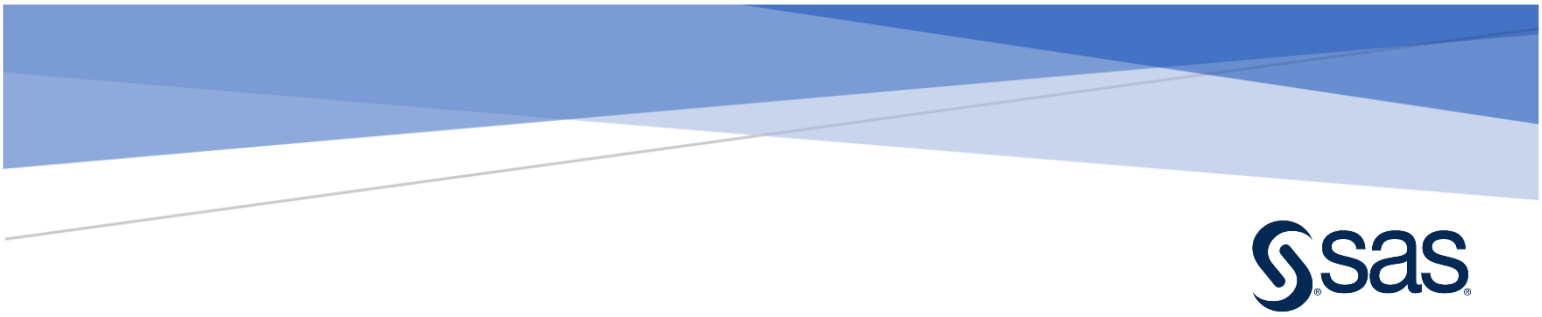

SAS® MACRO 1 CASE STUDY

Fill in the Blank Notes

SAS® Macro 1 Case Study Fill in the Blank Notes was developed by Carleigh Jo Crabtree.

SAS® Macro 1 Case Study Fill in the Blank Notes

Course code MC1CS, prepared date 24Oct2024.
MC1CS

Table of Contents

SAS Macro Facility

- SAS macro language interprets and manipulates _____ to generate SAS program code
- Macro code is sent to the macro processor
- 4 token types:
  - Name, number, special, literal
- Tokenization explains how SAS and the macro processor work together
- Macro _____- combination of % or & special token immediately followed by a name token

---

- Macro variable names and values are stored in _____ symbol table
  - Remain in global symbol table until SAS session ends
- To create macro variable:
  - **`%LET macroName= text;`**
- To use:
  - **`&macroName`**
  - Add formatting such as quotation marks around character values in _____
- Use a period to delimit macro variable names from text
  - **`Title "&Student.s Attendance";`**
- Use two periods between macro variable and _____ name
  - **`Proc print data= &lib..cars;`**

Macro Functions

- Macro functions manipulate text in a program rather than data _____
- Process text in the macro processor before code is compiled and executed
- Arguments of a macro function are not _____, if quotes are included, they are treated as part of the argument

---

- **`%SYSFUNC(function(arguments, <format>))`**
  - Enables _____ functions to be used by the macro processor
  - Optionally add a format to the result
- **`%SYSEVALF(expression, <conversion-type>)`**
  - Evaluate _____ expressions

- Returns a text value

- **%STR(character-string)**
  - Masks all _____ characters except % and & so they are regular text in macro processor
  - Add a % in front of single characters normally encountered in pairs
    - **%let title=%str(CJ%'s Report);**
- **%NRSTR(character-string)**
  - Masks all special characters including % and &
    - **%let dept=%nrstr(R&D);**

Using SQL to Create Macro Variables

- **PROC SQL;**
- **SELECT item1 <, item2, ….>**
  - **INTO :macroVar1 <, :macroVar2, …>**
  - **FROM clause**…
- **QUIT;**
- _____clause assigns values produced by the query to macro variables
  - TRIMMED removes spaces before values are stored in macro variable
  - INTO :mac1- :macn or INTO :mac1-
    - Creates a series of macro variables for distinct values of a column selected
  - SEPARARTED BY " " assigns multiple values to a _____ macro variable

---

Using the DATA Step to Create Macro Variables

- **CALL SYMPUTX(macroName, value);**
  - Creates macro variables and assigns values during _____ step execution
  - Leading and trailing blanks are removed from both arguments
  - Arguments can be literal strings, columns from the data, or expressions
  - Values of numeric columns are converted automatically to text
  - Use _____ from the input data to provide both macro variable names and values
  - Use functions or expressions to manipulate the CALL SYMPUTX arguments

---

Indirect References to Macro Variables

- Multiple _____ can be used to change the sequence in which macro variables are resolved
- Forward rescan rule
    - Multiple ampersands followed by a _____ token denote an indirect reference
    - &&nameToken resolves to &nameToken
    - &nameToken is resolved
    - Macro processor rescans the text until no more references can be resolved

---

Defining and Calling a Macro

- Macro _____-
    - Store text including:
        - Macro language statements or expressions
        - Complete or partial SAS program statements or program steps
    - Stored in catalogs in a SAS library
- Macro call executes the code store in macro definition
- Macro _____- alternative to %LET statements
    - Positional parameters-
        - In macro definition, names are in an ordered list
        - In macro call, values must be listed in same order
            - %ExPositional(a,b,c)
    - Keyword parameters-
        - Allow _____ values for parameters
        - Listed as name=value pairs, can be listed in any order
            - %ExKeyword(a=1, b=blue)

Macro Variable Scope

- When created inside a macro definition, macro variables can be global or local scope
    - _____: deleted at the end of the macro
    - Global: persists during SAS session
    - Use %LOCAL statement before any other reference to the macro variable to ensure it is created as local

Conditional Processing

- %IF %THEN statements and _____ groups can be used to execute code conditionally based on an expression
  - Can be used inside or outside a macro definition
- Inside a macro definition:
  - Use %IF %THEN to execute one statement
  - Use %DO groups to execute more than one statement conditionally
  - %ELSE %IF nested conditions are _____
  - %ELSE or %ELSE %DO is optional
- _____ a macro definition:
  - Must use %DO group
  - Cannot use %ELSE %IF nested conditions
  - %ELSE %DO is optional

- **`%return;`** terminates macro execution
- To enable the macro IN operator, add the mInOperator option after a _____ on the %macro statement
- %IF %THEN can conditionally execute statements within a step
- %IF %THEN can conditionally insert text within a statement

Iterative Processing

- %DO _____ are only valid within a macro definition
  - **`%DO index=start %TO stop <%BY increment>;`**
    - **`Text and macro statements`**
  - **`%END`**
    - Use when you know exactly how many times to execute the loop

---

Storing Macros

- Use _____ facility to reuse and share macro programs
  - Write macro definition and save program with the same name as the macro

- In each new SAS session, submit code to allow SAS to locate the SAS autocall macros and autocall macros you have defined
    - **`OPTIONS SASAUTOS=("path/autocall", SASAUTOS);`**
        - Autocall is a folder with personal .sas macro programs