Ch3. Working with Data (Data Step)

1. Creating and refining variables.

Create and redefine variables with assignment statements.

General form:

> *variable = expression*

Type of expression	Assignment statement
numeric constant	Qwerty = 10
character constant	Qwerty = 'ten'
a variable	Qwerty = OldVar
addition	Qwerty = OldVar + 10
subtraction	Qwerty = OldVar - 10
multiplication	Qwerty = OldVar * 10
division	Qwerty = OldVar / 10
exponentiation	Qwerty = OldVar ** 10

Example:

```
DATA homegarden;
       INFILE 'c:\MyRawData\Garden.dat';
       INPUT Name $ 1-7 Tomato Zucchini Peas Grapes;
       Zone = 14;
       type = 'home';
       Zucchini = Zucchini * 10;
       Total = Tomato + Zucchini + Peas + Grapes;
       PerTom = (Tomato / Total) * 100;
RUN;
PROC PRINT DATA = homegtarden;
       TITLE 'Home Gardening Survey';
RUN;
```

2. SAS Functions

General form:

> *[function-name] (argument, argument, ...)*

SAS has hundreds of functions in general areas including:

Character	Macro
Character String Matching	Mathematical
Date and Time	Probability
Descriptive Statistics	Random Number
Distance	State and ZIP Code
Financial	Variable Information

Example:

```
DATA contest;
       INFILE 'c:\MyRawData\Pumpkin.dat';
       INPUT Name $16. Age 3. +1 Type $1. +1 Date MMDDYY10.
             (Scr1 Scr2 Scr3 Scr4 Scr5) (4.1);
       AvgScore = MEAN(Scr1, Scr2, Scr3, Scr4, Scr5);
       DayEntered = DAY(Date);
       Type = UPCASE(Type);
RUN;
PROC PRINT DATA = contest;
       TITLE 'Pumpkin Carving Contest';
RUN;
```

Pumpkin Carving Contest 1

Obs	Name	Age	Type	Date[1]	Scr1	Scr2	Scr3	Scr4	Scr5	Avg Score	Day Entered
1	Alicia Grossman	13	C	17467	7.8	6.5	7.2	8.0	7.9	7.48	28
2	Matthew Lee	9	D	17469	6.5	5.9	6.8	6.0	8.1	6.66	30
3	Elizabeth Garcia	10	C	17468	8.9	7.9	8.5	9.0	8.8	8.62	29
4	Lori Newcombe	6	D	17469	6.7	5.6	4.9	5.2	6.1	5.70	30
5	Jose Martinez	7	D	17470	8.9	9.5	10.0	9.7	9.0	9.42	31
6	Brian Williams	11	C	17468	7.8	8.4	8.5	7.9	8.0	8.12	29

Selected SAS Character Functions

Function name	Syntax[1]	Definition
Character		
ANYALNUM	ANYALNUM(*arg,start*)	Returns position of first occurrence of any alphabetic character or numeral at or after optional start position
ANYALPHA	ANYALPHA(*arg,start*)	Returns position of first occurrence of any alphabetic character at or after optional start position
ANYDIGIT	ANYDIGIT(*arg,start*)	Returns position of first occurrence of any numeral at or after optional start position
ANYSPACE	ANYSPACE(*arg,start*)	Returns position of first occurrence of a white space character at or after optional start position
CAT	CAT(*arg-1,arg-2,…arg-n*)	Concatenates two or more character strings together leaving leading and trailing blanks
CATS	CATS(*arg-1,arg-2,…arg-n*)	Concatenates two or more character strings together stripping leading and trailing blanks
CATX	CATX('*separator-string*', *arg-1,arg-2,…arg-n*)	Concatenates two or more character strings together stripping leading and trailing blanks and inserting a separator string between arguments
COMPRESS	COMPRESS(*arg*, '*char*')	Removes spaces or optional characters from character string
INDEX	INDEX(*arg*, '*string*')	Returns starting position for string of characters
LEFT	LEFT(*arg*)	Left aligns a SAS character expression
LENGTH	LENGTH(*arg*)	Returns the length of an argument not counting trailing blanks (missing values have a length of 1)
PROPCASE	PROPCASE(*arg*)	Converts first character in word to uppercase and remaining characters to lowercase
SUBSTR	SUBSTR(*arg,position,n*)	Extracts a substring from an argument starting at *position* for *n* characters or until end if no *n*[2]
TRANSLATE	TRANSLATE(*source,to-1, from-1,...to-n,from-n*)	Replaces *from* characters in *source* with *to* characters (one to one replacement only—you can't replace one character with two, for example)
TRANWRD	TRANWRD(*source,from,to*)	Replaces *from* character string in *source* with *to* character string
TRIM	TRIM(*arg*)	Removes trailing blanks from character expression
UPCASE	UPCASE(*arg*)	Converts all letters in argument to uppercase

[1] *arg* is short for argument, which means a literal value, variable name, or expression.

Function name	Example	Result	Example	Result
Character				
ANYALNUM	a='123 E St, #2 '; x=ANYALNUM(a);	x=1	a='123 E St, #2 '; y=ANYALNUM(a,10);	y=12
ANYALPHA	a='123 E St, #2 '; x=ANYALPHA(a);	x=5	a='123 E St, #2 '; y=ANYALPHA(a,10);	y=0
ANYDIGIT	a='123 E St, #2 '; x=ANYDIGIT(a);	x=1	a='123 E St, #2 '; y=ANYDIGIT(a,10);	y=12
ANYSPACE	a='123 E St, #2 '; x=ANYSPACE(a);	x=4	a='123 E St, #2 '; y=ANYSPACE(a,10);	y=10
CAT	a=' cat';b='dog '; x=CAT(a,b);	x=' catdog '	a='cat ';b=' dog'; y=CAT(a,b);	y='cat dog'
CATS	a=' cat';b='dog '; x=CATS(a,b);	x='catdog'	a='cat ';b=' dog'; y=CATS(a,b);	y='catdog'
CATX	a=' cat';b='dog '; x=CATX(' ',a,b);	x='cat dog'	a=' cat';b='dog '; y=CATX('&',a,b);	y='cat&dog'
COMPRESS	a=' cat & dog'; x=COMPRESS(a);	x='cat&dog'	a=' cat & dog'; y=COMPRESS(a,'&');	y=' cat dog'
INDEX	a='123 E St, #2'; x=INDEX(a,'#');	x=11	a='123 E St, #2'; y=INDEX(a,'St');	y=7
LEFT	a=' cat'; x=LEFT(a);	x='cat '	a=' my cat'; y=LEFT(a);	y='my cat '
LENGTH	a='my cat'; x=LENGTH(a);	x=6	a=' my cat '; y=LENGTH(a);	y=7
PROPCASE	a='MyCat'; x=PROPCASE(a);	x='Mycat'	a='TIGER'; y=PROPCASE(a);	y='Tiger'
SUBSTR	a='(916)734-6281'; x=SUBSTR(a,2,3);	x='916'	y=SUBSTR('1cat',2);	y='cat'
TRANSLATE	a='6/16/99'; x=TRANSLATE (a,'-','/');	x='6-16-99'	a='my cat can'; y=TRANSLATE (a, 'r','c');	y='my rat ran'
TRANWRD	a='Main Street'; x=TRANWRD (a,'Street','St');	x='Main St'	a='my cat can'; y=TRANWRD (a,'cat','rat');	y='my rat can'
TRIM	a='my '; b='cat'; x=TRIM(a)||b;[3]	x='mycat '	a='my cat '; b='s'; y=TRIM(a)||b;	y='my cats '
UPCASE	a='MyCat'; x=UPCASE(a);	x='MYCAT'	y=UPCASE('Tiger');	y='TIGER'

[2] SUBSTR has a different function when on the left side of an equal sign.

[3] The concatenation operator || concatenates character strings.

Selected SAS Numeric Function

Function name	Syntax[1]	Definition
Numeric		
INT	INT(*arg*)	Returns the integer portion of argument
LOG	LOG(*arg*)	Natural logarithm
LOG10	LOG10(*arg*)	Logarithm to the base 10
MAX	MAX(*arg-1,arg-2,…arg-n*)	Largest non-missing value
MEAN	MEAN(*arg-1,arg-2,…arg-n*)	Arithmetic mean of non-missing values
MIN	MIN(*arg-1,arg-2,…arg-n*)	Smallest non-missing value
N	N(*arg-1,arg-2,…arg-n*)	Number of non-missing values
NMISS	NMISS(*arg-1,arg-2,…arg-n*)	Number of missing values
ROUND	ROUND(*arg, round-off-unit*)	Rounds to nearest round-off unit
SUM	SUM(*arg-1,arg-2,…arg-n*)	Sum of non-missing values
Date		
DATEJUL	DATEJUL(*julian-date*)	Converts a Julian date to a SAS date value[2]
DAY	DAY(*date*)	Returns the day of the month from a SAS date value
MDY	MDY(*month,day,year*)	Returns a SAS date value from month, day, and year values
MONTH	MONTH(*date*)	Returns the month (1–12) from a SAS date value
QTR	QTR(*date*)	Returns the yearly quarter (1–4) from a SAS date value
TODAY	TODAY()	Returns the current date as a SAS date value
WEEKDAY	WEEKDAY(date)	Returns day of week (1=Sunday) from SAS date value
YEAR	YEAR(date)	Returns year from a SAS date value
YRDIF	YRDIF(*start-date,end-date,'*ACTUAL*')*	Computes difference in years between two SAS date values using actual number of days

[1] *arg* is short for argument, which means a literal value, variable name, or expression.
[2] A SAS date value is the number of days since January 1, 1960.

Function name	Example	Result	Example	Result
Numeric				
INT	`x=INT(4.32);`	`x=4`	`y=INT(5.789);`	`y=5`
LOG	`x=LOG(1);`	`x=0.0`	`y=LOG(10);`	`y=2.30259`
LOG10	`x=LOG10(1);`	`x=0.0`	`y=LOG10(10);`	`y=1.0`
MAX	`x=MAX(9.3,8,7.5);`	`x=9.3`	`y=MAX(-3,.,5);`	`y=5`
MEAN	`x=MEAN(1,4,7,2);`	`x=3.5`	`y=MEAN(2,.,3);`	`y=2.5`
MIN	`x=MIN(9.3,8,7.5);`	`x=7.5`	`y=MIN(-3,.,5);`	`y=-3`
N	`x=N(1,.,7,2);`	`x=3`	`y=N(.,4,.,.);`	`y=1`
NMISS	`x=NMISS(1,.,7,2);`	`x=1`	`y=NMISS(.,4,.,.);`	`y=3`
ROUND	`x=ROUND(12.65);`	`x=13`	`y=ROUND(12.65,.1);`	`y=12.7`
SUM	`x=SUM(3,5,1);`	`x=9.0`	`y=SUM(4,7,.);`	`y=11`
Date				
DATEJUL	`a=60001;` `x=DATEJUL(a);`	`x=0`	`a=60365;` `y=DATEJUL(a);`	`y=364`
DAY	`a=MDY(4,18,2008);` `x=DAY(a);`	`x=18`	`a=MDY(9,3,60);` `y=DAY(a);`	`y=3`
MDY	`x=MDY(1,1,1960);`	`x=0`	`m=2; d=1; y=60;` `Date=MDY(m,d,y);`	`Date=31`
MONTH	`a=MDY(4,18,2008);` `x=MONTH(a);`	`x=4`	`a=MDY(9,3,60);` `y=MONTH(a);`	`y=9`
QTR	`a=MDY(4,18,2008);` `x=QTR(a);`	`x=2`	`a=MDY(9,3,60);` `y=QTR(a);`	`y=3`
TODAY	`x=TODAY();`	*x=today's date*	`y=TODAY()-1;`	*y=yesterday's date*
WEEKDAY	`a=MDY(4,13,2008);` `x=WEEKDAY(a);`	`x=1`	`a=MDY(4,18,2008);` `y=WEEKDAY(a);`	`y=6`
YEAR	`a=MDY(4,13,2008);` `x=YEAR(a);`	`x=2008`	`a=MDY(1,1,1960);` `y=YEAR(a);`	`y=1960`

YRDIF	`a=MDY(4,13,2000);`
	`b=MDY(4,13,2008);`
	`x=YRDIF(a,b,'ACTUAL');`
		`x=8.0`	`a=MDY(4,13,2000);`
			`b=MDY(8,13,2008);`
			`y=YRDIF(a,b,'ACTUAL');`
				`y=8.33333`

3. IF-THEN/ELSE Statements

- **General form (one action):**

> IF [condition] THEN [action];

Symbolic	Mnemonic	Meaning
=	EQ	equals
¬=,^=,or~=	NE	not equal
>	GT	greater than
<	LT	less than
>=	GE	greater than or equal
<=	LE	less than or equal
	IN	equal to one in a list ('A','B','C') or ('A' 'B' 'C') [separated by commas or spaces]

- **General form (multiple actions):**

```
IF [condition] THEN DO;
    [action];
    (action);
END;
```

- **General form (multiple conditions):**

> IF [condition] AND [condition] THEN [action];

Symbolic	Mnemonic	Meaning	Examples
&	AND	all comparisons must be true	IF condition AND condition THEN ...
| or !	OR	only one comparison must be true	IF X AND (Y OR Z) THEN IF (X AND Y) OR Z THEN ...
	NOT	highest precedence; performed before AND or OR	IF (X AND NOT Y) OR Z THEN ... IF (X AND (NOT Y)) OR Z THEN...

Example:

```
Corvette 1955 .      2 black
XJ6      1995 Jaguar 2 teal
Mustang  1966 Ford   4 red
Miata    2002 .      . silver
CRX      2001 Honda  2 black
Camaro   2000 .      4 red
```

```
DATA sportscars;
       INFILE 'c:\MyRawData\UsedCars.dat';
       INPUT Model $ Year Make $ Seats Color $;
       IF Year < 1975 THEN Status = 'classic';
       IF Model = 'Corvette' OR Model = 'Camaro' THEN Make = 'Chevy';
       IF Model = 'Miata' THEN DO;
          Make = 'Mazda';
          Seats = 2;
       END;
 RUN;
 PROC PRINT DATA = sportscars;
       TITLE "Eddy's Excellent Emporium of Used Sports Cars";
RUN;
```

```
              Eddy's Excellent Emporium of Used Sports Cars        1

          Obs    Model      Year    Make     Seats    Color     Status

           1     Corvette   1955    Chevy      2      black     classic
           2     XJ6        1995    Jaguar     2      teal
           3     Mustang    1966    Ford       4      red       classic
           4     Miata      2002    Mazda      2      silver
           5     CRX        2001    Honda      2      black
           6     Camaro     2000    Chevy      4      red
```

- **General form (multiple actions):**

```
IF [condition] THEN [action];
    ELSE IF [condition] THEN [action];
    ELSE [action];
```

Advantages:

1. it is more efficient, using less computer time; once an observation satisfies a condition, SAS skips the rest of the series.
2. ELSE logic ensures that your groups are mutually exclusive so you don't accidentally have an observation fitting into more than one group.

Example (Grouping Observations):

```
Bob     kitchen cabinet face-lift   1253.00
Shirley bathroom addition          11350.70
Silvia  paint exterior                    .
Al      backyard gazebo             3098.63
Norm    paint interior               647.77
Kathy   second floor addition      75362.93
```

```
* Group observations by cost;
DATA homeimprovements;
   INFILE 'c:\MyRawData\Home.dat';
   INPUT Owner $ 1-7 Description $ 9-33 Cost;
   IF Cost = . THEN CostGroup = 'missing';
      ELSE IF Cost < 2000 THEN CostGroup = 'low';
      ELSE IF Cost < 10000 THEN CostGroup = 'medium';
      ELSE CostGroup = 'high';
RUN;
PROC PRINT DATA = homeimprovements;
   TITLE 'Home Improvement Cost Groups';
RUN;
```

```
                    Home Improvement Cost Groups                    1

                                                                 Cost
   Obs    Owner     Description                    Cost         Group

    1     Bob       kitchen cabinet face-lift     1253.00      low
    2     Shirley   bathroom addition            11350.70      high
    3     Silvia    paint exterior                   .         missing
    4     Al        backyard gazebo              3098.63      medium
    5     Norm      paint interior                647.77      low
    6     Kathy     second floor addition       75362.93      high
```

Example (Subsetting Data):

> IF [expression] THEN DELETE;

```
A Midsummer Night's Dream 1595 comedy
Comedy of Errors          1590 comedy
Hamlet                    1600 tragedy
Macbeth                   1606 tragedy
Richard III               1594 history
Romeo and Juliet          1596 tragedy
Taming of the Shrew       1593 comedy
Tempest                   1611 romance
```

```
* Choose only comedies;
    DATA comedy;
       INFILE 'c:\MyRawData\Shakespeare.dat';
       INPUT Title $ 1-26 Year Type $;
       IF Type = 'comedy';
       * IF Type = 'tragedy' OR Type = 'romance' OR Type = 'history' THEN
DELETE;
    RUN;
    PROC PRINT DATA = comedy;
       TITLE 'Shakespearean Comedies';
    RUN;
```

```
                          Shakespearean Comedies                    1

            Obs    Title                          Year      Type

             1     A Midsummer Night's Dream      1595      comedy
             2     Comedy of Errors               1590      comedy
             3     Taming of the Shrew            1593      comedy
```

4. SELECT Statement

An alternative to a series of IF and ELSE IF statements

General form (multiple actions):

```
Select [select-expression];
    when [when-expression];
    when [when-expression];
    Otherwise [otherwise-expression];
End;
```

Continued example (Grouping Observations):

```
* Group observations by cost;
DATA homeimprovements;
    INFILE 'c:\MyRawData\Home.dat';
    INPUT Owner $ 1-7 Description $ 9-33 Cost;
    *use a select statement
    Select;
    When (Cost = .) CostGroup = 'missing';
    When (Cost < 2000) CostGroup = 'low';
    When (Cost < 10000) CostGroup = 'medium';
```

```
    Otherwise CostGroup = 'high';
END;
RUN;
PROC PRINT DATA = homeimprovements;
    TITLE 'Home Improvement Cost Groups';
RUN;
```

5. Dates

Date	SAS date value
January 1, 1959	-365
January 1, 1960	0
January 1, 1961	366
January 1, 2008	17532

- Informat:

  > INPUT BirthDate ANYDTDTE9.;

- Setting the default century for input:

  > OPTIONS YEARCUTOFF = 1950;

- SAS value for a date:

  Creates a variable named EarthDay08, which is equal to the SAS date value for April 22, 2008.

  > EarthDay08 = '22APR2008'D;

- Functions:

  > AGE = INT (YRDIF (BirthDate, TODAY(), 'ACTUAL'));
  >
  > - Returns the integer portion of the value.
  >
  > AGE = ROUND(YRDIF(BirthDate, TODAY(), 'ACTUAL'));
  >
  > - Round to the nearest year.
  >
  > If Missing(AGE) then Date=mdy(Month,15,Year)
  >
  > - Substituting a day of the month when the day value is missing.

- Date interval functions:

  > intck ('qtr', '31Dec2002'd, Today())

- Difference in quarters between December 31, 2002 and today.

intex ('month', today(), 6)

- Compute Dates 6 months after today

Informats	Definition	Width range	Default width
ANYDTDTE*w*.	Reads dates in various date forms	5–32	9
DATE*w*.	Reads dates in form: *ddmmmyy* or *ddmmmyyyy*	7–32	7
DDMMYY*w*.	Reads dates in form: *ddmmyy* or *ddmmyyyy*	6–32	6
JULIAN*w*.	Reads Julian dates in form: *yyddd* or *yyyyddd*	5–32	5
MMDDYY*w*.	Reads dates in form: *mmddyy* or *mmddyyyy*	6–32	6

Functions	Syntax	Definition
DATEJUL	DATEJUL(*julian-date*)	Converts a Julian date to a SAS date value[1]
DAY	DAY(*date*)	Returns the day of the month from a SAS date value
MDY	MDY(*month,day,year*)	Returns a SAS date value from month, day, and year values
MONTH	MONTH(*date*)	Returns the month (1–12) from a SAS date value
QTR	QTR(*date*)	Returns the yearly quarter (1–4) from a SAS date value
TODAY	TODAY()	Returns the current date as a SAS date value
WEEKDAY	WEEKDAY(date)	Returns day of week (1=Sunday) from SAS date value
YEAR	YEAR(date)	Returns year from a SAS date value
YRDIF	YRDIF(*start-date,end-date,* 'ACTUAL')	Computes difference in years between two SAS date values using actual number of days

Formats	Definition	Width range	Default width
DATE*w*.	Writes SAS date values in form: *ddmmmyy*	5–9	7
DAY*w*.	Writes the day of the month from a SAS date value	2–32	2
EURDFDD*w*.	Writes SAS date values in form: *dd.mm.yy*	2–10	8
JULIAN*w*.	Writes a Julian date from a SAS date value	5–7	5
MMDDYY*w*.	Writes SAS date values in form: *mmddyy* or *mmddyyyy*	2–10	8
WEEKDATE*w*.	Writes SAS date values in form: *day-of-week, month-name dd, yy or yyyy*	3–37	29
WORDDATE*w*.	Writes SAS date values in form: *month-name dd, yyyy*	3–32	18

[1] A SAS date value is the number of days since January 1, 1960

Informats	Input data	INPUT statement	Results
ANYDTDTE*w.*	1jan1961 01/01/61	INPUT Day ANYDTDTE10.;	366 366
DATE*w.*	1jan1961	INPUT Day DATE10.;	366
DDMMYY*w.*	01.01.61 02/01/61	INPUT Day DDMMYY8.;	366 367
JULIAN*w.*	61001	INPUT Day JULIAN7.;	366
MMDDYY*w.*	01-01-61	INPUT Day MMDDYY8.;	366

Functions	Example	Result	Example	Results
DATEJUL	a=60001; x=DATEJUL(a);	x=0	a=60365; y=DATEJUL(a);	y=364
DAY	a=MDY(4,18,99); x=DAY(a);	x=18	a=MDY(9,3,60); y=DAY(a);	y=3
MDY	x=MDY(1,1,60);	x=0	m=2; d=1; y=60; Date=MDY(m,d,y);	Date=31
MONTH	a=MDY(4,18,1999); x=MONTH(a);	x=4	a=MDY(9,3,60); y=MONTH(a);	y=9
QTR	a=MDY(4,18,99); x=QTR(a);	x=2	a=MDY(9,3,60); y=QTR(a);	y=3
TODAY	x=TODAY(); a=MDY(4,13,2008);	x=today's date	x=TODAY()-1; a=MDY(4,18,2008);	x=yesterday's date
WEEKDAY	x=WEEKDAY(a); a=MDY(4,13,2008);	x=1	y=WEEKDAY(a); a=MDY(1,1,1960);	y=6
YEAR	x=YEAR(a); a=MDY(4,13,2000);	x=2008	y=YEAR(a); a=MDY(4,13,2000);	y=1960
YRDIF	b=MDY(4,13,2008); x=YRDIF(a,b,'ACTUAL');	x=8.0	b=MDY(8,13,2008); y=YRDIF(a,b,'ACTUAL');	y=8.33333

Formats	Input data	PUT statement[2]	Results
DATE*w.*	8966	PUT Birth DATE7.; PUT Birth DATE9.;	19JUL84 19JUL1984
DAY*w.*	8966	PUT Birth DAY2.; PUT Birth DAY7.;	19 19
EURDFDD*w.*	8966	PUT Birth EURDFDD8. PUT Birth EURDFDD10.;	19.07.84 19.07.1984
JULIAN*w.*	8966	PUT Birth JULIAN5.; PUT Birth JULIAN7.;	84201 1984201
MMDDYY*w.*	8966	PUT Birth MMDDYY8.; PUT Birth MMDDYY6.;	07/19/84 071984
WEEKDATE*w.*	8966	PUT Birth WEEKDATE15.; PUT Birth WEEKDATE29.;	Thu, Jul 19, 84 Thursday, July 19, 1984
WORDDATE*w.*	8966	PUT Birth WORDDATE12.; PUT Birth WORDDATE18.;	Jul 19, 1984 July 19, 1984

[2] Formats can be used in PUT statements and PUT functions in DATA steps, and in FORMAT statements in either DATA or PROC steps.

Example:

```
A. Jones     1-1-60      9-15-06     29Feb08
M. Rincon    05/10/1949  29feb2008   23MAY2008
Z. Grandage  03181988    31 10 2007  31may08
K. Kaminaka  052903      2008024     12JUN08
```

```
DATA librarycards;
    INFILE 'c:\MyRawData\Library.dat' TRUNCOVER;
    INPUT Name $11. + 1 BirthDate MMDDYY10. +1 IssueDate ANYDTDTE10.
          DueDate DATE11.;
    DaysOverDue = TODAY() - DueDate;
    Age = INT(YRDIF(BirthDate, TODAY(), 'ACTUAL'));
    IF IssueDate > '01JAN2008'D THEN NewCard = 'yes';
RUN;
PROC PRINT DATA = librarycards;
    FORMAT Issuedate MMDDYY8. DueDate WEEKDATE17.;
    TITLE 'SAS Dates without and with Formats';
RUN;
```

SAS Dates without and with Formats 1

Obs	Name	Birth Date	Issue Date	DueDate	Days OverDue	Age	New Card
1	A. Jones	0	09/15/06	Fri, Feb 29, 2008	84	48	
2	M. Rincon	-3888	02/29/08	Fri, May 23, 2008	0	59	yes
3	Z. Grandage	10304	10/31/07	Sat, May 31, 2008	-8	20	
4	K. Kaminaka	15854	01/24/08	Thu, Jun 12, 2008	-20	4	yes

6. Retain and Sum Statements

Retain Statement

Preserve avariable's value from the previous iteration of the DATA step.

General form:

> RETAIN [variable-list] (initial-value);

Sum Statement

Retains values from the previous iteration, and cumulatively add the value of an expression to a variable.

General form:

> SUM([variable], [expression]);

- **Retain + Sum**

> **General form:**

```
RETAIN [variable] 0;
[variable] = SUM([variable], [expression]);
```

The variable must be numeric and has the initial value of zero.

Example:

```
Data MonthSales;
    Input month  sales @@;
    retain SumSales 0;
    SumSales = sum(SumSales, SumSales + sales);
    Datalines;
    1 4000 2 5000 3 . 4 5500 5 500 6 6000 7 6500 8 4500
    9 5100 10 5700 11 6500 12 7500
;
run;
proc print data=MonthSales;
run;
```

Example:

```
6-19 Columbia Peaches      8  3
6-20 Columbia Peaches     10  5
6-23 Plains Peanuts        3  4
6-24 Plains Peanuts        7  2
6-25 Plains Peanuts       12  8
6-30 Gilroy Garlics        4  4
7-1  Gilroy Garlics        9  4
7-4  Sacramento Tomatoes  15  9
7-4  Sacramento Tomatoes  10 10
7-5  Sacramento Tomatoes   2  3
```

```
* Using RETAIN and sum statements to find most runs and total runs;
    DATA gamestats;
       INFILE 'c:\MyRawData\Games.dat';
       INPUT Month 1 Day 3-4 Team $ 6-25 Hits 27-28 Runs 30-31;
       RETAIN MaxRuns;
       MaxRuns = MAX(MaxRuns, Runs);
       RunsToDate + Runs;
    RUN;
    PROC PRINT DATA = gamestats;
       TITLE "Season's Record to Date";
    RUN;
```

Season's Record to Date 1

Obs	Month	Day	Team	Hits	Runs	Max Runs	Runs ToDate
1	6	19	Columbia Peaches	8	3	3	3
2	6	20	Columbia Peaches	10	5	5	8
3	6	23	Plains Peanuts	3	4	5	12
4	6	24	Plains Peanuts	7	2	5	14
5	6	25	Plains Peanuts	12	8	8	22
6	6	30	Gilroy Garlics	4	4	8	26
7	7	1	Gilroy Garlics	9	4	8	30
8	7	4	Sacramento Tomatoes	15	9	9	39
9	7	4	Sacramento Tomatoes	10	10	10	49
10	7	5	Sacramento Tomatoes	2	3	10	52

7. Iterative Processing: Looping

General form1:

```
DO [index-variable] = [start #] to [stop #] by [increment];
    [action];
END;
```

default increment is 1

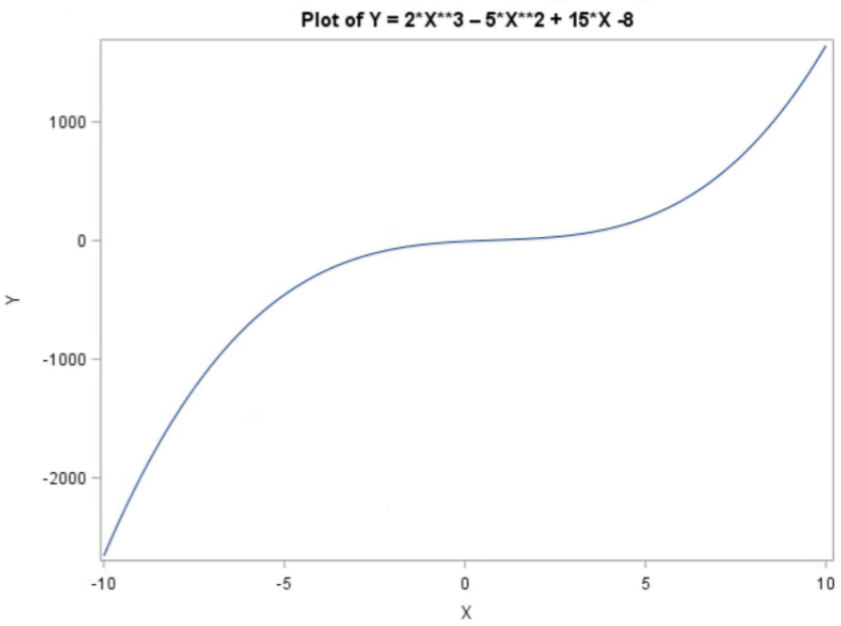
Example1

```
data Compound;
    Interest = .0375;
    Total = 100;
    do Year = 1 to 3;
        Total + Interest*Total;
        output;
    end;
    format Total dollar10.2;
run;
proc print data=Compound noobs;
run;
```

Interest	Total	Year
0.0375	$103.75	1
0.0375	$107.64	2
0.0375	$111.68	3

Example2: graph an equation

```
data Equation;
    do X = -10 to 10 by .01;
        Y = 2*X**3 - 5*X**2 + 15*X - 8;
        output;
    end;
run;
proc sgplot;
    series x=X y=Y;
run;
```



Example3: character values

```
data Easyway;
    do Group = 'Placebo','Active';
        do Subj = 1 to 5;
            input Score @;
            output;
        end;
    end;
datalines;
250 220 230 210 199
166 183 123 129 234
;
proc print data=Easyway noobs;
run;
```

Group	Subj	Score
Placebo	1	250
Placebo	2	222
Placebo	3	230
Placebo	4	210
Placebo	5	199
Active	1	166
Active	2	183
Active	3	123
Active	4	129
Active	5	234

General form2 (DO UNTIL):

```
DO [index-variable] = [start #] to [stop #] by [increment] until
([codition]);
    action;
END;
```

Example:

```
data Continue_on;
    Interest = .0375;
    Total = 100;
    do Year = 1 to 100 until (Total ge 200);
        Total + Interest*Total;
        if Total le 150 then continue;
        output;
    end;
    format Total dollar10.2;
run;
proc print data=Continue_on noobs;
run;
```

Equivalent to (output Year 1 to 19):

```
* Do loop
do until (Total ge 200);
        Year + 1
        Total + Interest*Total;
        if Total le 150 then continue;
        output;
end;
```

> `continue` halts further statements within the loop from executing and continues iteration in the Loop.

Interest	Total	Year
0.0375	$155.55	12
0.0375	$161.38	13
0.0375	$167.43	14
0.0375	$173.71	15
0.0375	$180.22	16
0.0375	$186.98	17
0.0375	$193.99	18
0.0375	$201.27	19

```
*without until (condition)
data Continue_on;
    Interest = .0375;
    Total = 100;
    do Year = 1 to 100;
        Total + Interest*Total;
        output;
        if Total ge 200 then leave;
    end;
    format Total dollar10.2;
run;
proc print data=Continue_on noobs;
run;
```

> `LEAVE` inside a DO Loop shifts control to the statement following the END statement. At the Bottom of the Loop.

General form3 (DO WHILE):

```
DO [index-variable] = [start #] to [stop #] by [increment] while
([codition]);
    action;
END;
```

Equivalent example:

```
* Do loop
do until (Total less 200);
        Year + 1
        Total + Interest*Total;
        if Total le 150 then continue;
        output;
end;
```

8. Subsetting a Data Set

(1) WHERE

General forms:.

```
WHERE condition;
```

> WHERE statement can be used in a DATA step or in a PROC step.

Symbolic	Mnemonic	Example
=	EQ	WHERE Region = 'Spain';
¬=, ~=, ^=	NE	WHERE Region ~= 'Spain';
>	GT	WHERE Rainfall > 20;
<	LT	WHERE Rainfall < AvgRain;
>=	GE	WHERE Rainfall >= AvgRain + 5;
<=	LE	WHERE Rainfall <= AvgRain / 1.25;
&	AND	WHERE Rainfall > 20 AND Temp < 90;
|,!	OR	WHERE Rainfall > 20 OR Temp < 90;
	IS NOT MISSING	WHERE Region IS NOT MISSING;
	BETWEEN AND	WHERE Region BETWEEN 'Plain' AND 'Spain';
	CONTAINS	WHERE Region CONTAINS 'ain';
	IN (list)	WHERE Region IN ('Rain', 'Spain', 'Plain');

- **KEEP/DROP**

General forms:

> ```
> KEEP = variable-list
> ```
>
> *tells SAS which variables to keep*
>
> ```
> DROP = variable-list
> ```
>
> *tells SAS which variables to drop*
>
> KEEP and DROP options can be used in a DATA step or in a PROC step.

Example:

```
data Females;
    set Learn.Survey (keep = ID Gender Age Ques1-Ques5);
    where Gender = 'F';
run;
```

(2) IF

Example:

```
data Males Females;
    set Learn.Survey (drop = ID);
    if Gender = 'F' then output Females;
    else if Gender = 'M' then output Males;
run;
```

9. Merging Data Sets

(1) data sets contain the same variables

Example:

```
data One_Two;
    set One Two;
run;
```

Data set One

ID	Name	Weight
7	Adams	210
1	Smith	190
2	Schneider	110
4	Gregory	90

Data set Two

ID	Name	Weight
9	Shea	120
3	O'Brien	180
5	Bessler	207

Data set One_Two

Obs	ID	Name	Weight
1	7	Adams	210
2	1	Smith	190
3	2	Schneider	110
4	4	Gregory	90
5	9	Shea	120
6	3	O'Brien	180
7	5	Bessler	207

(2) data sets not contain the same variables

- **Simple Merge**

Example:

```
proc sort data=Employee;
    by ID;
proc sort data=Hours;
    by ID;
data Combine;
    merge Employee Hours;
    by ID
run;
```

Listing of Data Set Hours

ID	JobClass	Hours
1	A	39
4	B	44
9	B	57
5	A	35

Listing of Data Set Employee

ID	Name
7	Adams
1	Smith
2	Schneider
4	Gregory
5	Washington

```
beluga   whale 15   dwarf     shark .5    sperm    whale 60
basking  shark 30   humpback    .   50    whale    shark 40
gray     whale 50   blue      whale 100   killer   whale 30
mako     shark 12   whale     shark 40
```

- **Control Obs. in a Merged Data Set**
  - IN

> IN = option

illustrate:

```
data NEW;
    merge Employee (in = In_Employee) /* new var., if in Employee, 1 else
0 */
          Hours (in = In_Hours) /* new variable, if in Hours, 1 else 0 */
    by ID;
    file print;
    put ID= In_Employee= In_Hours= Name= JobClass= Hours=;
    /* PUT lists the values of these variables */
run;
```

Listing of Data Set Combine

```
ID=1 In_Employee=1 In_Hours=1 Name=Smith JobClass=A Hours=39
ID=2 In_Employee=1 In_Hours=0 Name=Schneider JobClass=  Hours=.
ID=4 In_Employee=1 In_Hours=1 Name=Gregory JobClass=B Hours=44
ID=5 In_Employee=1 In_Hours=1 Name=Washington JobClass=A Hours=35
ID=7 In_Employee=1 In_Hours=0 Name=Adams JobClass=  Hours=.
ID=9 In_Employee=0 In_Hours=1 Name=  JobClass=B Hours=57
```

Example:

```
data In_both Missing_Name;
    Missing_Hours (drop = NAME); /* drop NAME in the date set
Missing_Hours */
    merge Employee (in = In_Employee) /* new var., if in Employee, 1 else
0 */
          Hours (in = In_Hours) /* new variable, if in Hours, 1 else 0 */
    by ID;
    if In_Employee & In_Hours then output In_both;
    else if NOT In_Employee & In_Hours then output Missing_Hours;
run;
```

Listing of Data Set Missing_Hours

ID	JobClass	Hours
9	B	57

Listing of Data Set In_Both

ID	Name	JobClass	Hours
1	Smith	A	39
4	Gregory	B	44
5	Washington	A	35

- RENAME

Rename = (old = new)

Example:

```
data Combine;
    merge A
          B (rename = (EmpNo = ID));
    by ID
run;
```

(3) Different By Variable Data Type

Example:

Listing of Data Set Division2

SS	JobCode	Salary
111-22-3333	A10	45123
123-45-6789	B5	35400
987-65-4321	A20	87900

Listing of Data Set Division1

SS	DOB	Gender
111223333	11/14/1956	M
123456789	05/17/1946	F
987654321	04/01/1977	F

```
data Division1C;
    set Division1(rename=(SS = NumSS));
    SS = put(NumSS,ssn11.);
    drop NumSS;
run;
data Combine;
/* sorted */
    merge Division1C Division2;
    by SS;
run;
```

Alternative:

```
data Division2N;
    set Division2(rename=(SS = CharSS));
    SS = input(compress(CharSS,'kd'),9.);
    /* Alternative:
       SS = input(CharSS,comma11.); */
    drop CharSS;
run;
data Combine;
/* sorted */
    merge Division1 Division2N;
    by SS;
run;
```

(4) Updating a Master File from a Transaction file

```
proc sort data=Price;
    by ItemCode;
run;
proc sort data=New15Dec2017;
    by ItemCode;
run;
data Price_15Dec2017;
    update Price New15Dec2017;
    by ItemCode;
run;
```

Listing of Data Set New15Dec2017

ItemCode	Price
175	25.11
204	17.87
208	.

Listing of Data Set Prices

ItemCode	Description	Price
150	50 foot hose	19.95
175	75 foot hose	29.95
200	greeting card	1.99
204	25 lb. grass seed	18.88
208	40 lb. fertilizer	17.98

Listing of Data Set Prices_15Dec2017

ItemCode	Description	Price
150	50 foot hose	19.95
175	75 foot hose	25.11
200	greeting card	1.99
204	25 lb. grass seed	17.87
208	40 lb. fertilizer	17.98