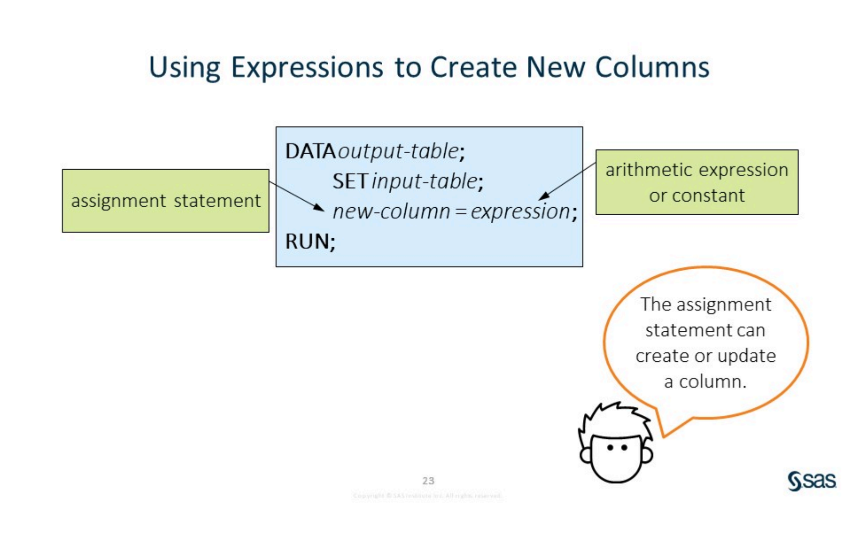
B4.2 - Computing New Columns

Creating New Columns with Expressions



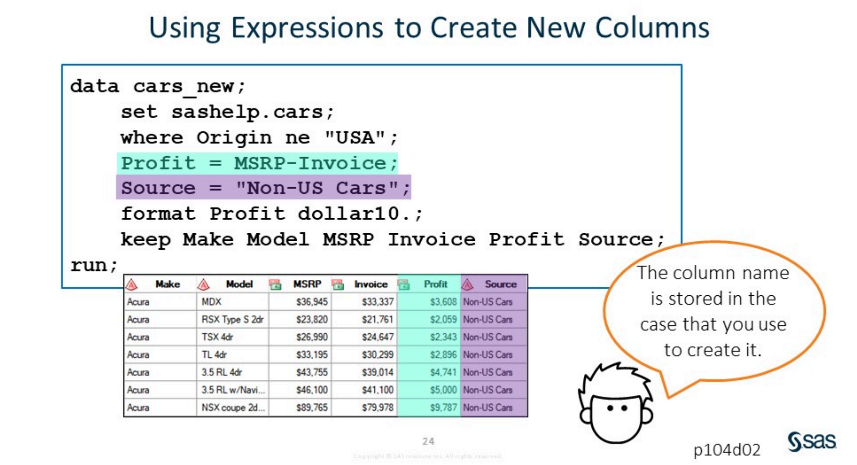
Using Expressions to Create New Columns

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

The assignment statement can create or update a column.

23

Often your data does not have ***all*** the columns that you need, and you might want to calculate or derive new columns from existing columns. Fortunately, this is easy to do in the DATA step. To create new columns, you use an assignment statement. You simply type the name of the new column, an equal sign, and then the expression that creates a new data value.



Using Expressions to Create New Columns

```
data cars_new;
    set sashelp.cars;
    where Origin ne "USA";
    Profit = MSRP-Invoice;
    Source = "Non-US Cars";
    format Profit dollar10.;
    keep Make Model MSRP Invoice Profit Source;
run;
```

Make	Model	MSRP	Invoice	Profit	Source
Acura	MDX	$36,945	$33,337	$3,608	Non-US Cars
Acura	RSX Type S 2dr	$23,820	$21,761	$2,059	Non-US Cars
Acura	TSX 4dr	$26,990	$24,647	$2,343	Non-US Cars
Acura	TL 4dr	$33,195	$30,299	$2,896	Non-US Cars
Acura	3.5 RL 4dr	$43,755	$39,014	$4,741	Non-US Cars
Acura	3.5 RL w/Navi...	$46,100	$41,100	$5,000	Non-US Cars
Acura	NSX coupe 2d...	$89,765	$79,978	$9,787	Non-US Cars

The column name is stored in the case that you use to create it.

24

p104d02

In this example, the WHERE statement includes rows where **Origin** is not equal to *USA*. The first assignment statement creates the new column **Profit** using a simple arithmetic expression. SAS creates the numeric column **Profit** and generates a value for every row in the output table by subtracting **Invoice** from **MSRP**. The second assignment statement creates a column named **Source** and assigns the character string *Non-US Cars*. Notice that because there is a KEEP statement, you must explicitly list the new columns so that they are included in the **cars_new** table.

Demo: Using Expressions to Create New Columns

4_1 - Demo - Using Expressions to Create New Columns.pdf (https://clemson.instructure.com/courses/237270/files/23073939/download?wrap=1) **(https://clemson.instructure.com/courses/237270/files/23073939/download?download_frd=1)**

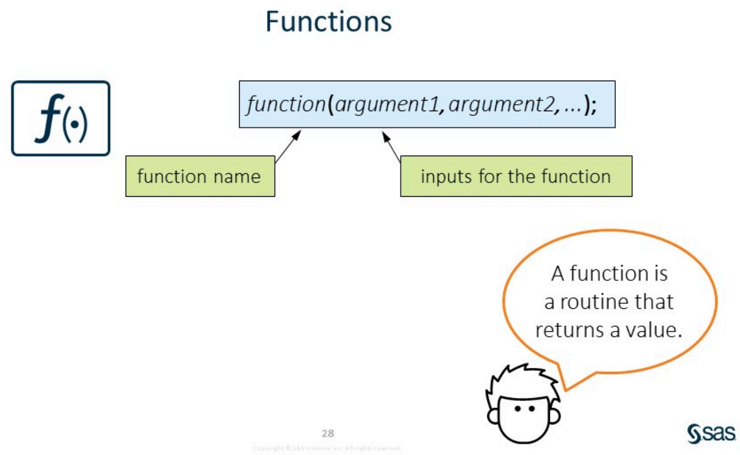
Activity 4.04:

Open **p104a04.sas** from the **activities** folder and perform the following tasks:

1. Add an assignment statement to create **StormLength** that represents the number of days between **StartDate** and **EndDate**.
2. Run the program. In 1980, how long did the storm named Agatha last?

Click here for Solution.

Functions



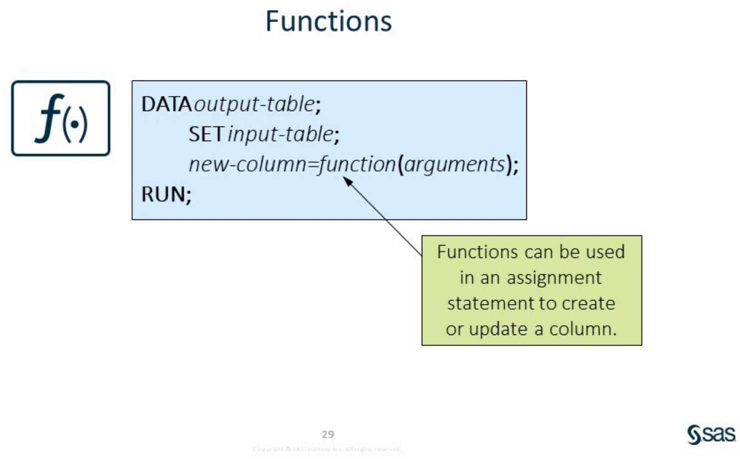
Functions

function(argument1, argument2, ...);

28

Arithmetic calculations and character constants are a good start for creating new columns, but often you need more elaborate or flexible methods for generating the new data values. SAS offers hundreds of functions that can be used in countless ways to manipulate numeric, character, and date values. We just scratch the surface of what you can do with functions in this class, but I encourage you to go to the documentation to see a complete listing of functions, along with syntax and examples.

The syntax for a function is the function name, followed by the arguments enclosed in parentheses. The arguments are separated by commas. The arguments consist of the input that the function needs to perform its specific routine and return a value.



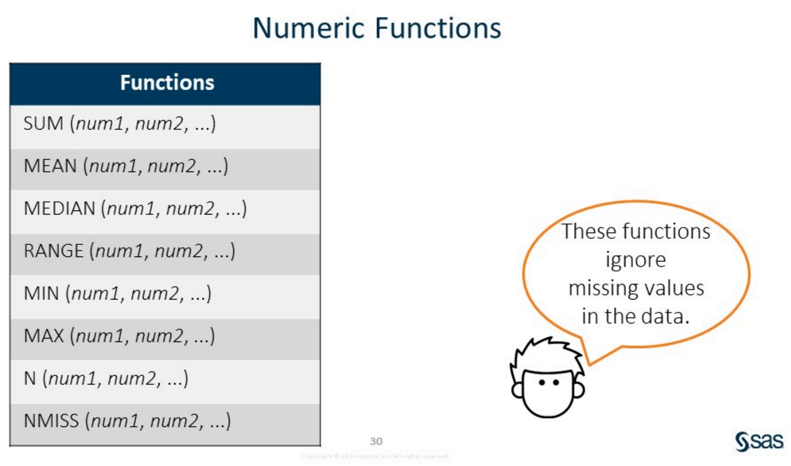
Functions

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

Functions can be used in an assignment statement to create or update a column.

29

Functions can be used in assignment statements to generate a value for the column.

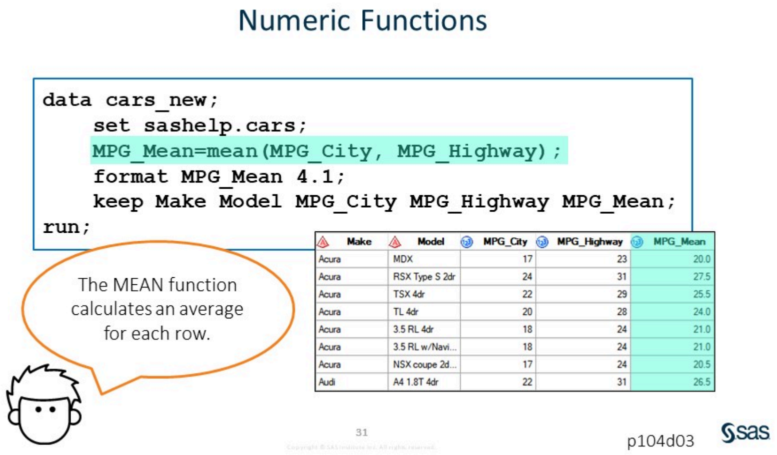


Numeric Functions

Functions
SUM (*num1*, *num2*, ...)
MEAN (*num1*, *num2*, ...)
MEDIAN (*num1*, *num2*, ...)
RANGE (*num1*, *num2*, ...)
MIN (*num1*, *num2*, ...)
MAX (*num1*, *num2*, ...)
N (*num1*, *num2*, ...)
NMISS (*num1*, *num2*, ...)

These functions ignore missing values in the data.
30
sas

Let’s start with a few examples of numeric functions. SAS has a collection of summary statistics functions, including SUM, MEAN, MEDIAN, and RANGE. Each of these functions can have an unlimited number of arguments, and each argument provides either a numeric constant or numeric column in the data. The function calculates the summary statistic from the values of the arguments for each row in the data. One interesting note about these summary functions is that if any of the input values are missing, the missing value or values are ignored, and the calculation is based on the known values.



Numeric Functions

```
data cars_new;
    set sashelp.cars;
    MPG_Mean=mean(MPG_City, MPG_Highway);
    format MPG_Mean 4.1;
    keep Make Model MPG_City MPG_Highway MPG_Mean;
run;
```

Make	Model	MPG_City	MPG_Highway	MPG_Mean
Acura	MDX	17	23	20.0
Acura	RSX Type S 2dr	24	31	27.5
Acura	TSX 4dr	22	29	25.5
Acura	TL 4dr	20	28	24.0
Acura	3.5 RL 4dr	18	24	21.0
Acura	3.5 RL w/Navi...	18	24	21.0
Acura	NSX coupe 2d...	17	24	20.5
Audi	A4 1.8T 4dr	22	31	26.5

In this example code, an assignment statement creates a column named **MPG_Mean**. The MEAN function is used with the arguments **MPG_City** and **MPG_Highway** to supply values for **MPG_Mean**. Notice that the FORMAT statement rounds the displayed values of **MPG_Mean** to one decimal place.

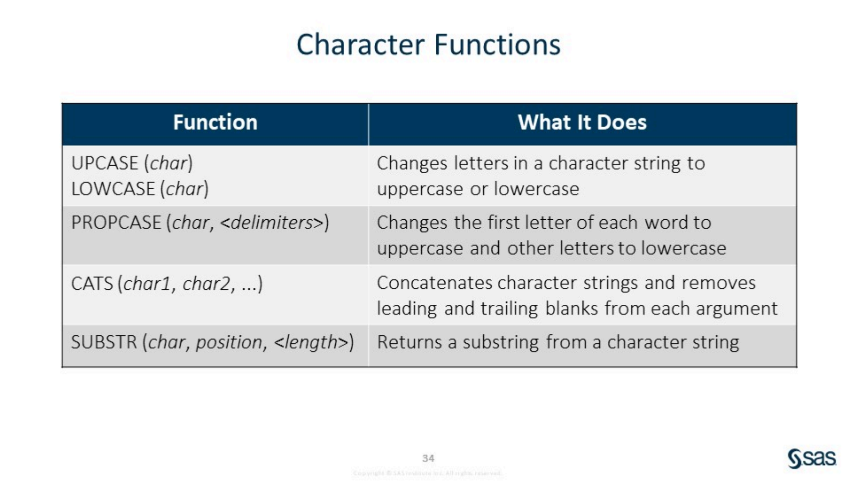
Activity 4.05

Open **p104a05.sas** from the **activities** folder and perform the following tasks:

1. Open the **pg1.storm_range** table and examine the columns. Notice that each storm has four wind speed measurements.
2. Create a new column named **WindAvg** that is the mean of **Wind1**, **Wind2**, **Wind3**, and **Wind4**.
3. Create a new column **WindRange** that is the range of **Wind1**, **Wind2**, **Wind3**, and **Wind4**.

Click here for Solution.

Character Functions

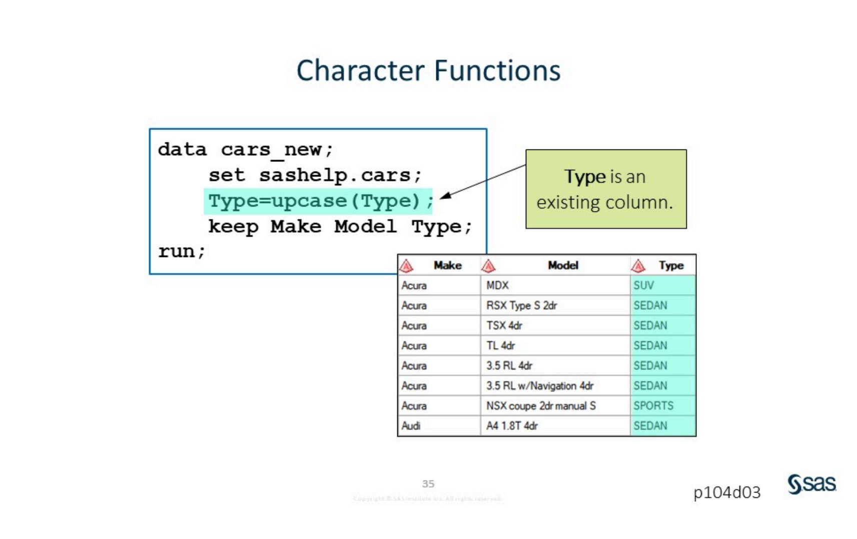


Character Functions

Function	What It Does
UPCASE (*char*) LOWCASE (*char*)	Changes letters in a character string to uppercase or lowercase
PROPCASE (*char*, <*delimiters*>)	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

34

Now let’s take a look at some ways we can create new columns by manipulating character columns. The syntax is the same: you start with the function name and specify the arguments in parentheses. This time the arguments include one or more character columns in your data. This is just four of the many character functions that SAS offers.



Character Functions

```
data cars_new;
    set sashelp.cars;
    Type=upcase(Type);
    keep Make Model Type;
run;
```

Type is an existing column.

Make	Model	Type
Acura	MDX	SUV
Acura	RSX Type S 2dr	SEDAN
Acura	TSX 4dr	SEDAN
Acura	TL 4dr	SEDAN
Acura	3.5 RL 4dr	SEDAN
Acura	3.5 RL w/Navigation 4dr	SEDAN
Acura	NSX coupe 2dr manual S	SPORTS
Audi	A4 1.8T 4dr	SEDAN

35

p104d03

As a simple example, let’s look at the UPCASE function. It requires one argument: a character column. As you would expect, the UPCASE function returns the uppercase equivalent of the input data values. In this case, we aren't creating a new column in the output data. We are changing the values in the **Type** column to uppercase in the **cars_new** data.

Demo: Using Character Functions

4_2 - Demo - Using Character Functions.pdf
(https://clemson.instructure.com/courses/237270/files/23073942/download?wrap=1)
(https://clemson.instructure.com/courses/237270/files/23073942/download?download_frd=1)

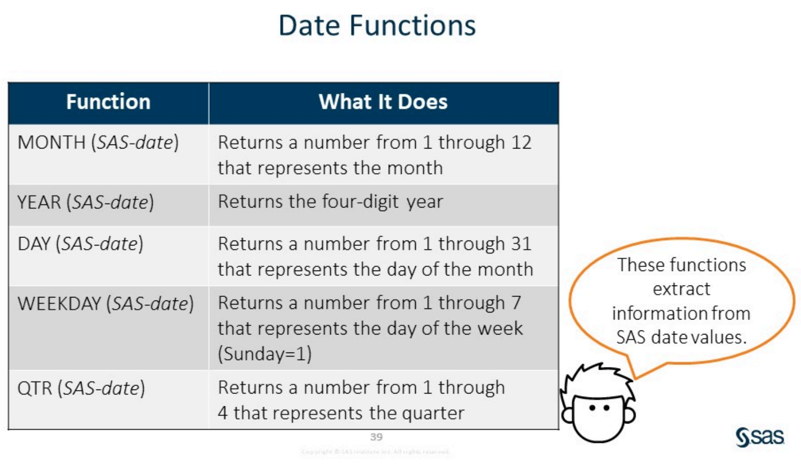
Activity 4.06

Open **p104a06.sas** from the **activities** folder and perform the following tasks:

1. Add a WHERE statement that uses the SUBSTR function to include rows where the second letter of **Basin** is *P* (Pacific ocean storms).
2. Run the program and view the log and data. How many storms were in the Pacific basin?

Click here for Solution.

Date Functions



Date Functions

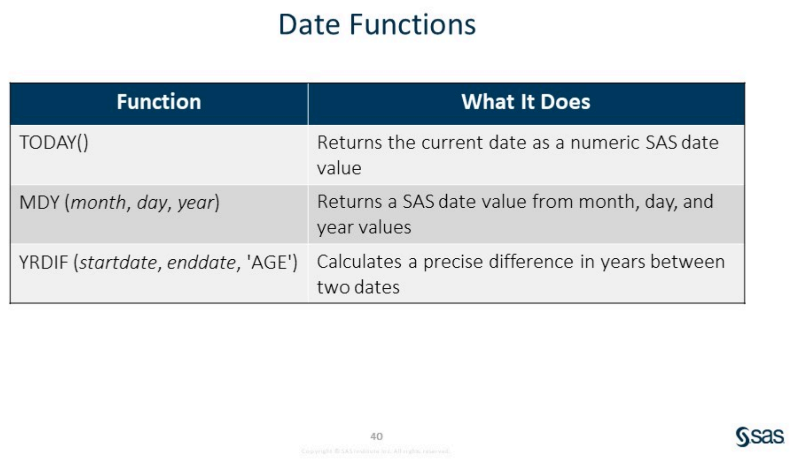
Function	What It Does
MONTH (*SAS-date*)	Returns a number from 1 through 12 that represents the month
YEAR (*SAS-date*)	Returns the four-digit year
DAY (*SAS-date*)	Returns a number from 1 through 31 that represents the day of the month
WEEKDAY (*SAS-date*)	Returns a number from 1 through 7 that represents the day of the week (Sunday=1)
QTR (*SAS-date*)	Returns a number from 1 through 4 that represents the quarter

These functions extract information from SAS date values.

39

SAS date functions are incredibly helpful for creating and manipulating SAS dates. Let’s take a look at a few of them.

These functions extract information from SAS date columns or values. The MONTH function has one argument: a SAS date value. The function then returns a number from 1 to 12 that represents the month. Similar functions include YEAR, DAY, WEEKDAY, and QTR.



Date Functions

Function	What It Does
TODAY()	Returns the current date as a numeric SAS date value
MDY (*month*, *day*, *year*)	Returns a SAS date value from month, day, and year values
YRDIF (*startdate*, *enddate*, 'AGE')	Calculates a precise difference in years between two dates

40

Other date functions enable to you build or create date values. One of my favorites is the dynamic TODAY function that always returns today’s date in numeric form. In other words, it returns the number of days since January 1, 1960. You must use the parentheses after TODAY, but you don't have to specify an argument because the date is retrieved from the system clock.

Another function that creates a SAS date value is MDY. The MDY function requires three arguments – the numeric month, day, and year – and it returns the corresponding SAS date value.

Another favorite of mine is the YRDIF function. It enables us to calculate a precise age between two dates.

Demo: Using Date Functions

4_2 - Demo - Using Date Functions.pdf

(https://clemson.instructure.com/courses/237270/files/23073935/download?wrap=1)

(https://clemson.instructure.com/courses/237270/files/23073935/download?download_frd=1)