Beyond Labels: Mastering PROC FORMAT for Dynamic, Scalable SAS Programming

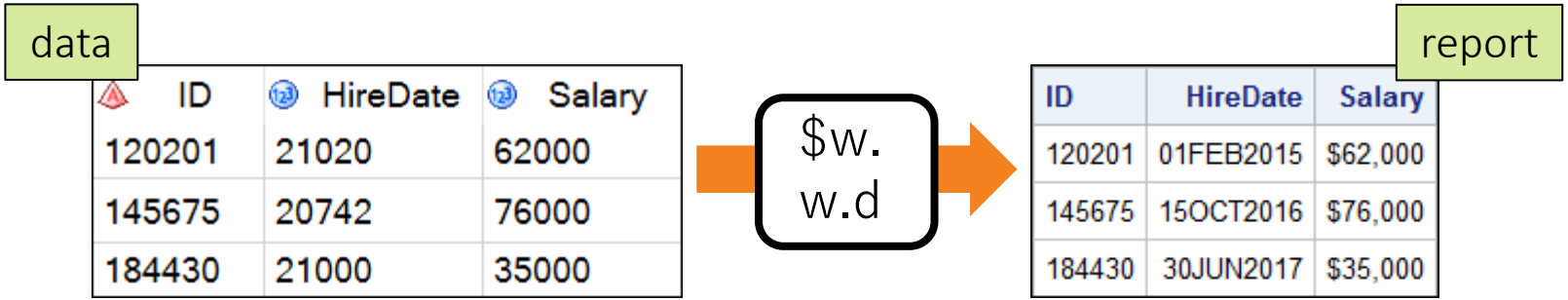
Michael Salé, Stonehill College & BASUG

Agenda

Basics of the FORMAT Statement
Common Format Types
Creating and Using Custom Formats
Creating Custom Formats from Tables
Storing Custom Formats
Using the PICTURE Statement in Custom Formats
Questions and Answers

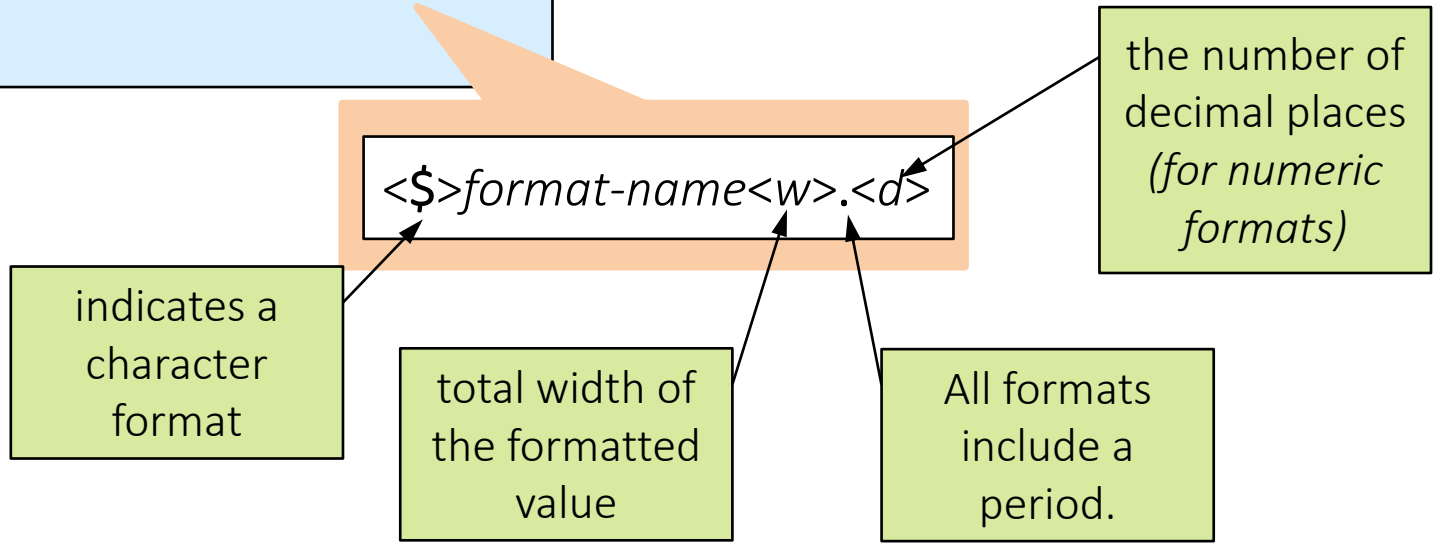
Basics of the FORMAT Statement

Formatting Data Values



Formatting Data Values in Results

```
PROC PRINT DATA=input-table;
      FORMAT col-name(s)  format;
RUN;
```



Formatting Multiple Columns

```
PROC PRINT DATA=work.class_birthdate;
    FORMAT Height Weight 3. Birthdate date9.;
RUN;
```

Name	Sex	Age	Height	Weight	Birthdate
Alfred	M	14	69	112.5	16370
Alice	F	13	56.5	84	16756
Barbara	F	13	65.3	98	16451
Carol	F	14	62.8	102.5	16256

Name	Sex	Age	Height	Weight	Birthdate
Alfred	M	14	69	113	26OCT2004
Alice	F	13	57	84	16NOV2005
Barbara	F	13	65	98	15JAN2005
Carol	F	14	63	103	04JUL2004

Remember! *These formats impact the way the values are displayed, but they do not change the raw data values themselves.*

Formatting Permanence

Remember that using the format statement in a PROC step such as

- PROC PRINT
- PROC MEANS
- PROC UNIVARIATE

only formats the variable temporarily (for the duration of the program run).

Formatting in a DATA step permanently applies the format to the variable.

Common Format Types

for Numeric and Date Values

Common Formats for Numeric Values

Format Name	Example Value	Format Applied	Formatted Value
w.d	12345.67	5.	12346
w.d	12345.67	8.1	12345.7
COMMA*w.d*	12345.67	COMMA8.1	12,345.7
DOLLAR*w.d*	12345.67	DOLLAR10.2	$12,345.67
DOLLAR*w.d*	12345.67	DOLLAR10.	$12,346
YEN*w.d*	12345.67	YEN7.	¥12,346
EUROX*w.d*	12345.67	EUROX10.2	€12.345,67

Common Formats for Date Values

Value	Format	Formatted Value
23919	DATE7.	27JUN25
23919	DATE9.	27JUN2025
23919	MMDDYY10.	05/27/2025
23919	DDMMYY8.	27/06/25
23919	MONYY7.	JUN2025
23919	MONNAME.	June
23919	WEEKDATE.	Friday, June 27, 2025

Why Format Widths Matter

What happens when a format width is too small?

```
PROC PRINT DATA=work.storm_damage;
    FORMAT Cost dollar16. Date mmddyy10.;
RUN;
```

Obs	Cost	Date
1	$161,300,000,000	08/25/2005
2	$125,000,000,000	08/25/2017
3	$90,000,000,000	09/19/2017
4	$70,900,000,000	10/30/2012
5	$50,000,000,000	09/06/2017

```
PROC PRINT DATA=work.storm_damage;
    FORMAT Cost dollar14. Date mmddyy6.;
RUN;
```

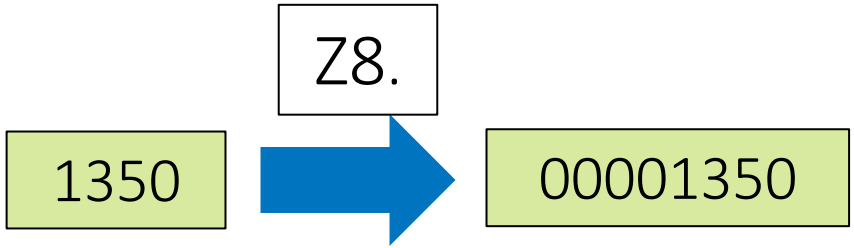
Obs	Cost	Date
1	$161300000000	082505
2	$125000000000	082517
3	$90000000000	091917
4	$70900000000	103012
5	$50000000000	090617

Notice that SAS tries to preserve the data at the expense of your format.

Pop Quiz!

What does the Z8. format do?

Pop Quiz – Answer



Creating and Using Custom Formats

The Need for Custom Formats

Name	Registration	Height
Alfred	C	69
Alice	C	56.5
Barbara	I	65.3
Carol	C	62.8

Name	Registration	Height
Alfred	Complete	Above Average
Alice	Complete	Below Average
Barbara	Incomplete	Above Average
Carol	Complete	Above Average

```
FORMAT Registration ? Height ?;
```

FORMAT Procedure

```
PROC FORMAT;
      VALUE format-name  value-or-range-1 = 'formatted-value'
                         value-or-range-2 = 'formatted-value'
                         . . . ;
RUN;
```

- The name can be up to 32 characters in length.
- Character formats must begin with a $ followed by a letter or underscore.
- Numeric formats must begin with a letter or underscore.
- The name cannot end in a number or match an existing SAS format.

FORMAT Procedure

```
PROC FORMAT;
      VALUE format-name  value-or-range-1 = 'formatted-value'
                         value-or-range-2 = 'formatted-value'
                         . . . ;
RUN;
```

individual value or range of values that you want to format

format that you want to apply to the individual value or range of values

Creating and Using Custom Formats

create format

```
PROC FORMAT;
     VALUE $regfmt 'C'='Complete'
                   'I'='Incomplete';
RUN;
```

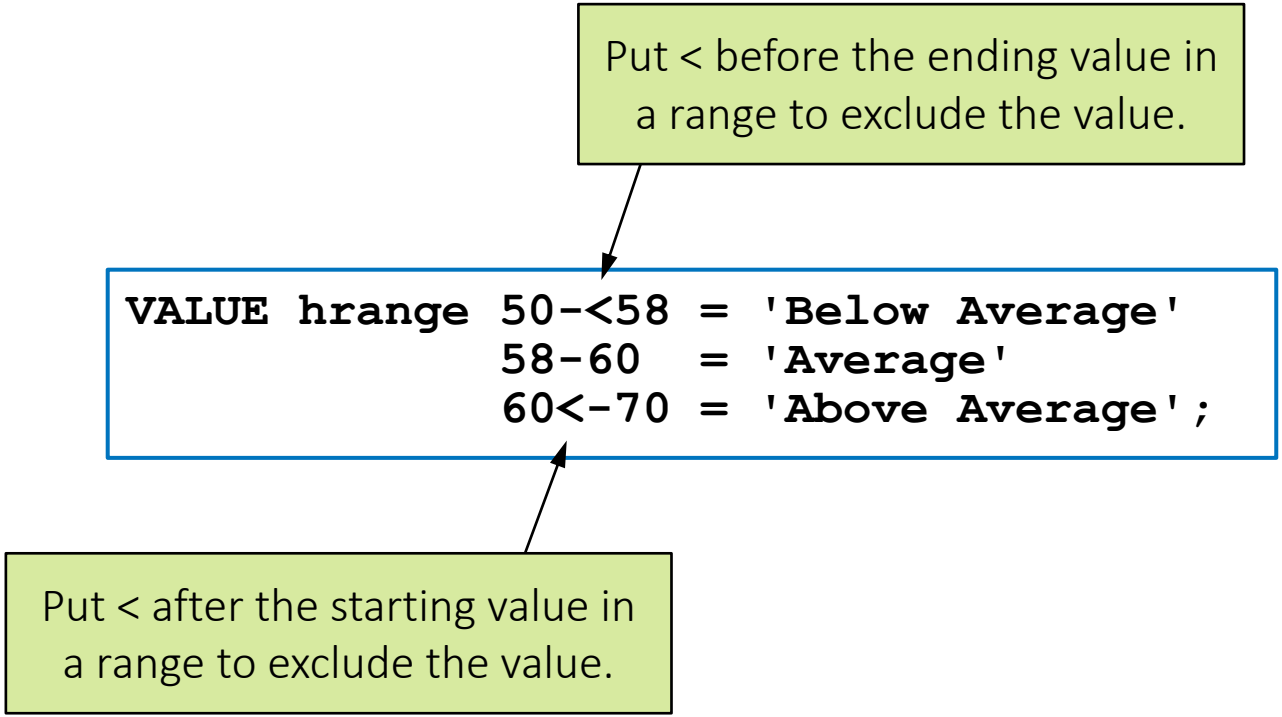
no period in format name

apply format

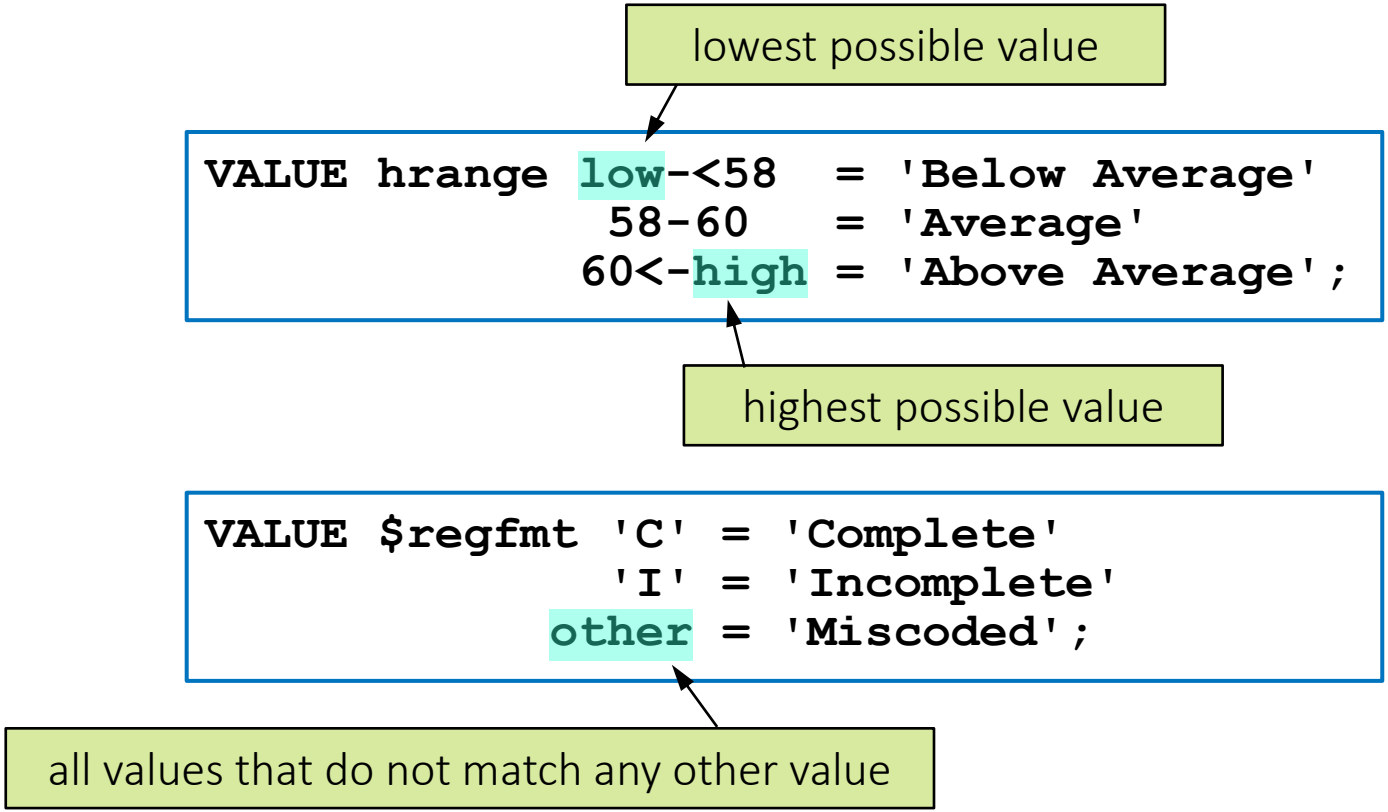
```
PROC PRINT DATA=work.class_birthdate;
     FORMAT Registration $regfmt.;
RUN;
```

period in format name

Defining a Continuous Range



Using Keywords



p204d01

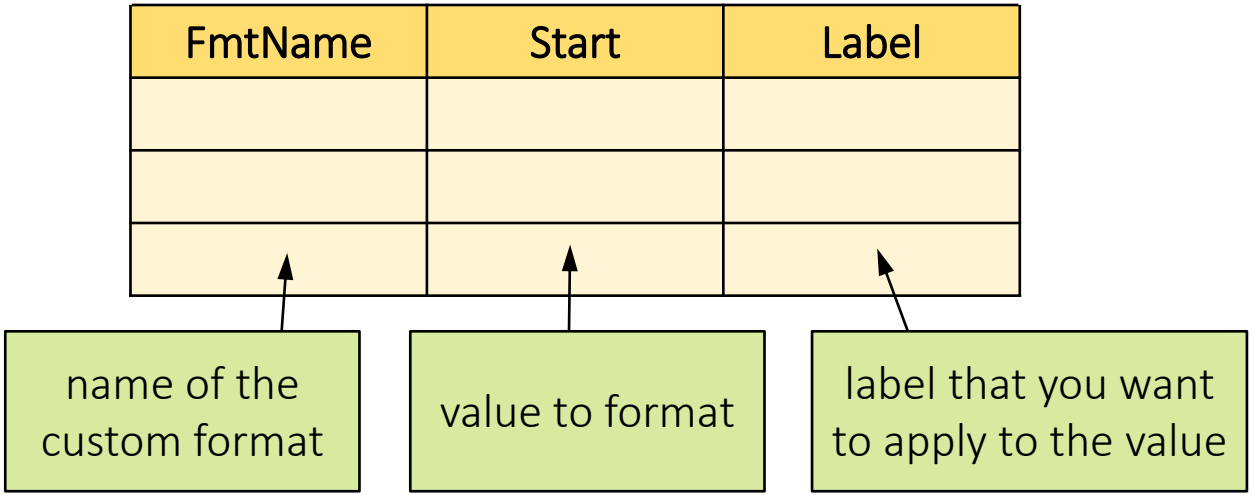
Creating Custom Formats from Tables

Creating Custom Formats from Tables

Sub_Basin	SubBasin_Name
AS	Arabian Sea
BB	Bay of Bengal
EA	Eastern Australia
WA	Western Australia
CP	Central Pacific
CS	Caribbean Sea
GM	Gulf of Mexico
MM	Missing

```
PROC FORMAT;
     VALUE $sbfmt
               'AS'='Arabian Sea'
               'BB'='Bay of Bengal'
               'EA'='Eastern Australia'
               'WA'='Western Australia'
               'CP'='Central Pacific'
               'CS'='Caribbean Sea'
               'GM'='Gulf of Mexico'
               'MM'='Missing';
RUN;
```

Rules for the Input Table



The input table must have at least these three columns.

Rules for the Input Table

Sub_Basin	SubBasin_Name
AS	Arabian Sea
BB	Bay of Bengal
EA	Eastern Australia
WA	Western Australia
CP	Central Pacific
CS	Caribbean Sea
GM	Gulf of Mexico
MM	Missing

FmtName	Start	Label
$sbfmt	AS	Arabian Sea
$sbfmt	BB	Bay of Bengal
$sbfmt	EA	Eastern Australia
$sbfmt	WA	Western Australia
$sbfmt	CP	Central Pacific
$sbfmt	CS	Caribbean Sea
$sbfmt	GM	Gulf of Mexico
$sbfmt	MM	Missing

```
DATA work.sbdata;
     RETAIN FmtName '$sbfmt';
     SET pg2.storm_subbasincodes(RENAME=(Sub_Basin=Start
                                         SubBasin_Name=Label));
     KEEP Start Label FmtName;
RUN;
```


p204d02

CNTLIN= Option

```
PROC FORMAT CNTLIN=work.sbdata;
RUN;
```

Use the CNTLIN= option to specify a table for building a format.

Storing Custom Formats

Using FMTLIB to Output All Custom Formats

```
PROC FORMAT FMTLIB LIBRARY=work;
RUN;
```

```
|       FORMAT NAME: $SBFMT   LENGTH:   17   NUMBER OF VALUES:    8        |
|   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH:  17  FUZZ:        0  |
|-------------------------------------------------------------------------|
|START            |END              |LABEL  (VER. V7|V8   19JUL2018:10:57:40)|
|-----------------+-----------------+-----------------------------------|
|AS               |AS               |Arabian S
|BB               |BB               |Bay of Be
|CP               |CP               |Central P
|CS               |CS               |Caribbean
|EA               |EA               |Eastern A
|GM               |GM               |Gulf of M
|MM               |MM               |Missing
|WA               |WA               |Western A
```

```
|       FORMAT NAME: CATFMT   LENGTH:   11   NUMBER OF VALUES:    6        |
|   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH:  11  FUZZ: STD       |
|-------------------------------------------------------------------------|
|START            |END              |LABEL  (VER. V7|V8   19JUL2018:10:57:40)|
|-----------------+-----------------+-----------------------------------|
|LOW              |               63|No Category                        |
|               64|               82|Category 1                         |
|               83|               95|Category 2                         |
|               96|              112|Category 3                         |
|              113|              136|Category 4                         |
|              137|HIGH             |Category 5                         |
```

Location of Custom Formats

```
PROC FORMAT;
    VALUE $sttype
          'TS'='Tropical Storm'
          'SS'='Subtropical Storm'
          'ET'='Extratropical Storm'
          'DS'='Disturbance'
          'NR'='Not Reported';
RUN;
```

By default, custom formats are stored in the WORK library in a catalog named FORMATS.

These formats are deleted when you end your SAS session.

Creating Permanent Custom Formats

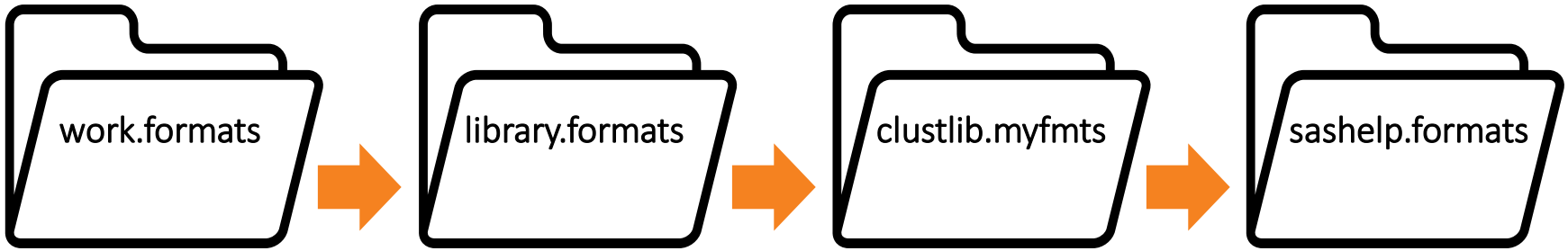
```
PROC FORMAT LIBRARY=clustlib.myfmts;
```

Use the LIBRARY= option to save formats in a permanent location.

You can specify the library and a catalog name, such as CLUSTLIB.MYFMTS. If you specify just the library, the default catalog FORMATS will be used.

Searching for Custom Formats

```
OPTIONS FMTSEARCH=(clustlib.myfmts sashelp);
```



Using the PICTURE Statement in a Custom Format

What is the PICTURE Statement?

- The PICTURE statement is used in PROC FORMAT to create **custom numeric display formats**.

- Unlike VALUE formats, which substitute values with labels, PICTURE formats control **how a number is presented visually** — including things like leading zeros, commas, currency symbols, percentage signs, or even phone number-style formatting.

- Unlike VALUE formats, which replace values or ranges with text labels, PICTURE formats preserve the numeric structure but change how it's **displayed**.

When to Use the PICTURE Statement

- **Only within PROC FORMAT** to define formats
- **Only for numeric variables**
- When you want **template-level control** over how numbers appear in output
- Can be applied in:
  - PROC PRINT
  - DATA steps
  - PROC REPORT, PROC TABULATE, and more

PICTURE Statement Details

The PICTURE statement defines a **numeric format template** by using characters like:

- 9 – Placeholder for digits
- 0 – Forces display of leading/trailing zeros
- . – Decimal point
- , – Comma separator
- %, $ – Literal characters
- Z – Suppresses leading zeros
- < and > – Used to define value ranges
- MULT= – Multiplies the value before formatting (great for percentages)

PICTURE Statement Syntax Example

```
PROC FORMAT;
     PICTURE phonefmt
        low - high  = '000) 000-0000'
        (prefix='(');
RUN;
```

This special phone number format:

- Adds a left parathesis prefix
- Adds a hyphen
- Adds spacing between segments

Obs	PhoneNum
1	5085552134
2	8849983667
3	4441389945

Obs	PhoneNum
1	(508) 555-2134
2	(884) 998-3667
3	(444) 138-9945

Optional PICTURE Statement Keywords

- MULT=: multiply the input before formatting
- ROUND: round the number before formatting
- PREFIX='text' or SUFFIX='text': append/prepend strings
- NOEDIT: prevents literal characters in the template from being overwritten by digits

Questions and Answers