID,        /* GET CUSTOMERID                           */
      COALESCE(FRCHARGE,0)   /* GET FREIGHT CHG                          */
        AS FRCHARGE          /* AS FRCHARGE WITH DEFAULT OF ZERO*/
   FROM TESTDB.PAYMENTS;     /* FROM TABLE                               */
QUIT;
```

Proc SQL: Display Macro Variables (Jun '08)

There are times when a program has a large number of macro variables that you want to display.

You are probably familiar with the %PUT statement:

```
%PUT <TEXT|_ALL_ |_AUTOMATIC_ |_GLOBAL_|_LOCAL_ |_USER_ >;
```

The list that displays does not appear in any particular order. This makes it hard to look for a specific one.

The SAS dataset sashelp.vmacro contains all macro variables and their values. You can use this dataset like any other SAS dataset.

Select the scope of macro that you want and sort by name(variable name) to get a report in alphabetical order.

```
PROC SQL;
   SELECT NAME, VALUE
   FROM SASHELP.VMACRO
   WHERE SCOPE = 'GLOBAL'
   ORDER BY NAME;
QUIT;
```

Report Formatting Tips:

1. Formdlm
2. Pageno
3. Missing
4. Macro Processing
5. Call Symput
6. Width
7. Formats - Times

Report Formatting: Remove Page Breaks (Jun '98)

To stop SAS from issuing page breaks, set the FORMDLIM option equal to a quoted blank.

Example:

```
OPTIONS FORMDLIM=' ';

PROC PRINT DATA=SOFTSALE;
RUN;

PROC FREQ DATA=SOFTSALE;
RUN;
```

Report Formatting: Remove Page Breaks (Jun '98)

FORMDLIM=' ' removes all page breaks.

Obs	Name	Division	Years	Sales	Expense	Stat
1	CHRIS	H	2	233.11	94.12	WI
2	MARK	H	5	298.12	52.65	WI
3	SARAH	S	6	301.21	65.17	MN
4	PAT	H	4	4009.21	322.12	IL
5	JOHN	H	7	678.43	150.11	WI
6	WILLIAM	H	11	3231.75	644.55	MN
7	ANDREW	S	24	1762.11	476.13	MN

...

The FREQ Procedure

Name	Frequency	Percent	Cumulative Frequency	Cumulative Percent
ANDREW	1	6.67	1	6.67
BENJAMIN	1	6.67	2	13.33
BETH	1	6.67	3	20.00

Report Formatting: PAGENO= (Feb '99)

If your job contains many PROCs, and you want the first page of each PROC to be number 1, code the following line before each PROC:

Example:

```
OPTIONS PAGENO=1;
```

General Processing: MISSING= (Feb '99)

The MISSING= option can be used to print another character in the place of missing values.

Example:

```
OPTIONS MISSING='*';
```

Report Formatting: Macro Variables (Oct '99)

To include text on a report and avoid modifying each PROC, create a macro variable at the top of the program and refer to it in any subsequent step.

Simply modify the macro the next time you run the same job.

Example:

```
%LET MONTH=August 1999;
TITLE1  "Sales Summary as of &MONTH ";
. . . .
FOOTNOTE2 "End of &MONTH Sales Summary Report ";
```

Report Formatting: CALL SYMPUT (Jan '00)

To put the number of observations in a title, use the SET statement and the NOBS option to query the compiler.

SYMPUT can then create a macro variable usable in a title.

Example:

```
DATA _NULL_;
   CALL SYMPUT('TOTOBS',TRIM(LEFT(PUT(NMEMB,8.))));
   SET INDATA NOBS=NMEMB;
   STOP;
RUN;

TITLE "&TOTOBS OBSERVATIONS IN THE DATASET";
PROC MEANS DATA=INDATA;
RUN;
```

Report Formatting: WIDTH=MIN (Jul '00)

To condense the size of your Proc Print output, use the width=min option.

Example:

```
TITLE;
PROC PRINT DATA=SOFTSALE WIDTH=MIN;
RUN;
```

Time Processing: Time Format (Apr '00)

Use the TIME**AMPM**w. format to display times with AM or PM.

Example:

Value	Format	Result
'18:15'T	TIME5.	18:15
	TIMEAMPM8.	6:15 PM

Contact Us

SYSTEMS SEMINAR CONSULTANTS, INC.

SAS® Training, Consulting, & Help Desk Services

2997 Yarmouth Greenway Drive • Madison, WI 53711

(608) 278-9964 x304 • Fax (608) 278-0065

www.sys-seminar.com

sas | SAS Alliance *Silver Member*

Greg Herker

Business Developer

gherker@sys-seminar.com

sas | Certified Professional *Version 8*

Contact Us

SYSTEMS SEMINAR CONSULTANTS, INC.

SAS® Training, Consulting, & Help Desk Services
2997 Yarmouth Greenway Drive • Madison, WI 53711
(608) 278-9964 x308 • Fax (608) 278-0065
www.sys-seminar.com

SAS Alliance Silver Member

Rosalind Gusinow
Senior Consultant and Trainer
rgusinow@sys-seminar.com

SAS Certified Professional Version 8