CLASS I

Goal:

1) Pharma. Industry intro
2) Work through all the basic and common use SAS commands.

PART I: Working as SAS Programmer

Industry:

1) Business: Credit card, Insurance, and Mkt.

Required Skills:

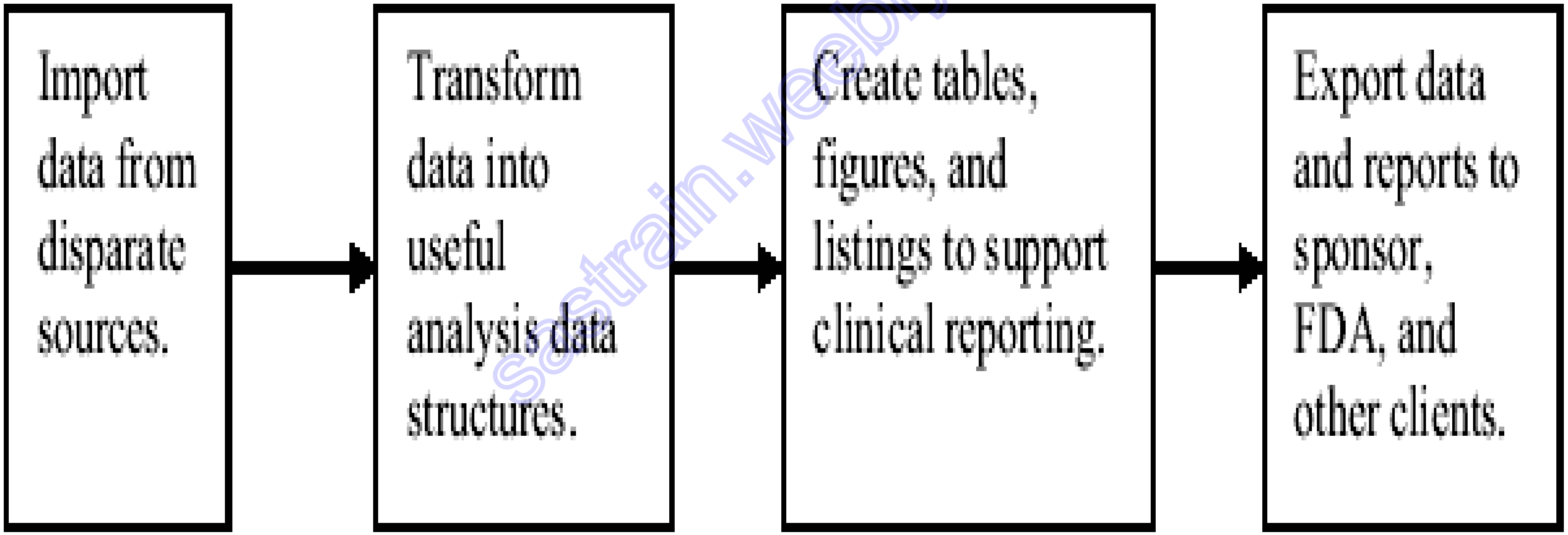
- -Basic SAS
- -Financial common sense
- -Entry level statistics background.

2) Pharma.: Clinical Research Org. ; Pharma.

Required Skills:

- -Adv. SAS skills
- -Experienced SQL skills
- -Experienced statistics background
- -UNIX commends
- -Fast learning ability

Illustration of the general work processes Of the SAS programmer:


Import data from disparate sources.
Transform data into useful analysis data structures.
Create tables, figures, and listings to support clinical reporting.
Export data and reports to sponsor, FDA, and other clients.

New Drug Dev. Process
-FDA makes sure both Safety and Efficacy

1) Pre-clinical studies: On Animal only. If the results is ok then file **Investigational New Drug(IND)**
2) Phase 1 trials-1st time on human: On healthy volunteers. If its safety then move to next level.
3) Phase 2: aimed at target populations: 100 – 200 patients. To explore efficacy of drug; to narrow dose range.
4) Phase 3: the largest- scale populations:1000 and most critical trial to prove new drug both safety and efficacy. If successful, then file **New Drug Application(NDA)**
5) Phase 4: Post-marketing trial to monitor the long term safety of new drug after its already on MKT.

Clinical Trial Study Designs

-Randomization: real drug VS placebo to reduce treatment bias.

-Blinding:

-Single-blind: Only patients don’t know.

-Double-blind: Patients + Doctor

-Triple-blind: Patients + Doctor + analysts

-Multi-center trials: to reduce site-specific bias.

-Equivalence trial: to show no clinical difference btw. new drug and existing ones.

-Superiority trial: to show one is significantly better.

-Parallel trial: patients stick with their assig. Trt.

-Crossover trial: patients switch or change therapy during the trial.

Industry Regulations and Standards

Clinical Data Interchange Standards Consortium (CDISC)

The Clinical Data Interchange Standards Consortium (CDISC) is a non-profit group that defines clinical data standards for the pharmaceutical industry.

Four important models:

- Study Data Tabulation Model (**SDTM**): cover later
- Analysis Dataset Models (**ADaM**): cover later
- Operational Data Model (**ODM**). : no need to know.
- Case Report Tabulation Data Definition Specification (**Define.xml**). To replace current **define.pdf** in submission. –cover later.

Things you need to read at work

Understand the Clinical Study

- *-protocol.*
- -statistical analysis plan *(SAP)*
- *-annotated CRF*

When programming

- *-look at specifications(specs)*
- *-look for sample codes*

Getting Started Using SAS Software

The SAS Language

- Every SAS statement **ends with a semicolon**.
- SAS statements **can be in upper- or lowercase.**
- Statements can continue on the next line (as long as you don’t split words in two).
- Statements can be on the same line as other statements.
- Statements can start in any column.

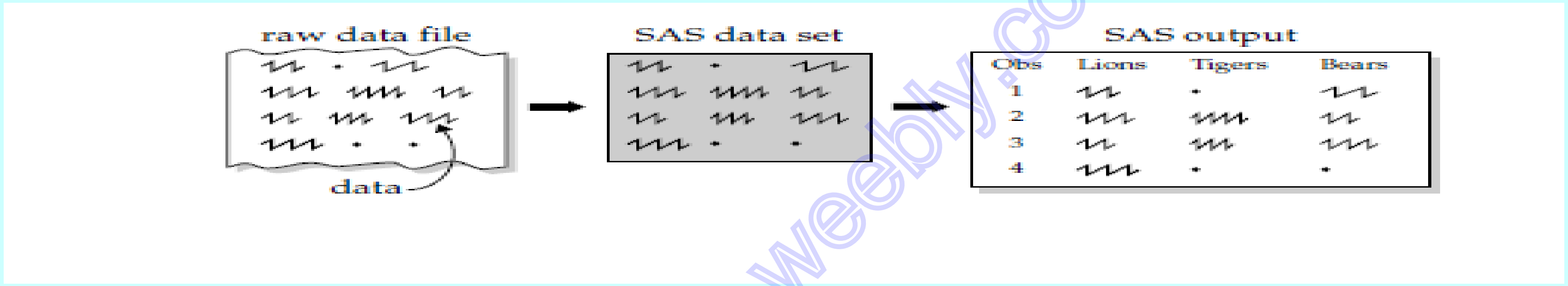
There are two styles of comments you can use: one starts with an asterisk (*) and ends with a semicolon (;). The other style starts with a slash asterisk (/*) and ends with an asterisk slash (*/).

Errors : SAS errors often come up in bright red letters,

SAS WORKING Enviroment

- 1) SAS windows
  - -could direct copy & drag
  - - use mouse to run part code
- 2) SAS UNIX
  - - Only very big pharma. corps have it
  - - Has built in macros and system support.
  - - Ugly, not easy to use but runs very fast

Program flow



- Observation by observation
- Line by line
- Looping: If.. Then ; do … end ; while..
- Could by interrupted by Macro programs.

SAS variable type:

- Numerical variable: R aligned & can be + - * /

  X = 9 ;

  Use SUM(x,y) not use +

  When / make sure **not in (. , 0) ;**

- Character var: L aligned & scan() , substr() length()

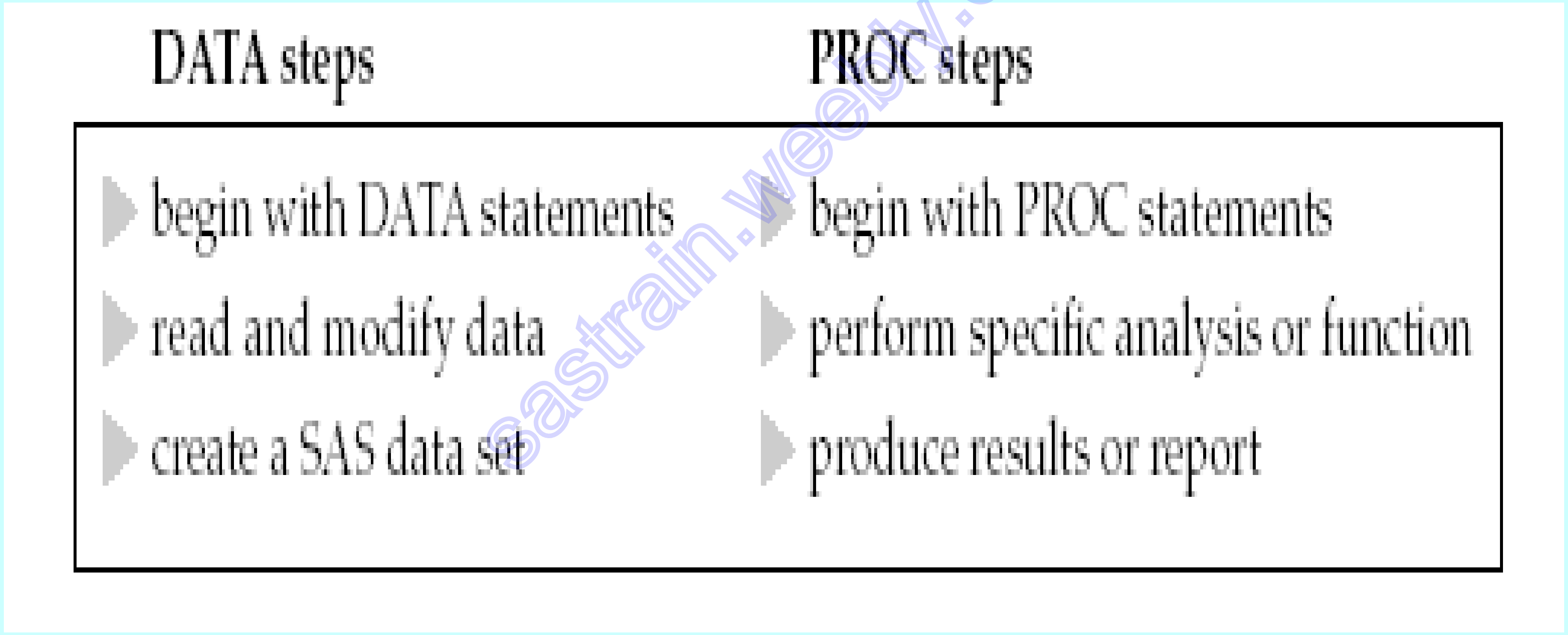
  X = ‘’ or “” ;

  X = ‘   a’ ; → trim(left(x)) = compress(x, ‘’) ;

  Y= ‘x’ || ’+5’ ;

The Two Parts of a SAS Program

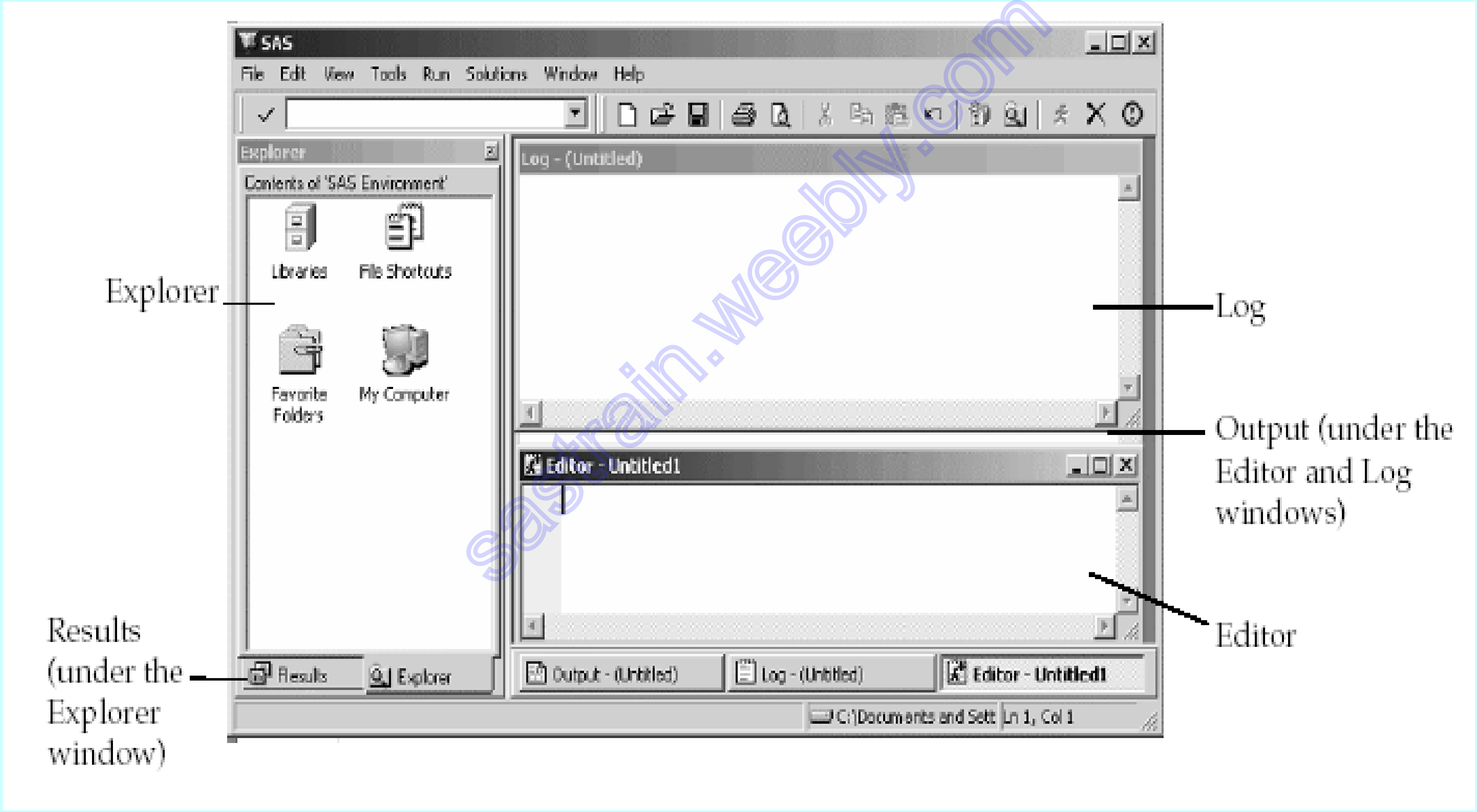
SAS programs are constructed from two basic building blocks: DATA steps and PROC steps.



DATA steps	PROC steps
begin with DATA statements	begin with PROC statements
read and modify data	perform specific analysis or function
create a SAS data set	produce results or report

Windows and Commands in the SAS Windowing Environment

The SAS Windows: There are five basic SAS windows: the Results and Explorer windows, and three programming windows: Editor, Log, and Output.



Common Options I use in beg. Of programming:

Options nofmterr mprint mlogic symbolgen ;

Other options better to know:

Linesize= n : control max. length of output lines

Pageno = n : starts numbering output pages with n. default is 1

Pagesize= n : controls the max number of lines per page of output

Yearcutoff =yyyy: specifies the first year in a hundred-year span for interpreting two-digit dates. Defalut is 1920> ex:10 is 1910 or 2010?

PART II: Get Data in

- Data Sources:
  - raw data(.txt, .dat, .csv): .csv: comma-separated values.
  - SAS datasets cleaned by data management group.
  - Other software formats: .xls using proc import

Raw data Source

- Two ways:

  -directly enter in sas: datalines statement/cards

  ---- refer to code 1.1

  - read in external raw datafiles

  ---- refer to code 1.2

Read Raw Data-
1. List style-ref. code 1.3

- Used when: all separated by at least one space

  The Good: direct and simple. **List vars in INPUT in the order as those in dataset!**

  Reqirement :

  - must read-in all variables- no skipping
  - Any missing must be indicated as **.**
  - Char. Cant have embedded spaces and length <=8
  - Char. Var Has to put $

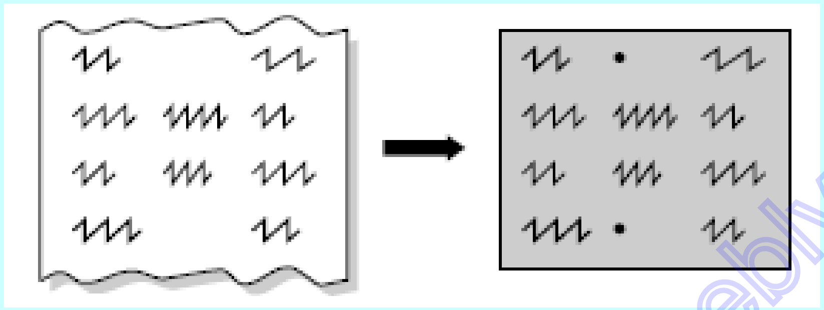
Read Raw Data-
2. Column styles-ref code 1.4

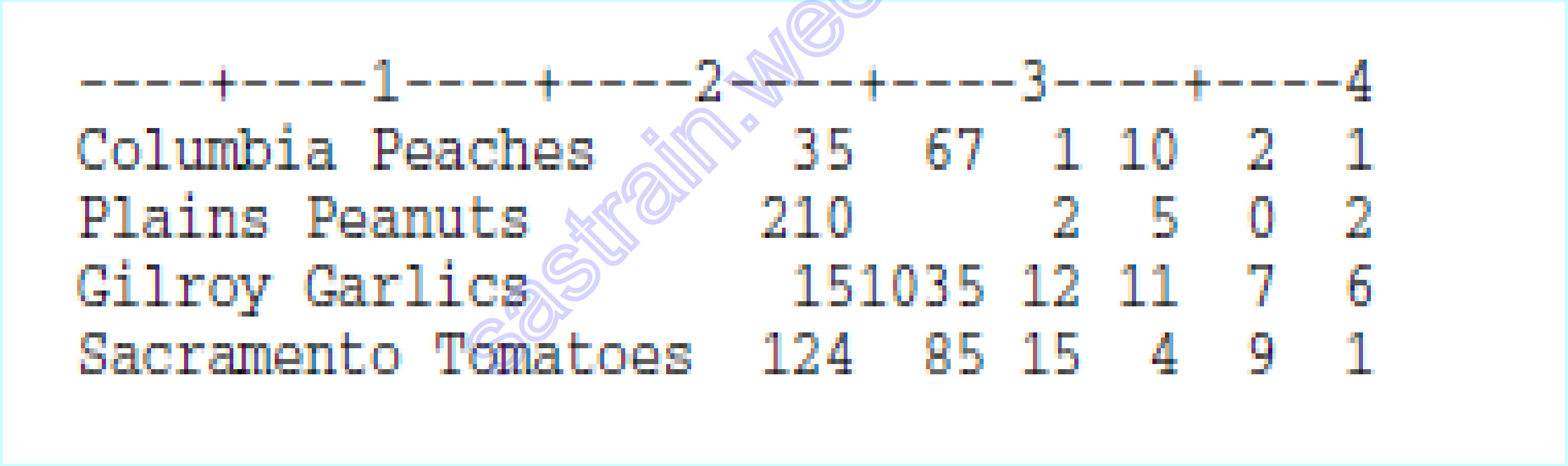
- Used when:
  - don’t have space between
  - No periods for missing data
  - But each value of Var always in exactly same place

Good things:

- Char var can have embedded spaces
- you can skip unwanted variables .

Cont. column styles





```
----+----1----+----2----+----3----+----4
Columbia Peaches          35   67  1 10  2  1
Plains Peanuts           210        2  5  0  2
Gilroy Garlics            151035 12 11  7  6
Sacramento Tomatoes      124   85 15  4  9  1
```

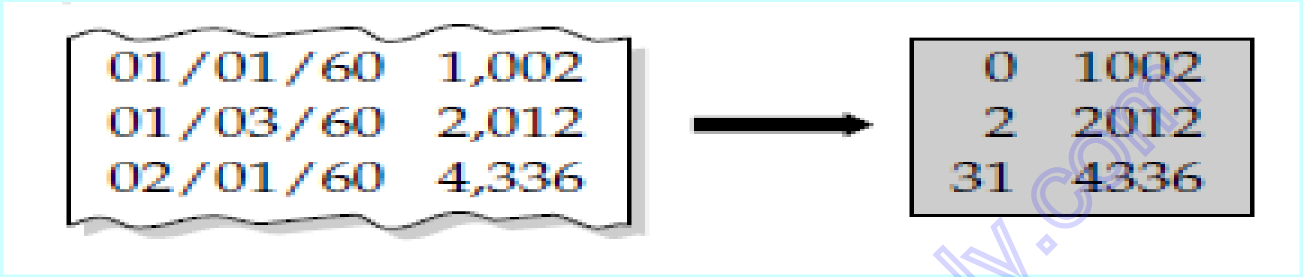
Read in Non-standard formats for Numeric var -ref code 1.5

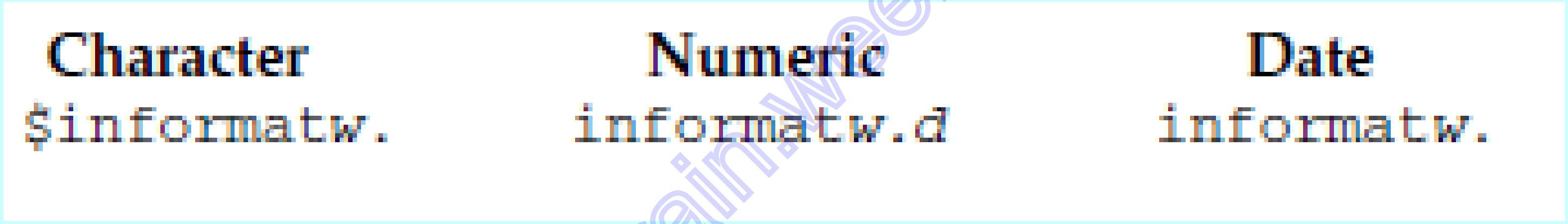
- Standard numeric data contain only:
  -numerals,
  -decimal points,
  -plus and minus signs, and E for scientific notation.

Non standard numeric data:

- comma: 4,065,493
- dates: 10-28-2003

Cont. read in Non-standard formats





Character	Numeric	Date
`$informatw.`	`informatw.d`	`informatw.`

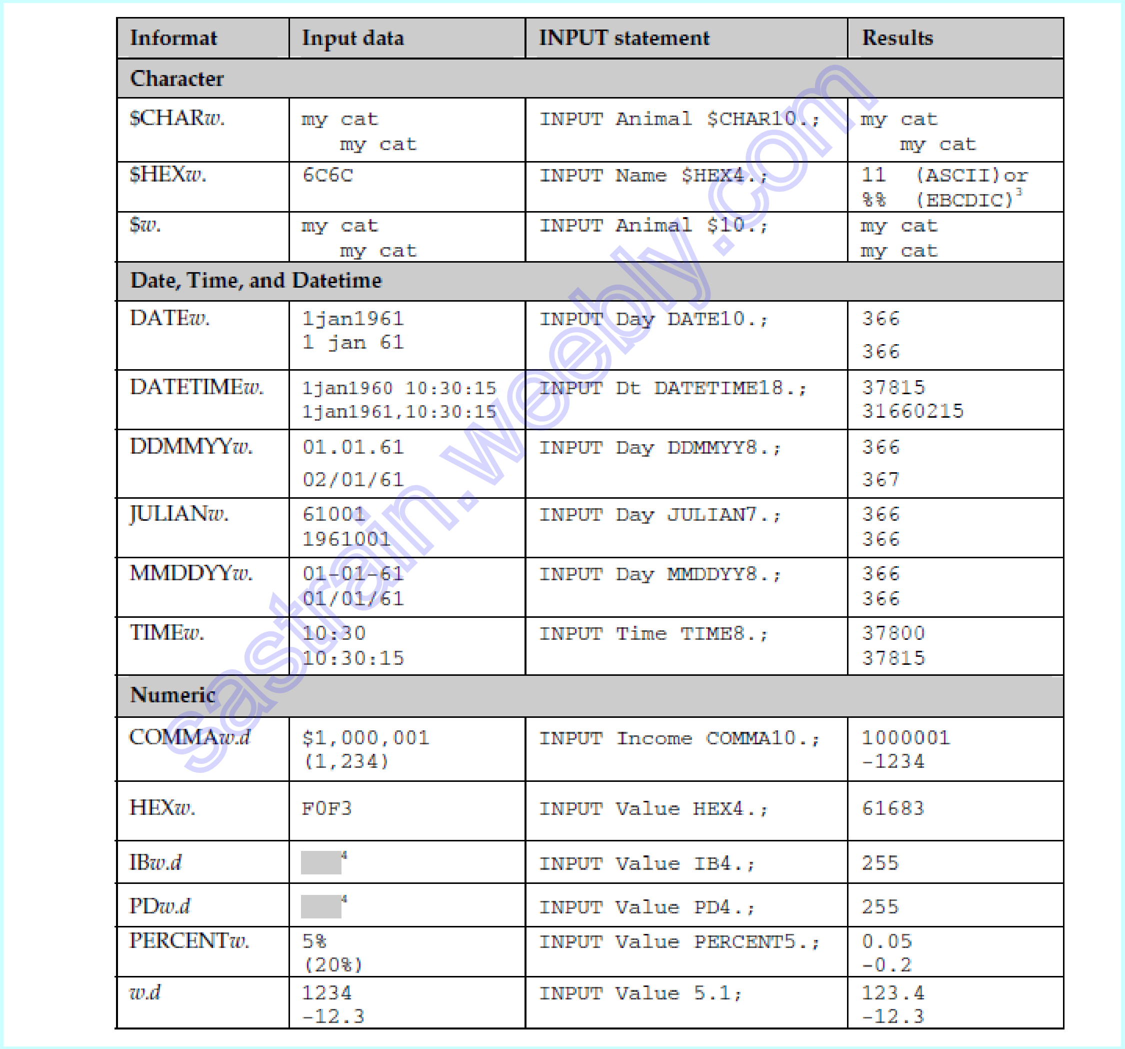
LETS REPEAT 5 TIMES:

PUT --→ output → FORMAT

INPUT → read in →INFORMAT

INPUT Name $16. Age **3. +1 Type $1. +1 Date MMDDYY10.**
(Score1 Score2 Score3 Score4 Score5) (**4.1);**

Common used informat for both Char and Num. variables



Informat	Input data	INPUT statement	Results
Character			

$CHAR*w*.	`my cat`
	`   my cat`
		`INPUT Animal $CHAR10.;`	`my cat`
			`   my cat`

$HEX*w*.	`6C6C`	`INPUT Name $HEX4.;`	`11  (ASCII) or` `%%  (EBCDIC)`[3]

$*w*.	`my cat`
	`   my cat`
		`INPUT Animal $10.;`	`my cat`
			`my cat`

Date, Time, and Datetime			

DATE*w*.	`1jan1961`
	`1 jan 61`
		`INPUT Day DATE10.;`	`366`
			`366`

DATETIME*w*.	`1jan1960 10:30:15`
	`1jan1961,10:30:15`
		`INPUT Dt DATETIME18.;`	`37815`
			`31660215`

DDMMYY*w*.	`01.01.61`
	`02/01/61`
		`INPUT Day DDMMYY8.;`	`366`
			`367`

JULIAN*w*.	`61001`
	`1961001`
		`INPUT Day JULIAN7.;`	`366`
			`366`

MMDDYY*w*.	`01-01-61`
	`01/01/61`
		`INPUT Day MMDDYY8.;`	`366`
			`366`

TIME*w*.	`10:30`
	`10:30:15`
		`INPUT Time TIME8.;`	`37800`
			`37815`

Numeric			

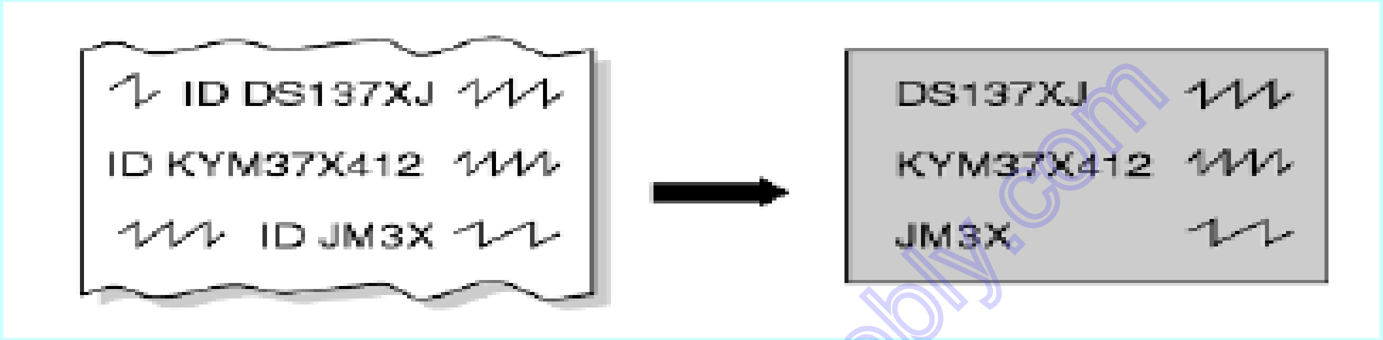
COMMA*w.d*	`$1,000,001`
	`(1,234)`
		`INPUT Income COMMA10.;`	`1000001`
			`-1234`

HEX*w*.	`F0F3`	`INPUT Value HEX4.;`	`61683`
IB*w.d*	▨[4]	`INPUT Value IB4.;`	`255`
PD*w.d*	▨[4]	`INPUT Value PD4.;`	`255`

PERCENT*w*.	`5%`
	`(20%)`
		`INPUT Value PERCENT5.;`	`0.05`
			`-0.2`

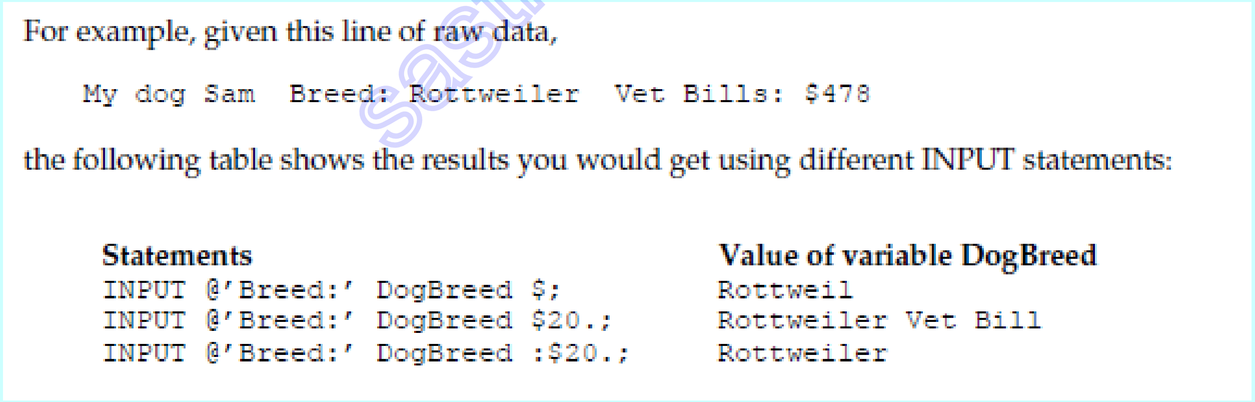
w.d	`1234`
	`-12.3`
		`INPUT Value 5.1;`	`123.4`
			`-12.3`

Read- in messy raw data - ref to code 1.6



- Using The @*'character' column pointer: start to read after that ''*

- Using The colon *modifier: read until it encounters a space*



For example, given this line of raw data,

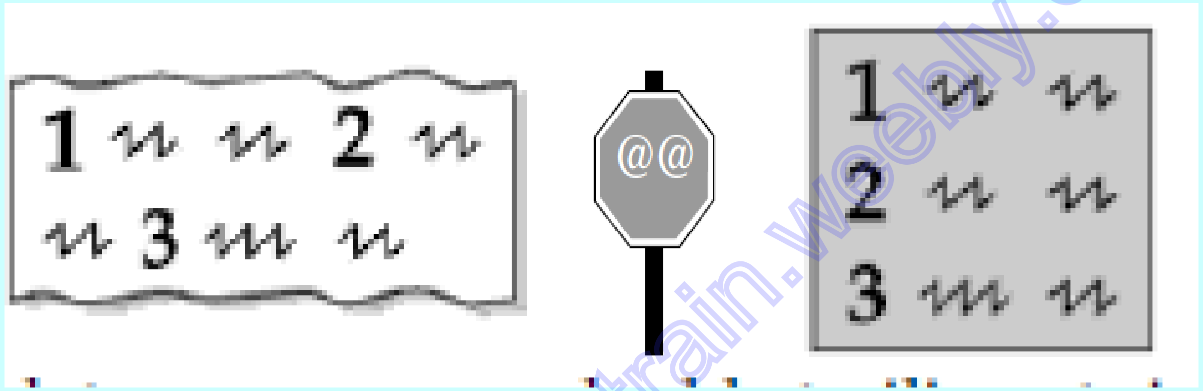
```
My dog Sam  Breed: Rottweiler  Vet Bills: $478
```

the following table shows the results you would get using different INPUT statements:

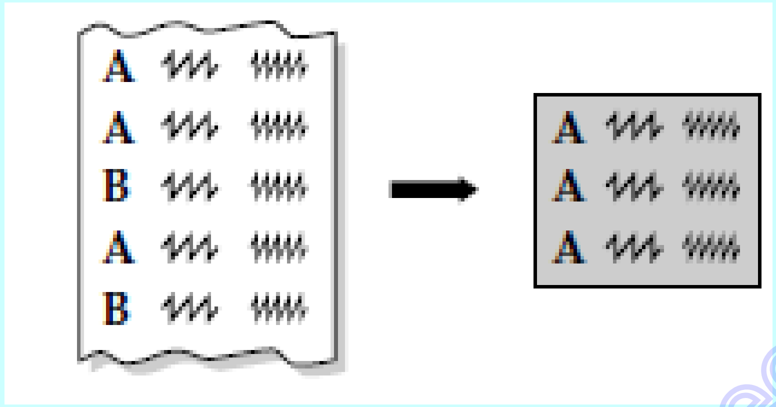
Statements	Value of variable DogBreed
`INPUT @'Breed:' DogBreed $;`	`Rottweil`
`INPUT @'Breed:' DogBreed $20.;`	`Rottweiler Vet Bill`
`INPUT @'Breed:' DogBreed :$20.;`	`Rottweiler`

Reading multiple obs per line of raw data- ref. code 1.7

- Goal: using double trailing at signs to read in multiple obs per line of raw data



Reading part of a Raw data file -ref code 1.8



- Use single trailing at sign @ to HOLD line of raw data to decide if to continue read in

“Stay tuned for more info, don’t touch dial”

Common INFILE options -ref to code 1.9

- FIRSTOBS=: where to begin
- OBS= : where to stop
- FIRSTOBS =3 OBS= 5: total 3 lines read in
- MISSOVER: by default, SAS goes to next line if there are still vars need to be assigned. This tell SAS to stop and give missing to them.

CONT. INFILE options -ref to code 1.9

- TRUNCOVER: tell SAS that’s it!

for column or formatted input and some data lines are shorter than others. By default SAS will go to next line to start reading. by default, SAS will go to the next line to start reading the variable’s value. This option tells SAS to read data for the variable until it reaches the end of the data line, or the last column specified in the format or column range, whichever comes first.

CONT. INFILE options

Question: Compare MISSOVER and TRUNCOVER ?

DLM= option: for common delimiters

DSD (Delimiter-Sensitive Data) option: default is comma, so if not, used with DLM

- ignores delimiters enclosed in quotation marks.

- it does not read quotation marks as part of the data value

- Third, it treats two delimiters in a row as a missing value.

LIBNAME & PROC CONTENTS -ref. code 1.10

- Where you want to store?

  - TEMP: work library, no need to define

  - PERM: use libname to read in and output

LIBNAME *libref 'your-SAS-data-library';*

Proc contents: to show # of obs, variables info and formats

PART III: WORKING WITH DATA

- define variables & + - * /: be careful about missing values—ref. code 1.11

Background info: what is clean log: ctrl +f

No error

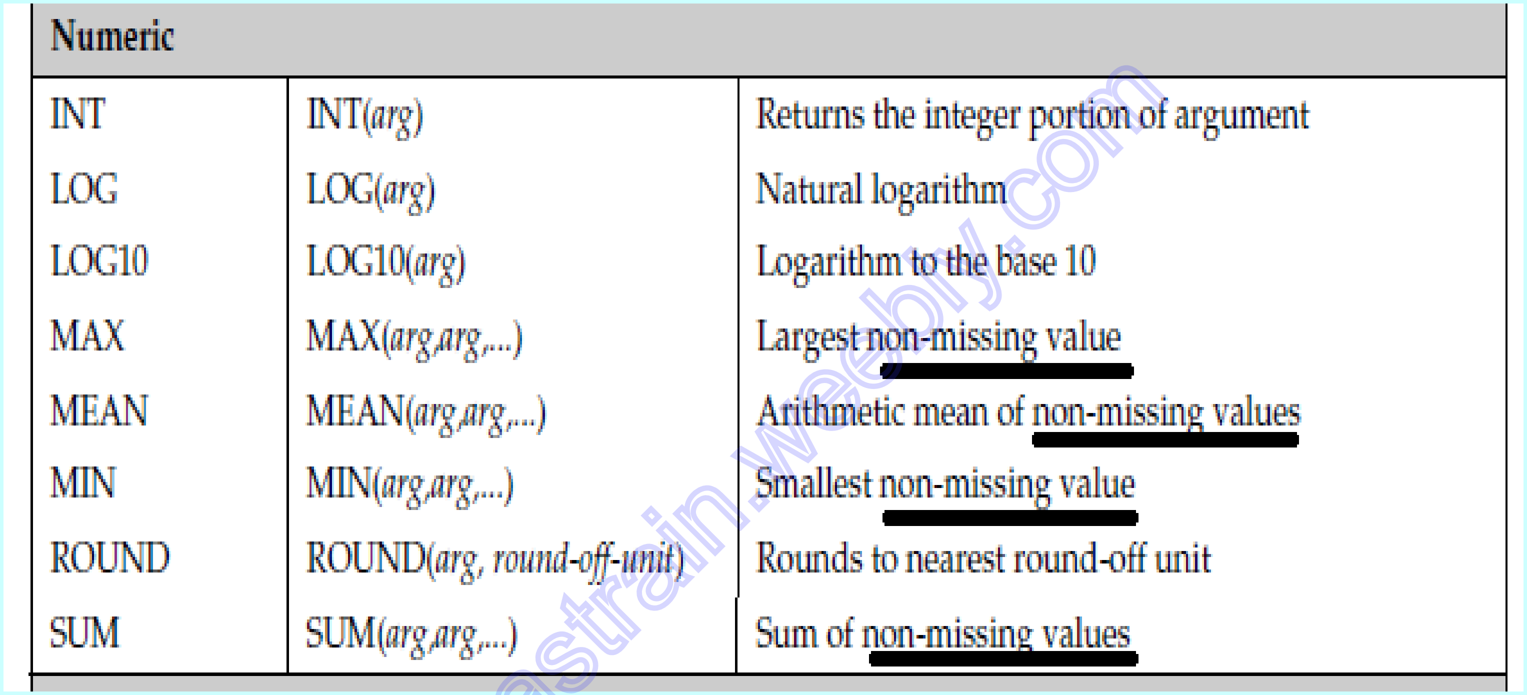
No warning

No missing

No un-initialize

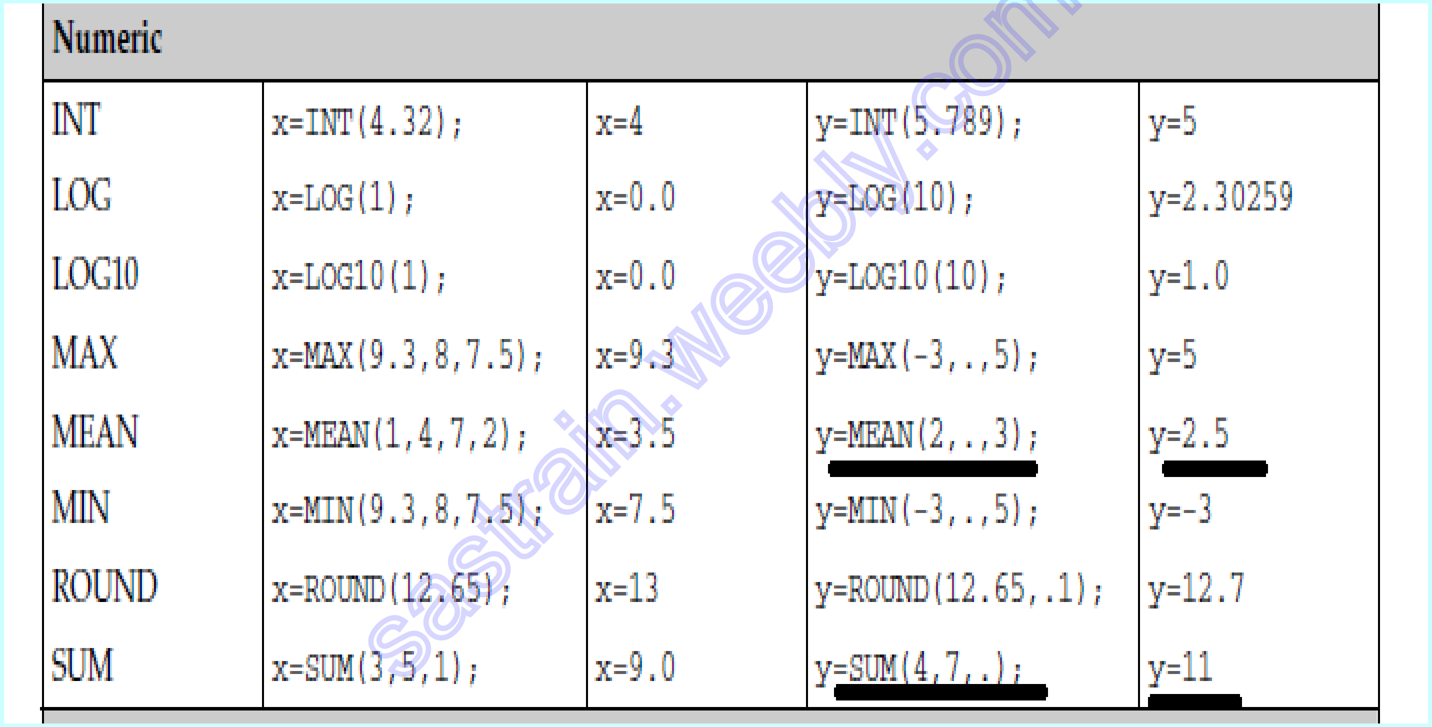
No multiple merging issue

Useful functions -Numeric



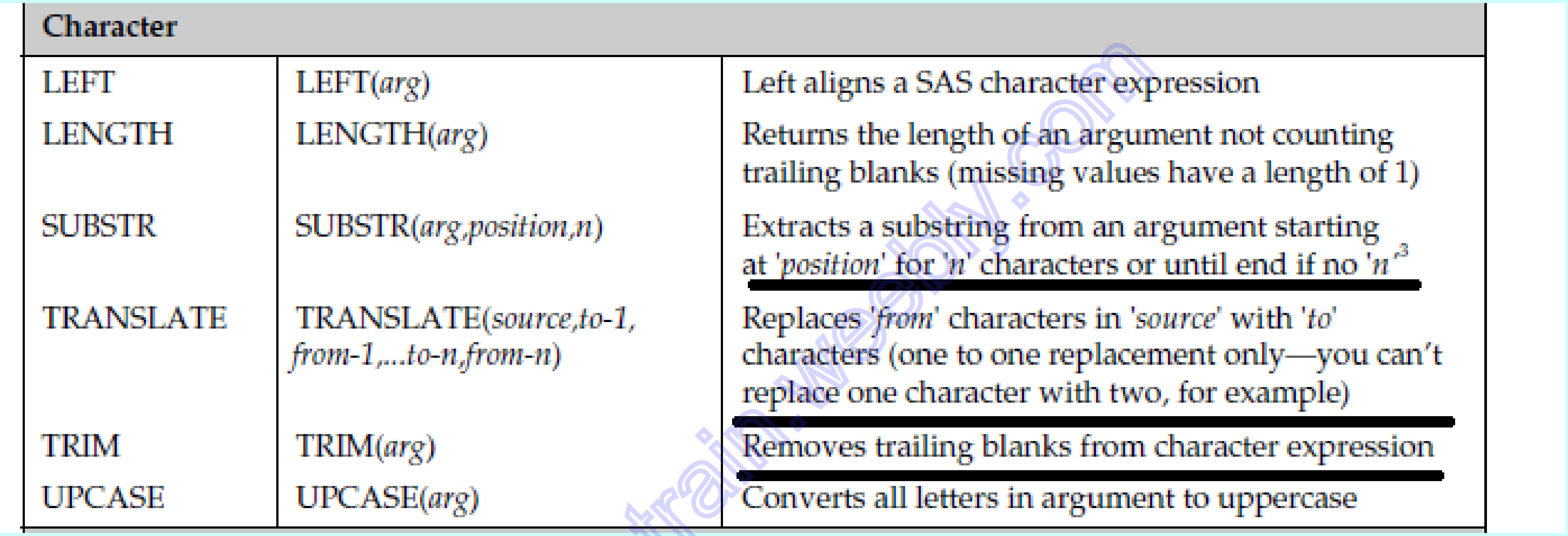
Numeric		
INT	INT(*arg*)	Returns the integer portion of argument
LOG	LOG(*arg*)	Natural logarithm
LOG10	LOG10(*arg*)	Logarithm to the base 10
MAX	MAX(*arg,arg,...*)	Largest non-missing value
MEAN	MEAN(*arg,arg,...*)	Arithmetic mean of non-missing values
MIN	MIN(*arg,arg,...*)	Smallest non-missing value
ROUND	ROUND(*arg, round-off-unit*)	Rounds to nearest round-off unit
SUM	SUM(*arg,arg,...*)	Sum of non-missing values

CONT.Useful functions -Numeric



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
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

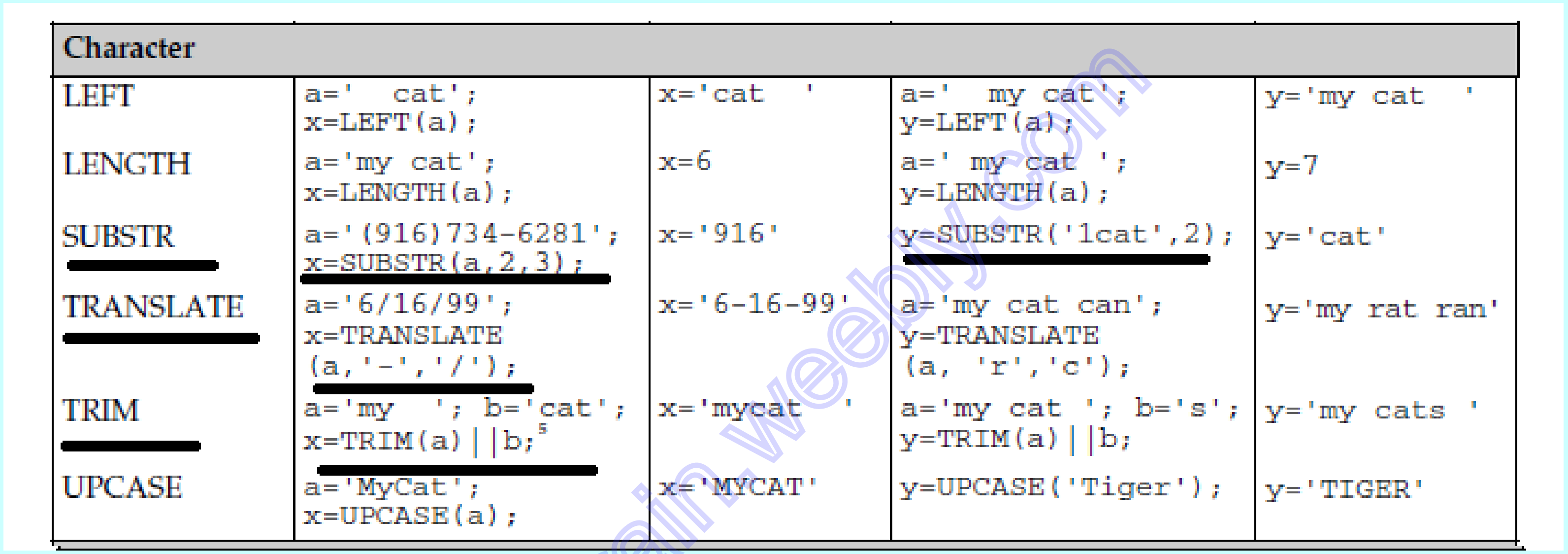
Useful functions
- Char.



Character		
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)
SUBSTR	SUBSTR(*arg,position,n*)	Extracts a substring from an argument starting at '*position*' for '*n*' characters or until end if no '*n*'[3]
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)
TRIM	TRIM(*arg*)	Removes trailing blanks from character expression
UPCASE	UPCASE(*arg*)	Converts all letters in argument to uppercase

COMPRESS(arg. , ‘’) : delete quoted char. = trim(left(arg.))
SCAN(arg., count, delimiters) :
1) If ***count*** is positive, SCAN counts words from left to right in the character string.
2) If ***count*** is negative, SCAN counts words from right to left in the character string.

CONT.Useful functions - Char.

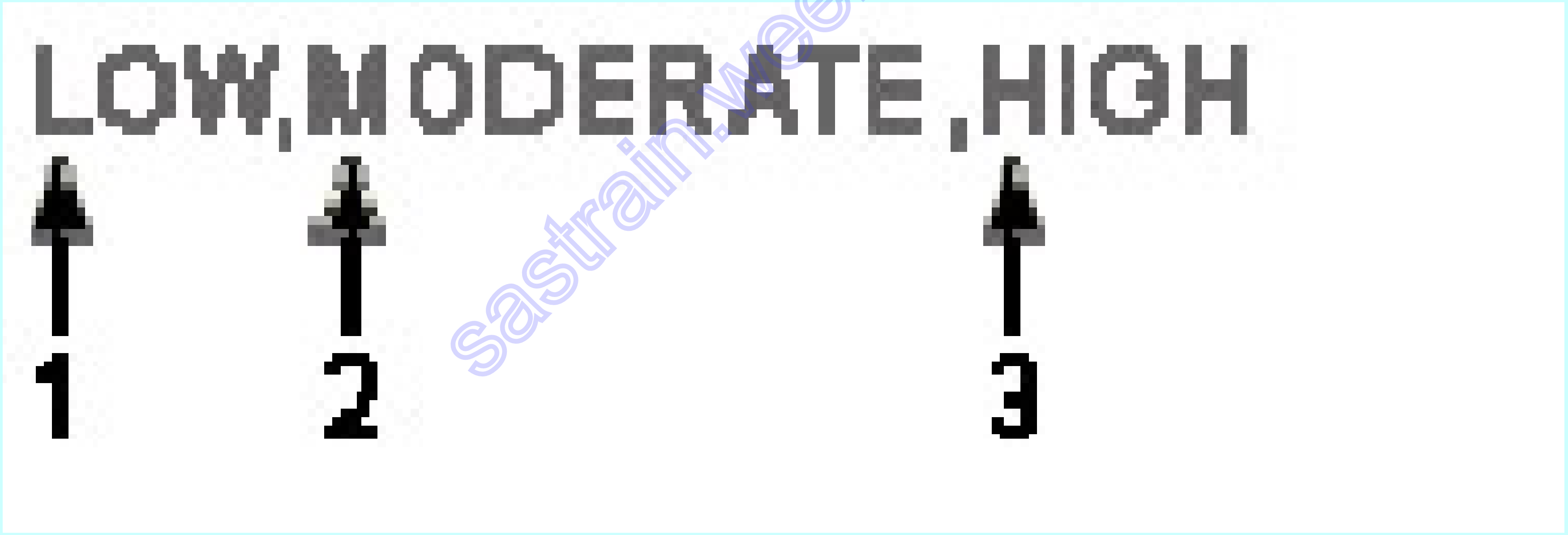


Character				
LEFT	a=' cat'; x=LEFT(a);	x='cat '	a=' my cat'; y=LEFT(a);	y='my cat '
LENGTH	a='my cat'; x=LENGTH(a);	x=6	a=' my cat '; y=LENGTH(a);	y=7
SUBSTR	a='(916)734-6281'; x=SUBSTR(a,2,3);	x='916'	y=SUBSTR('1cat',2);	y='cat'
TRANSLATE	a='6/16/99'; x=TRANSLATE (a,'-','/');	x='6-16-99'	a='my cat can'; y=TRANSLATE (a, 'r','c');	y='my rat ran'
TRIM	a='my '; b='cat'; x=TRIM(a)||b;[5]	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'

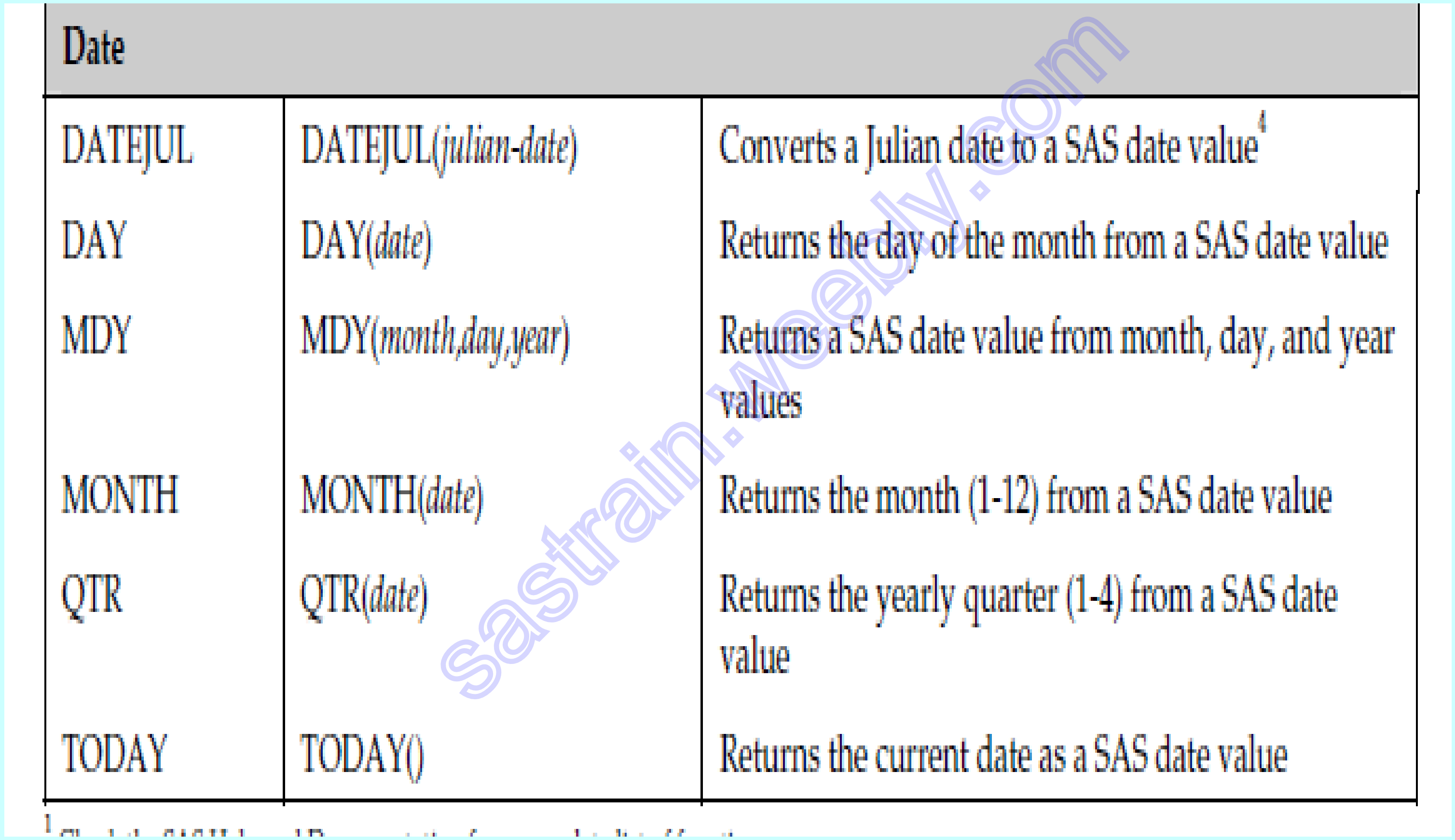
1) When combine "||", trim first!
2) When count length, trim first!
3) Use translate to deal with messy datasets. 2010-10/10

CONT.Useful functions - Char.

Question: tell me difference between SCAN() and SUBSTR() ; ??? ---ref. code 1.12



Useful functions - DATES



Date		
DATEJUL	DATEJUL(*julian-date*)	Converts a Julian date to a SAS date value[4]
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

1 [illegible]

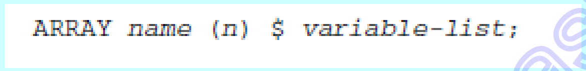
IF THEN LOOP

Work with SAS dates -ref code 1.13

Cont. Dates summary

Retain and Sum stat.
-ref code 1.14

Arrays
-ref. code 1.15



```
ARRAY name (n) $ variable-list;
```

Outlines of procs

PROC SORT
--ref code 1.16

PROC PRINT
-ref code 1.17

PROC FORMAT
-ref code 1.18

PROC MEANS
-ref. code 1.19

PROC FREQ

-

CONT. PROC FREQ

CONT. PROC FREQ
-ref. code 1.23

How to read in format -ref code 1.24

PROC REPORT- Part 1 -ref code 1.25

PROC REPORT- Part 2 -ref code 1.26

PROC TRANSPOSE

Lets talk about combining datasets

Stacking

Interleaving

One to one merge

One to many merge

Many to many merge. XXX

Character Truncation-p276

SAS DATASET OPTION

Tracking and selection with IN option

Output multiple datasets

Output duplicates records

Using SAS automatic Vars

Convert Num→Char. Or vice versa– p.266

COMMON MISTAKES