Data

Data Step

Data <new name>;

Retain <variables>; can be used to set the order of variables in the table (must come before the set statement)

Set <libref.file>; libref not required if the file is in the working library

Length <variable><$> W or <variable> w.d; where W=number of print places for character variable or for numerical W=bites and .d = decimal places

Assign new variables; type new variable name and then define

Label;

Format;

Where; create a data subset by limiting the table to a particular level of a variable

Drop <variables>; if the drop statement is used within set statement then variables are not read into the PDV (drop=) and are thus not available for processing

Keep <variables>; limits the variables that are saved to the new dataset

If condition;

run;

Multiple data sets an be created in one data step, which is best done with a select function:

Select (<variable>);

When ('level1') output <data1>;

When ('level2') output <data2>;

Otherwise; optional to otherwise output other

end;

To avoid errors due to capitalisation, use the **upcase** or **lowcase** function, e.g., **select (lowcase(<variable>))**

To collapse levels of a variable, e.g, to collapse value 3 into 2

<variable2> = <variable1>;

if variable1=3 then variable2=2;

Converting character to numerical variables:

numvar = INPUT(charvar, best32.);

or

<variable> = <variable> + 0

Conditional Processing

If <argument>; restricts data set to the condition

Use **else if** for mutually exclusive statements (more efficient). Only 1 executable statement is allowed per IF-THEN statement; to perform multiple arguments then use IF-DO statement:

If <argument> then do; argument 1; Argument 2; argument 3 …; end;

If …. then do; arguments…..; end;

Else do….; end; run;

Inclusive range for IF statement can be achieved with: **5 le <variable> le 7;** gives observations between 5-7 inclusive

Numerical operators

Mean	Arithmetic mean
Sum	Sum or arguments
Var	Variance
Sin	Sine
Log	Natural log
SQRT	Square root
ABS	Absolute value

Logical operators

Modify where expressions: **not / and / or**

Eg **where city not in('London','Rome','Paris')** = city is not London, Rome or Paris

Comparison operators

Equal to	eq	=
Not equal to	ne	
Less than	lt	<
Greater than or equal	ge	>=
Less than or equal	le	<=
Equal to one of a list	in	

Special where operators (can only be used for where statements)

Operator	Definition	Char	num
Contains	Includes substring	x	
Between and	Inclusive range	x	x
Is null	A missing value	x	x
Is missing	A missing value	x	x
Like	Matches a pattern	x	
Where same and	Augments original condition	x	x