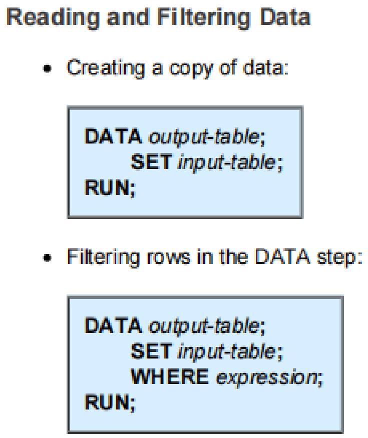
2.treatingData

March 3, 2025

0.0.1 Preparing Data

In Preparing Data you learn how to do some common data manipulations, such as filtering rows and columns, computing new columns, and performing conditional processing.



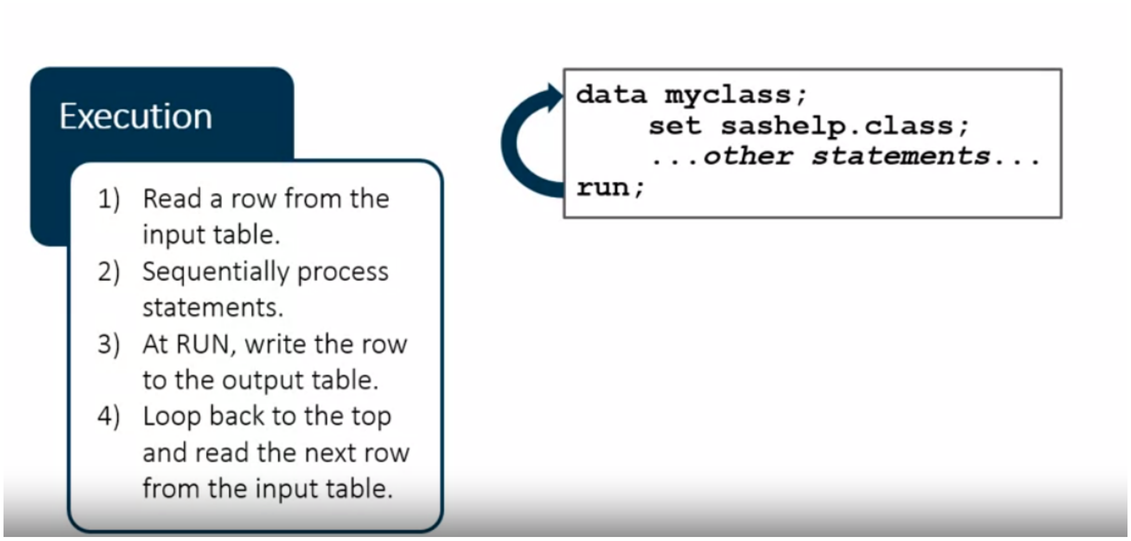
Reading and Filtering Data

- Creating a copy of data:

```
DATA output-table;
      SET input-table;
RUN;
```

- Filtering rows in the DATA step:

```
DATA output-table;
      SET input-table;
      WHERE expression;
RUN;
```

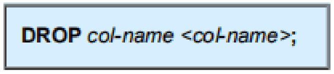


Execution

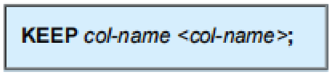
1) Read a row from the input table.
2) Sequentially process statements.
3) At RUN, write the row to the output table.
4) Loop back to the top and read the next row from the input table.

```
data myclass;
    set sashelp.class;
    ...other statements...
run;
```

- Specifying columns to include in the output data set:

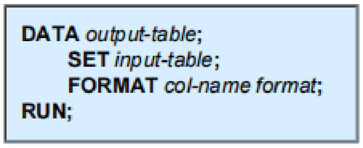


```
DROP col-name <col-name>;
```



```
KEEP col-name <col-name>;
```

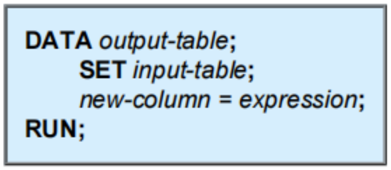
- Formatting columns in the DATA step:



```
DATA output-table;
      SET input-table;
      FORMAT col-name format;
RUN;
```

Computing New Columns

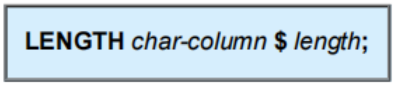
- Using expressions to create new columns:



```
DATA output-table;
      SET input-table;
      new-column = expression;
RUN;
```

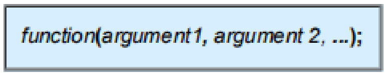
- The name of the column to be created or updated is listed on the left side of the equals sign.
- Provide an expression on the right side of the equal sign.

- SAS automatically defines the required attributes if the column is new – name, type, and length.
- A new numeric column has a length of 8.
- The length of a new character column is determined based on the length of the assigned string.
- Character strings must be quoted and are case sensitive.
- Creating character columns:

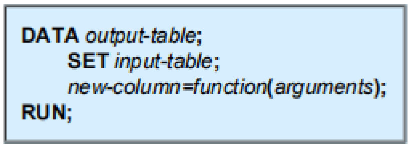


```
LENGTH char-column $ length;
```

- Using functions in expressions:

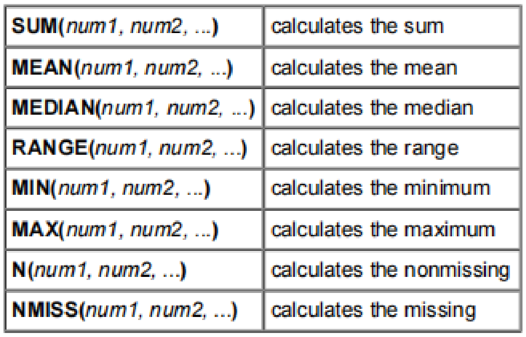


```
function(argument1, argument 2, ...);
```



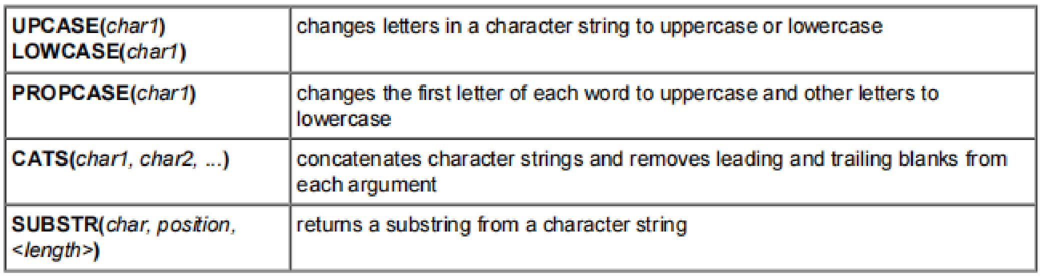
```
DATA output-table;
    SET input-table;
    new-column=function(arguments);
RUN;
```

- Functions for calculating summary statistics (ignore missing values):



SUM(*num1, num2, ...***)**	calculates the sum
MEAN(*num1, num2, ...***)**	calculates the mean
MEDIAN(*num1, num2, ...***)**	calculates the median
RANGE(*num1, num2, ...***)**	calculates the range
MIN(*num1, num2, ...***)**	calculates the minimum
MAX(*num1, num2, ...***)**	calculates the maximum
N(*num1, num2, ...***)**	calculates the nonmissing
NMISS(*num1, num2, ...***)**	calculates the missing

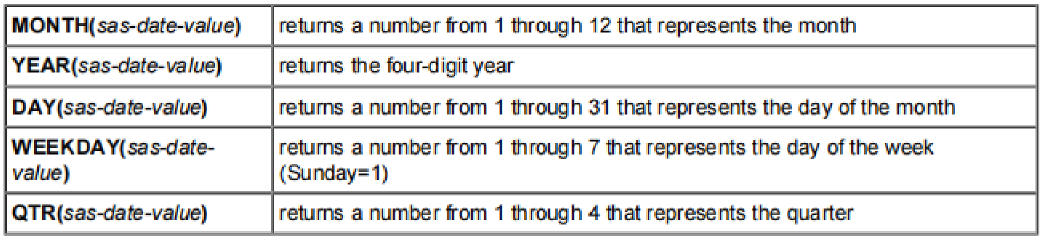
- Character functions:



UPCASE(*char1***)**
LOWCASE(*char1***)**
	changes letters in a character string to uppercase or lowercase

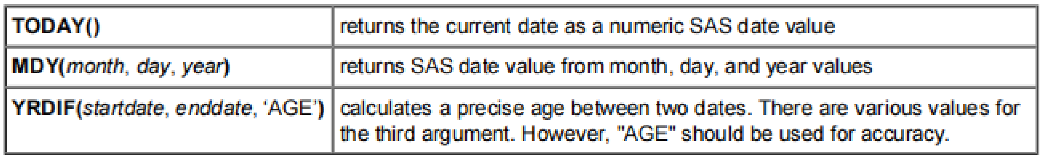
PROPCASE(*char1***)**	changes the first letter of each word to uppercase and other letters to lowercase
CATS(*char1, char2, ...***)**	concatenates character strings and removes leading and trailing blanks from each argument
SUBSTR(*char, position, <length>***)**	returns a substring from a character string

- Date functions that extract information from SAS date values:



MONTH(*sas-date-value***)**	returns a number from 1 through 12 that represents the month
YEAR(*sas-date-value***)**	returns the four-digit year
DAY(*sas-date-value***)**	returns a number from 1 through 31 that represents the day of the month
WEEKDAY(*sas-date-value***)**	returns a number from 1 through 7 that represents the day of the week (Sunday=1)
QTR(*sas-date-value***)**	returns a number from 1 through 4 that represents the quarter

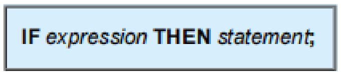
- Date functions that create SAS date values:



TODAY()	returns the current date as a numeric SAS date value
MDY(*month, day, year***)**	returns SAS date value from month, day, and year values
YRDIF(*startdate, enddate,* 'AGE'**)**	calculates a precise age between two dates. There are various values for the third argument. However, "AGE" should be used for accuracy.

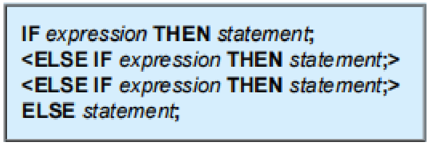
Conditional Processing

- Conditional processing with IF-THEN logic:



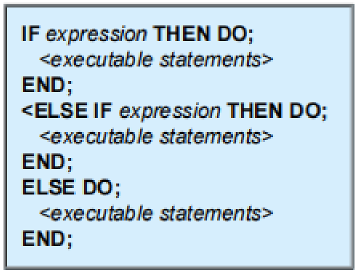
```
IF expression THEN statement;
```

- Conditional processing with IF-THEN-ELSE:



```
IF expression THEN statement;
<ELSE IF expression THEN statement;>
<ELSE IF expression THEN statement;>
ELSE statement;
```

- Processing multiple statements with IF-THEN-DO:



```
IF expression THEN DO;
  <executable statements>
END;
<ELSE IF expression THEN DO;
  <executable statements>
END;
ELSE DO;
  <executable statements>
END;
```

- After the IF-THEN-DO statement, list any number of executable statements.
- Close each DO block with an END statement.

[ ]: