K2 Analytics
Building Skills, Building Individuals

# SAS Macros

- Rajesh Jakhotia

21 Sep 2017

Earning is in Learning
- Rajesh Jakhotia

# About K2 Analytics

*At K2 Analytics, we believe that skill development is very important for the growth of an individual, which in turn leads to the growth of Society & Industry and ultimately the Nation as a whole. For this it is important that access to knowledge and skill development trainings should be made available easily and economically to every individual.*

**Our Vision:** *“To be the preferred partner for training and skill development”*

**Our Mission:** *“To provide training and skill development training to individuals, make them skilled & industry ready and create a pool of skilled resources readily available for the industry”*

*We have chosen Business Intelligence and Analytics as our focus area. With this endeavour we make this “**SAS Macros**” accessible to all those who wish to learn SAS. We hope it is of help to you. For any feedback / suggestion feel free to write back to us at ar.jakhotia@k2analytics.co.in*

*Welcome to Base SAS!!!*

# Welcome to SAS Macros

Earning is in Learning
- Rajesh Jakhotia

## Macros & Programming Structures

***Macros***

***Do Loop***
***IF-ELSE Condition***

***Macro Variables in Proc SQL***

Earning is in Learning
- Rajesh Jakhotia

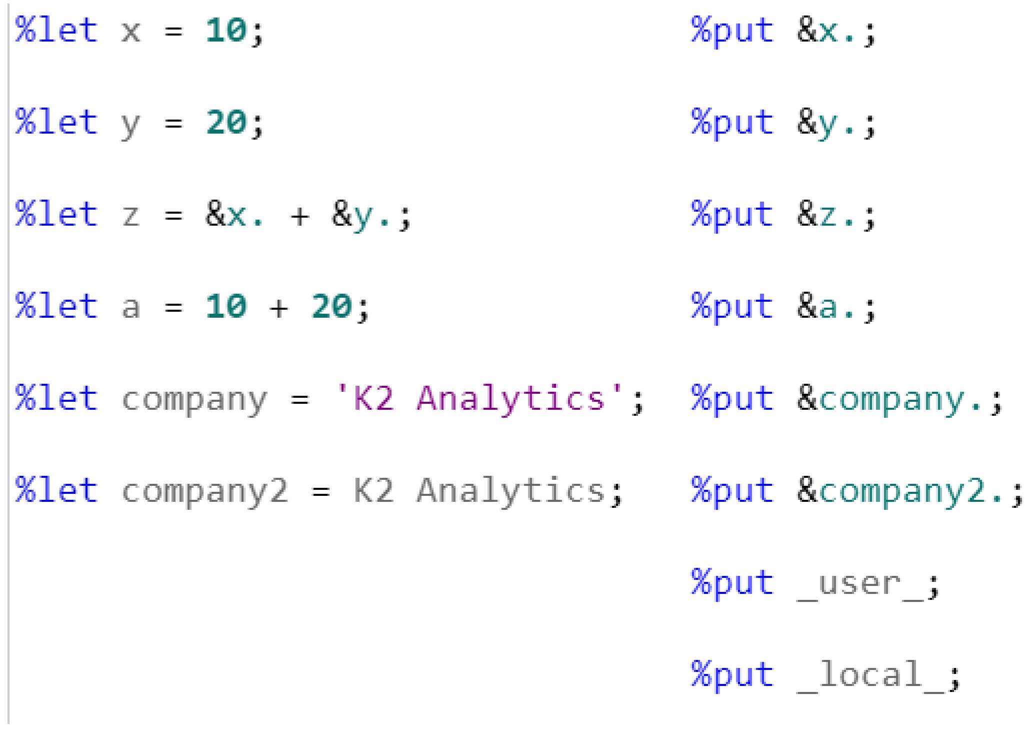
# Macros

- Macros helps make the code modular and structured
- It helps reduce the amount of regular SAS Code
- Brings versatility to SAS Code and make it more dynamic
- Two key building blocks of Macros
  - Macros (Macro Function)
  - Macro Variables

- Macro is reference by % and Macro Variable is referenced by &
- A macro can be considered as a larger piece of code that can contain data step, proc step and macro conditional statements
- Writing Macros is like meta-programming, i.e., you are writing a program that writes a program

# Macros basic example

- %let macro-variable-name; /* creates a macro variable */
- %put &macro-variable-name; /* displays macro variable value */



```
%let x = 10;                      %put &x.;

%let y = 20;                      %put &y.;

%let z = &x. + &y.;               %put &z.;

%let a = 10 + 20;                 %put &a.;

%let company = 'K2 Analytics';   %put &company.;

%let company2 = K2 Analytics;    %put &company2.;

                                  %put _user_;

                                  %put _local_;
```

| %put resolves to |
|---|
| 10 |
| 20 |
| 10 + 20 |
| 10 + 20 |
| 'K2 Analytics' |
| K2 Analytics |
| Displays all user defined macro variables |
| Displays all local macro variables |

<u>Note:</u>

Macro Variable does not have data types

# & and && for referencing macro variables

```
%let A = Duck;

%let Duck = Duckland;
```

Declaration of Macro Variables **A** and **Duck**

```
%put &A.;
```

Referencing the macro variable **A**

```
%put &&A.;
```

**Note:** SAS Automatically resolves double && in a row to single &

```
%put &&&A.;
```

**&&A.** is resolved first.

It becomes **&Duck**

**&Duck** is then resolved and you get the value **Duckland**

K2 Analytics
Building Skills, Building Individuals

## & and &&… contd

```
%let A = Duck;

%let DDuck = DonaldDuck;
```

Declaration of Macro Variables **A** and **DDuck**

```
%put &D&A.;
```

**&A. and &D** is attempted to be resolved simultaneously

If macro variable **D** is not defined you will get an error for it

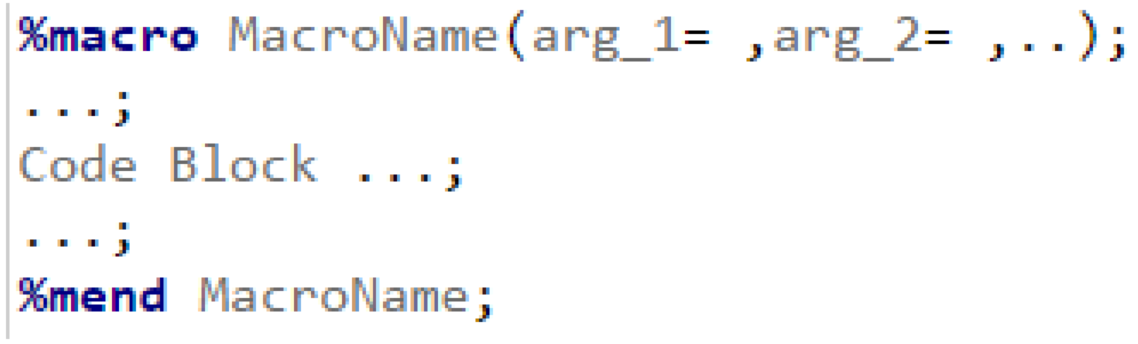
```
%put &&D&A.;
```

here **&A.** will be resolved first and it will get replaced by its value **Duck**

After that **&&DDuck** will get resolved and you get the output as **DonaldDuck**

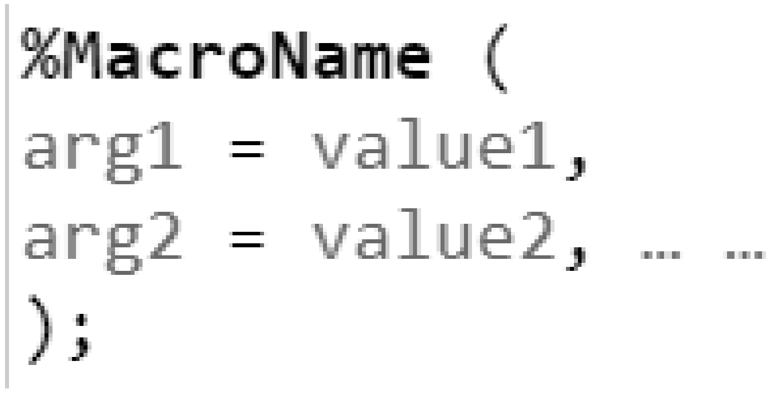
## Some more info on Macros…

- Macro Global & Local Variables
  - A macro variable’s scope is local if it is defined inside a macro
  - A macro variable’s scope is global if it is defined as part of open-code, which is everything outside a macro



```
%macro MacroName(arg_1= ,arg_2= ,..);
...;
Code Block ...;
...;
%mend MacroName;
```

- Two pitfalls to be avoided
  - Trying to use a local macro variable outside its own function
  - Creating global and local macro variable with same name



```
%MacroName (
arg1 = value1,
arg2 = value2, … …
);
```

- Use of single quote and double quote
  - Macro processor does not check for macros inside single quote
  - Use double quote for quoted strings that contain macro variable

## Scenario for Macro usage

Suppose we have multiple file having the same structure to be imported.

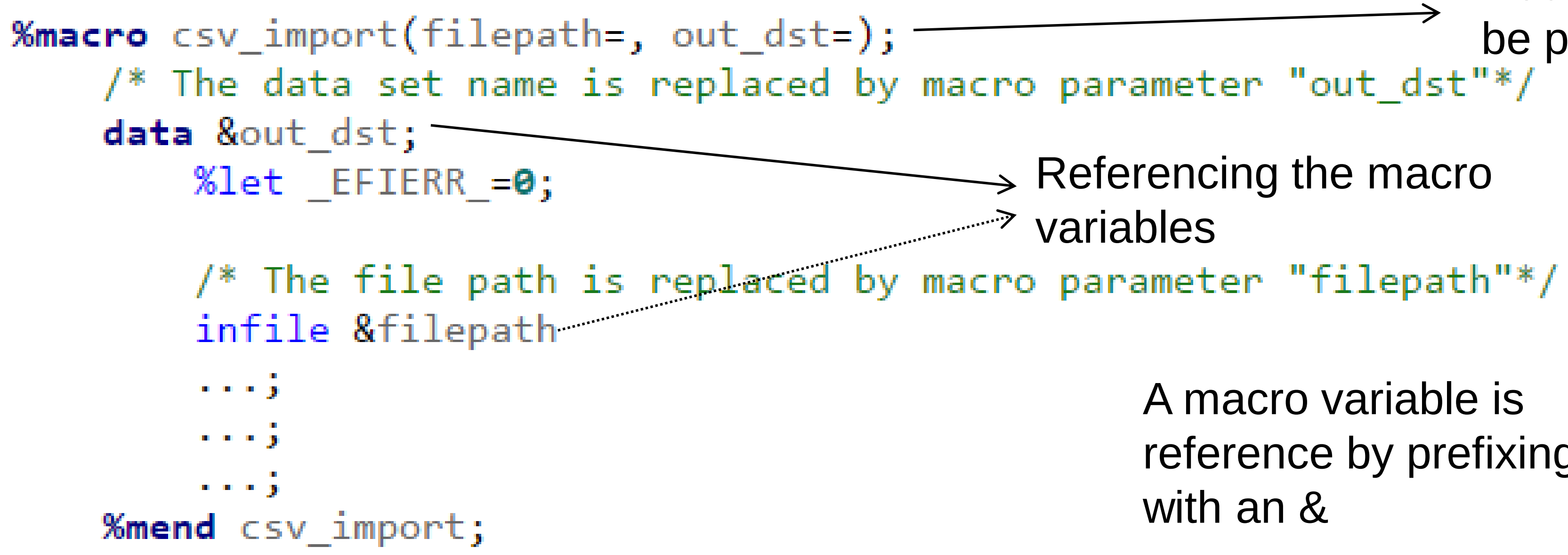
How can we import the files?

1. Import each file using Import Wizard one at a time

2. Write import code; copy-paste the import code along with replacing the file path and output dataset name

3. Use macros and looping structures to automate the import process

# Indicate macro code…



```
%macro csv_import(filepath=, out_dst=);
    /* The data set name is replaced by macro parameter "out_dst"*/
    data &out_dst;
        %let _EFIERR_=0;

        /* The file path is replaced by macro parameter "filepath"*/
        infile &filepath
        ...;
        ...;
        ...;
    %mend csv_import;
```

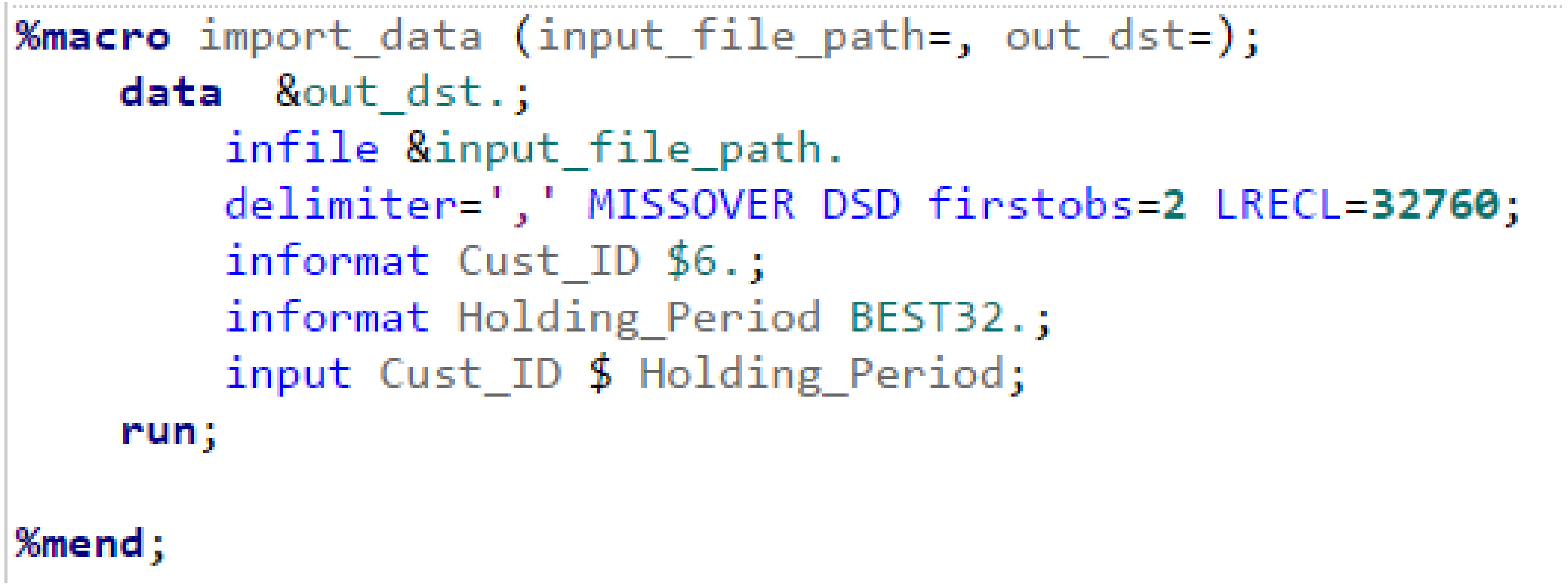
Declaring the arguments to be passed to the macro

Referencing the macro variables

A macro variable is reference by prefixing it with an &

The dot . after the macro variable is not mandatory but is a good practice

## Writing Macro to Import


```
%macro import_data (input_file_path=, out_dst=);
    data  &out_dst.;
        infile &input_file_path.
        delimiter=',' MISSOVER DSD firstobs=2 LRECL=32760;
        informat Cust_ID $6.;
        informat Holding_Period BEST32.;
        input Cust_ID $ Holding_Period;
    run;

%mend;
```

/* The above code is for importing the 4 sample csv files LR_HP_part1 to LR_HP_part4 */

## Calling the macro

```
%MarcoName(
arg1=Value1,
arg2=Value2, ..
);
```

Macro is called by % followed by macro name and then passing the arguments Value in round brackets () and finally a ;

### Code invoking the macro created in previous slide

```
%import_data (input_file_path = '~\LR_HP_part1.csv', out_dst = LR_HP_Part1);
%import_data (input_file_path = '~\LR_HP_part2.csv', out_dst = LR_HP_Part2);
%import_data (input_file_path = '~\LR_HP_part3.csv', out_dst = LR_HP_Part3);
%import_data (input_file_path = '~\LR_HP_part4.csv', out_dst = LR_HP_Part4);
```

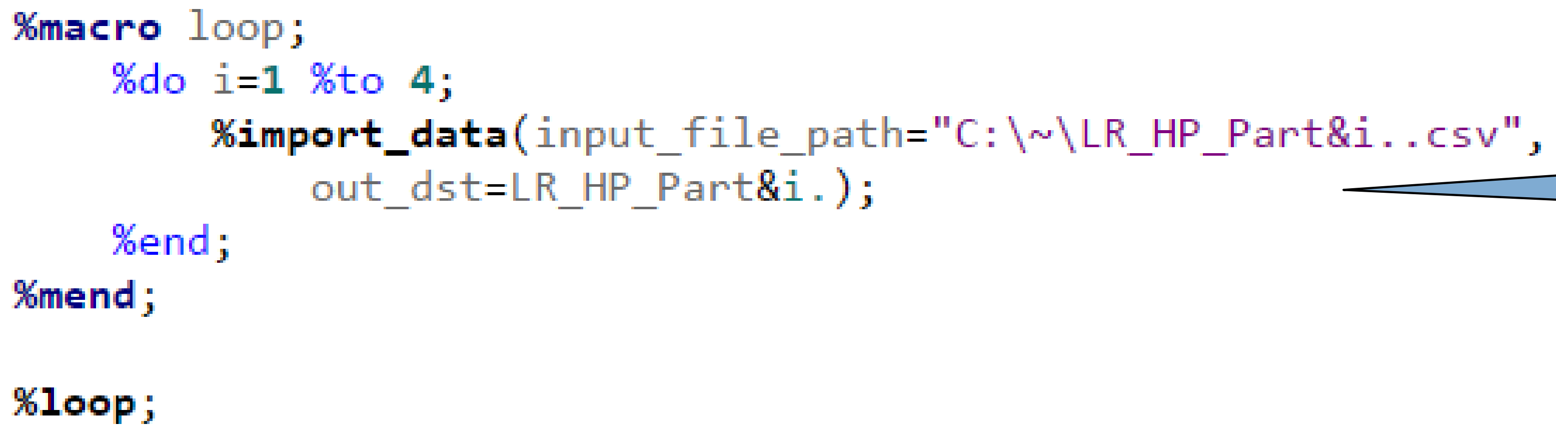
## DO loop in SAS

```
DO variable = start

<TO stop>

<BY increment>  /* default increment is by 1 */

<WHILE (expression) | UNTIL (expression)>

….

….

END;
```

***Note:***

***When coding for loops in SAS, one thing to remember is all of the parts of it are optional***

## Do Loop

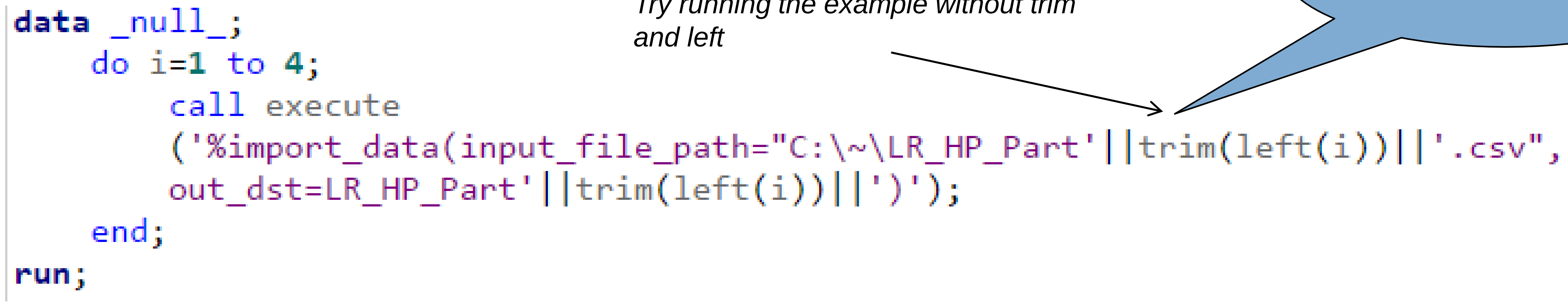


```
%macro loop;
    %do i=1 %to 4;
        %import_data(input_file_path="C:\~\LR_HP_Part&i..csv",
            out_dst=LR_HP_Part&i.);
    %end;
%mend;

%loop;
```

**Do Loop call in Macro**

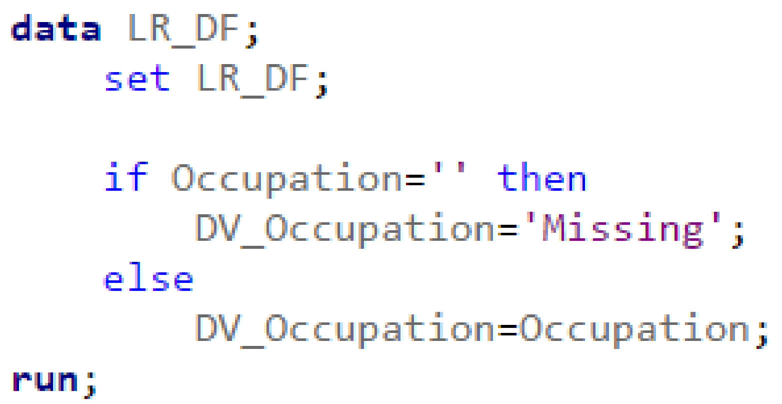
*Note the usage of trim & left…*
*Try running the example without trim and left*

**Do Loop call in Data Step**



```
data _null_;
    do i=1 to 4;
        call execute
        ('%import_data(input_file_path="C:\~\LR_HP_Part'||trim(left(i))||'.csv",
        out_dst=LR_HP_Part'||trim(left(i))||')');
    end;
run;
```

# IF-Else condition usage



```
data LR_DF;
    set LR_DF;

    if Occupation='' then
        DV_Occupation='Missing';
    else
        DV_Occupation=Occupation;
run;
```

- If multiple line of code are to be executed after IF – ELSE condition then put it between

  DO;

  END;

## Options SYMBOLGEN

```
OPTIONS SYMBOLGEN;  /* use in Development Mode */
```

- SYMBOLGEN option prints the value of each macro variable as it gets resolved

```
OPTIONS NOSYMBOLGEN;  /* use in Production Mode */
```

- NOSYMBOLGEN suppresses the macro variable values from being printed in log each time they are resolved

- Try running the DO Loop as in previous e.g. with the above two options

## %PUT _USER_ & %SYMDEL

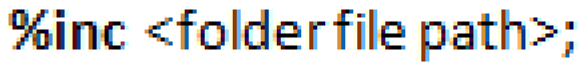
- Just in case you wish to delete some macro variable from global symbol table then use the in-built SAS Macro SYMDEL

```
%PUT  _USER_; /* Helps you see all the user defined macro variables */

%SYMDEL var_1  var_2 … ; /* Deletes the user defined macro variables */
```

## Macro Best Practice

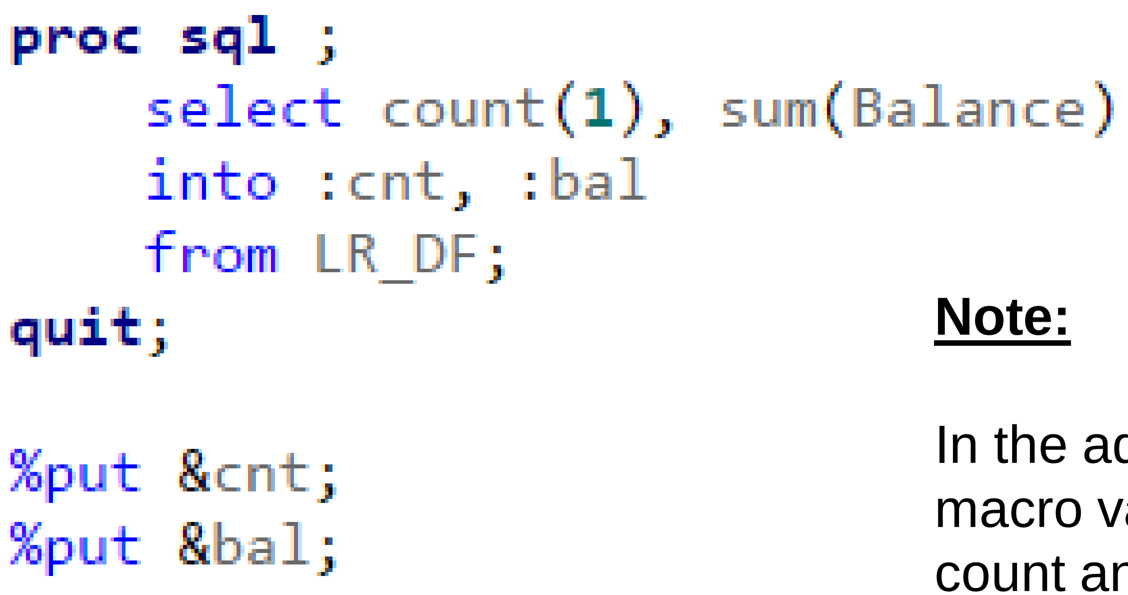
- In Production Mode
  - keep the Automated SAS Macro code separate files
  - and the Calling Macro code in different file

- Syntax to include code written in another SAS file



```
%inc <folder file path>;
```

# Creating Macro Variable in Proc SQL



```
proc sql ;
    select count(1), sum(Balance)
    into :cnt, :bal
    from LR_DF;
quit;

%put &cnt;
%put &bal;
```

**Note:**

In the adjacent e.g. we are creating global macro variables **‘cnt’** and **‘bal’** to which the count and sum values are assigned

## Exploratory Data Analysis

## using Macros, Procs, Looping

Earning is in Learning
- Rajesh Jakhotia

## Exercise

**Write Exploratory Data Analysis code that will do the following**

- Provide percentile distribution for all numeric variables in the dataset
- Provide frequency distribution for all categorical variables
- Provide min – max values for date type variables

## Steps to perform the EDA

- Declare a Macro with required input parameters
- Get all the fields in the dataset and their data types
- Create separate array structures for each data type
- Loop for numerical variables;
  - Looping
  - Get percentile distribution for each variable using Proc Mean
  - Save the output in one common dataset created for numeric variables
- Loop for categorical variables
  - Looping
  - Get Frequency distribution using Proc Freq
  - Save the output in a dataset created for categorical variables
- Loop to create SQL Min-Max query for Date Variables
  - Execute SQL Query and Store output in a dataset
- Export all the outputs to an excel

## Step 1 – Declare a macro

```
%macro DQ(dst=);


%mend
```

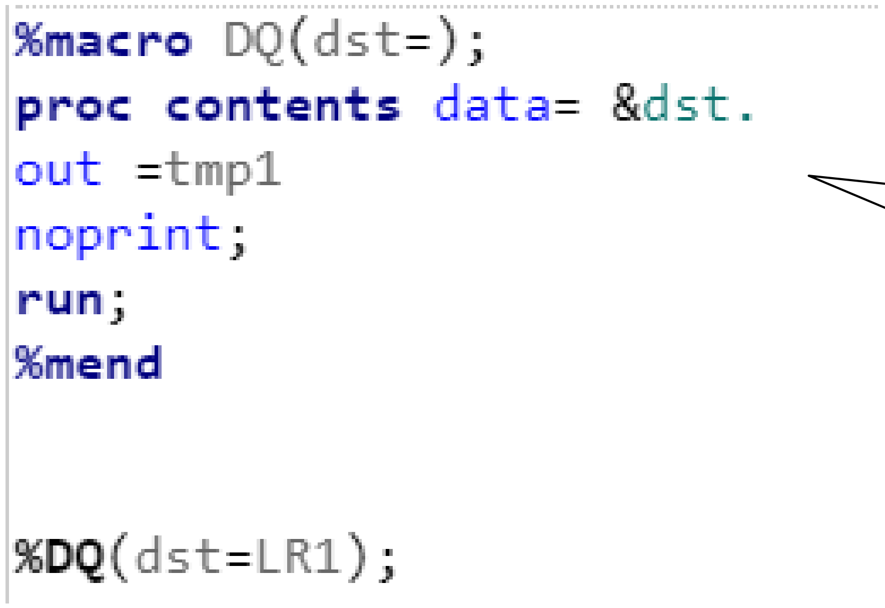
Declaring a Macro DQ
Inputs: DST – the dataset on which EDA is to be performed

- After Declaring Macro--- First step is to get the contents of dataset for which EDA is to be perfromed

```
proc contents data= <dataset name>
out =<outdataset name>
noprint;
run;
```

Remember PROC CONTENTS step with OUT Option

# Write the Proc Contents in Macro

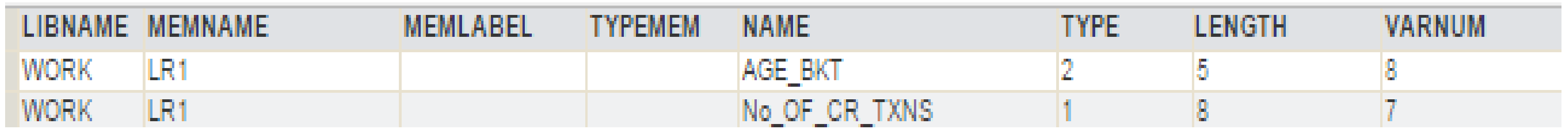


```
%macro DQ(dst=);
proc contents data= &dst.
out =tmp1
noprint;
run;
%mend


%DQ(dst=LR1);
```

The output of the PROC Contents is dumped into WORK.tmp1 dataset

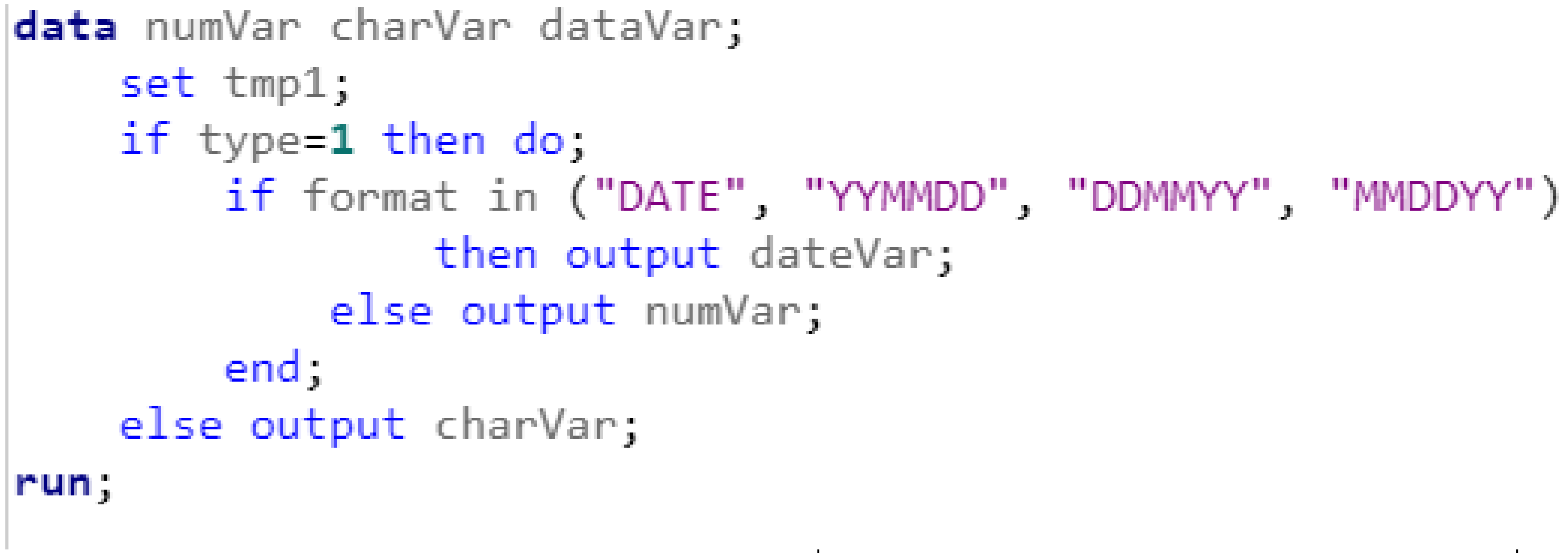
**Note:**
Eye-ball the tmp1 dataset

In PROC CONTENTS output TYPE = 1 indicates the variable is numeric and TYPE = 2 indicates the variable is categorical



| LIBNAME | MEMNAME | MEMLABEL | TYPEMEM | NAME | TYPE | LENGTH | VARNUM |
|---|---|---|---|---|---|---|---|
| WORK | LR1 | | | AGE_BKT | 2 | 5 | 8 |
| WORK | LR1 | | | No_OF_CR_TXNS | 1 | 8 | 7 |

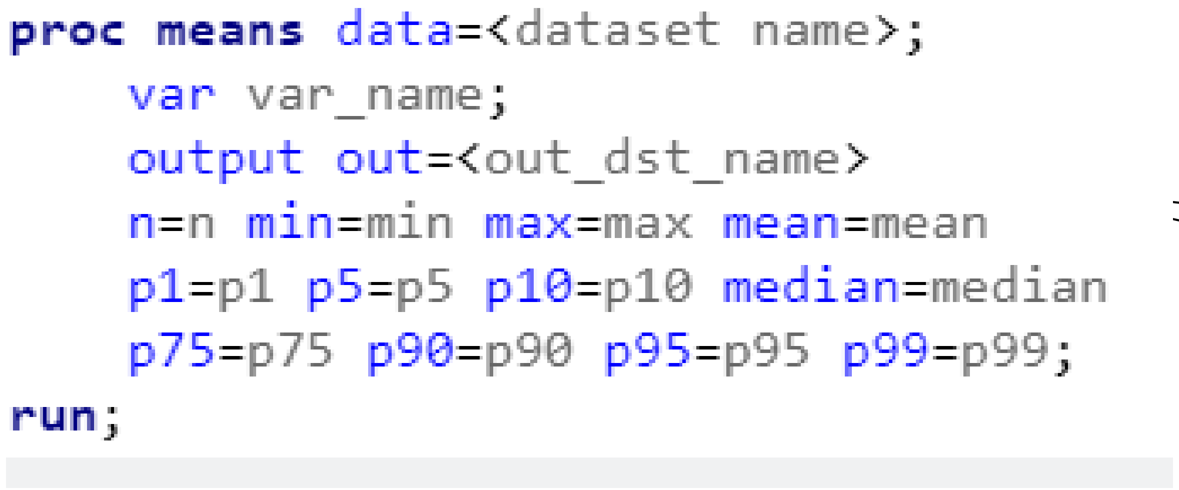
# Let us separate the Num, Char & Date variables



```
data numVar charVar dataVar;
    set tmp1;
    if type=1 then do;
        if format in ("DATE", "YYMMDD", "DDMMYY", "MMDDYY")
                then output dateVar;
            else output numVar;
        end;
    else output charVar;
run;
```

**Write this piece of code in DQ Macro**

## Getting percentile distribution for Numeric Variables



```
proc means data=<dataset name>;
    var var_name;
    output out=<out_dst_name>
    n=n min=min max=max mean=mean
    p1=p1 p5=p5 p10=p10 median=median
    p75=p75 p90=p90 p95=p95 p99=p99;
run;
```

Syntax for **Proc Means**

??? How do I loop through all the Numeric Variables to get their percentile distribution

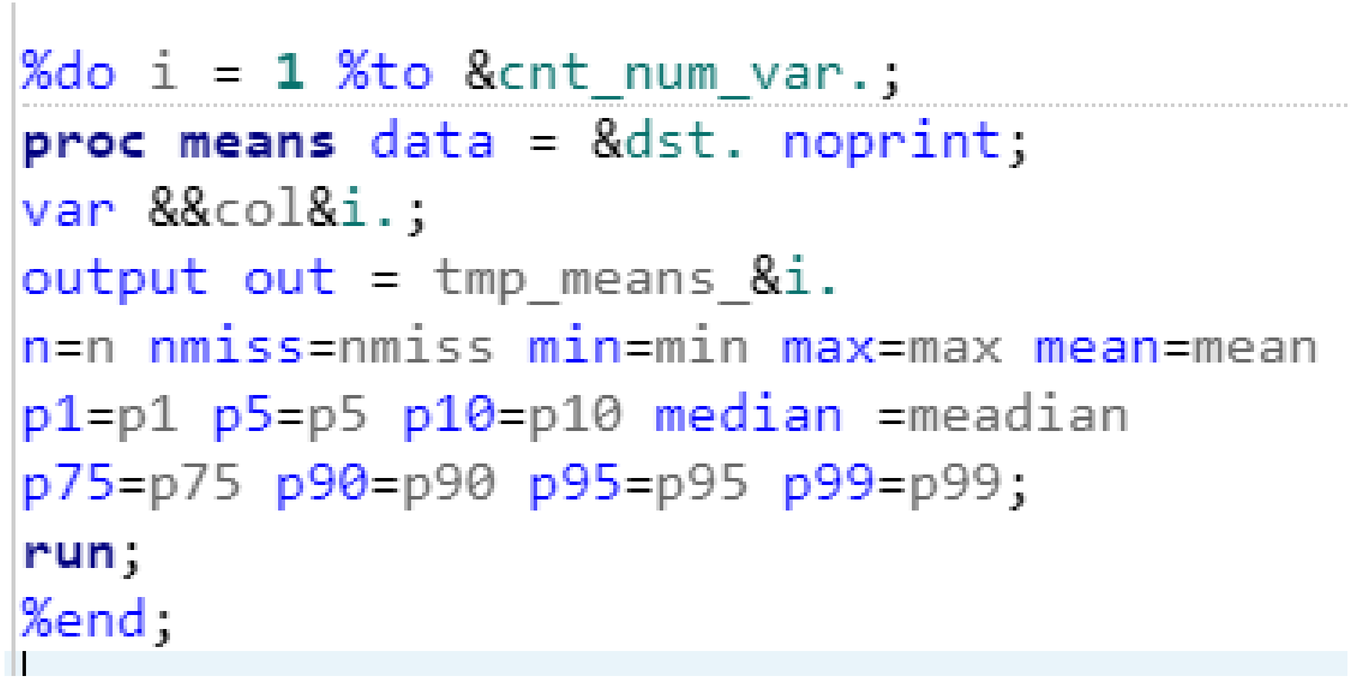
# Writing logic to loop through Num Var

```
%macro means_looping(dst=, num_var_dst=);
    proc sql ;
        select count(1) into :cnt_num_var from &num_var_dst.;
    quit;

    %let cnt_num_var = %trim(&cnt_num_var.);

    proc sql ;
        select name into :col1 - :col&cnt_num_var
        from &num_var_dst.;
    quit;

    %do i=1 %to &cnt_num_var.;
    /* proc means code to be written here */
    %end;

%mend
```

Note this. Try running this code without this statement

See next slide for the code

# Do Looping in the Means Macro

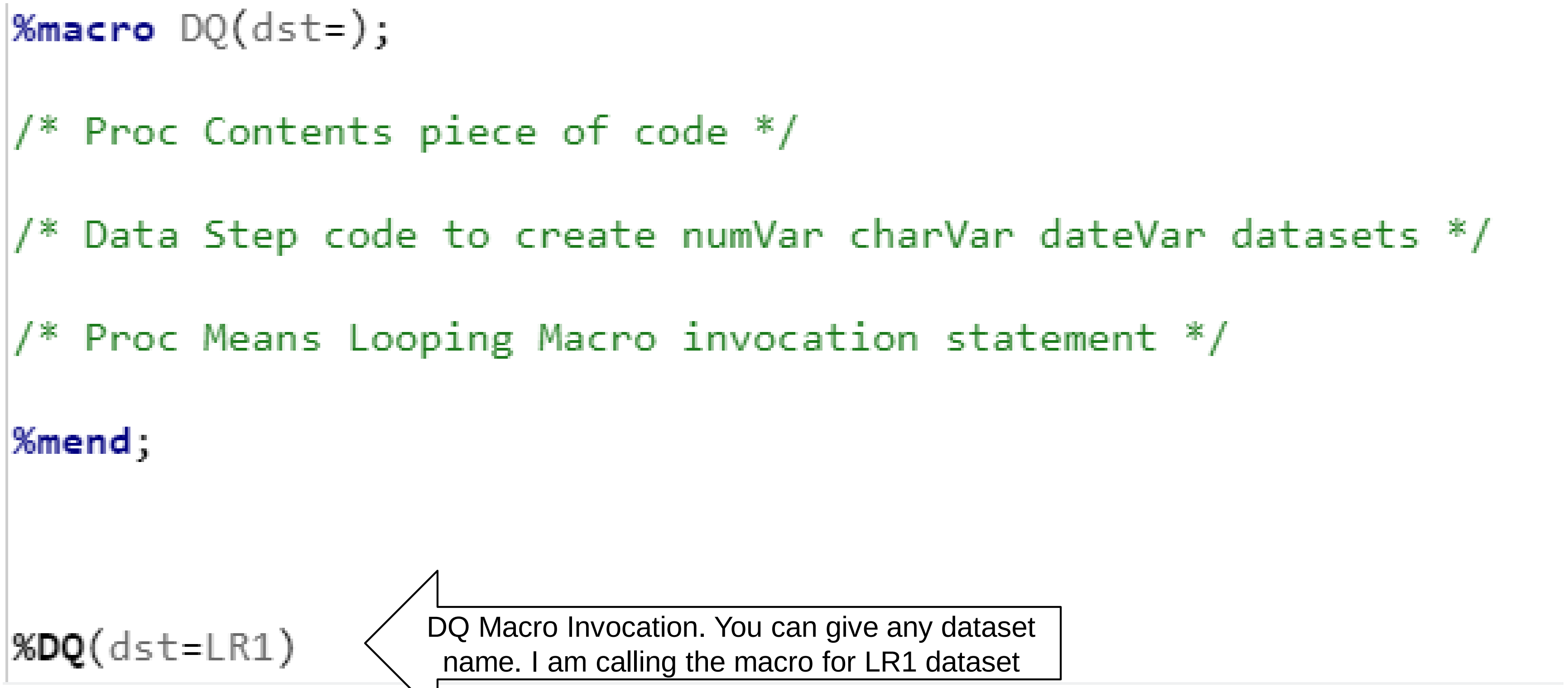


```
%do i = 1 %to &cnt_num_var.;
proc means data = &dst. noprint;
var &&col&i.;
output out = tmp_means_&i.
n=n nmiss=nmiss min=min max=max mean=mean
p1=p1 p5=p5 p10=p10 median =meadian
p75=p75 p90=p90 p95=p95 p99=p99;
run;
%end;
```

- Invoke the **MEANS_LOOPING** macro from **DQ** macro by writing the following statement in **DQ Macro**

%means_looping (dst = DQ, num_var_dst = numVar);

## Your DQ Macro would look something like this



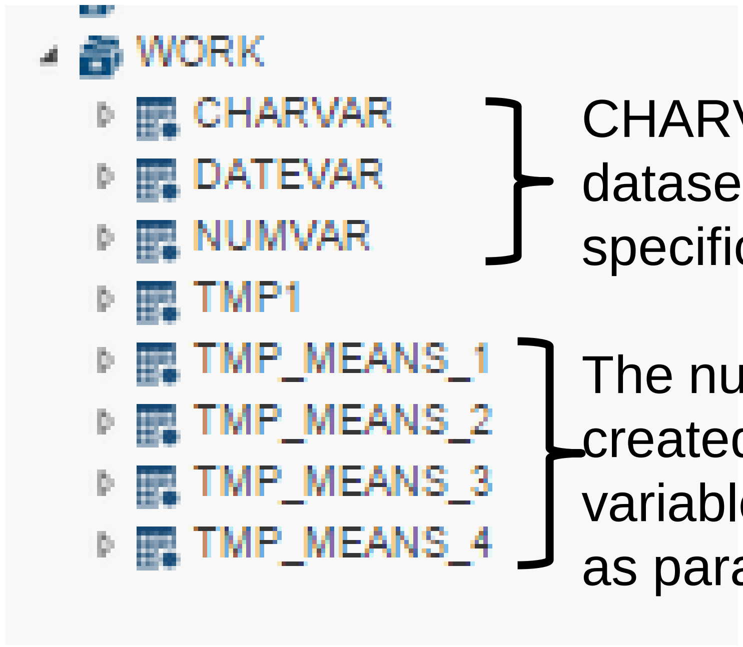
```
%macro DQ(dst=);

/* Proc Contents piece of code */

/* Data Step code to create numVar charVar dateVar datasets */

/* Proc Means Looping Macro invocation statement */

%mend;



%DQ(dst=LR1)
```

DQ Macro Invocation. You can give any dataset name. I am calling the macro for LR1 dataset

# DQ Macro Invocation output

```
%DQ (dst = LR1);
```

DQ Macro Invocation. You can give any dataset name. I am calling the macro for LR1 dataset



```
WORK
  CHARVAR
  DATEVAR
  NUMVAR
  TMP1
  TMP_MEANS_1
  TMP_MEANS_2
  TMP_MEANS_3
  TMP_MEANS_4
```

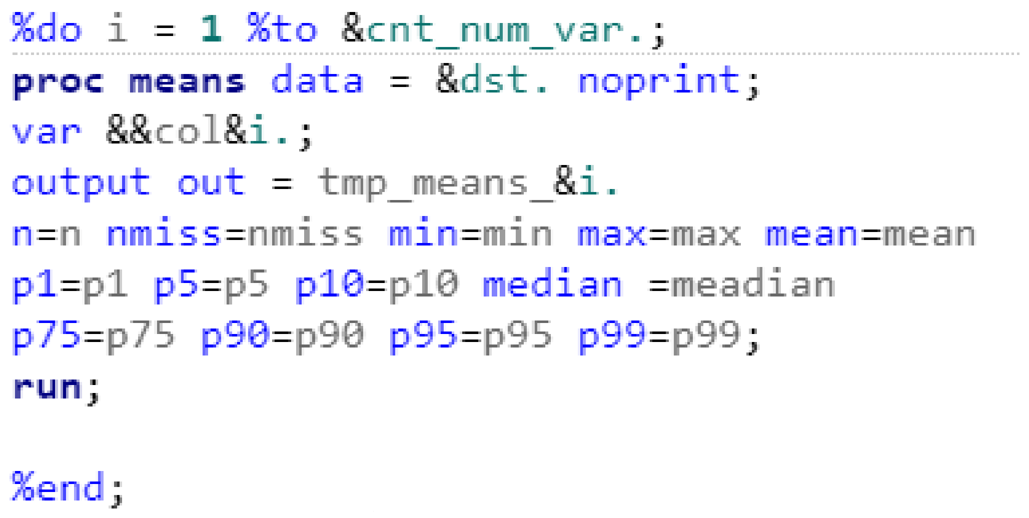
CHARVAR, DATEVAR, NUMVAR datasets are list of columns of the specific data types

The number of TMP_MEANS_[] dataset created will depend on the number of numeric variables present in the dataset you passed as parameter to DQ Macro

??? How do I append all the TMP_MEANS datasets into one single dataset

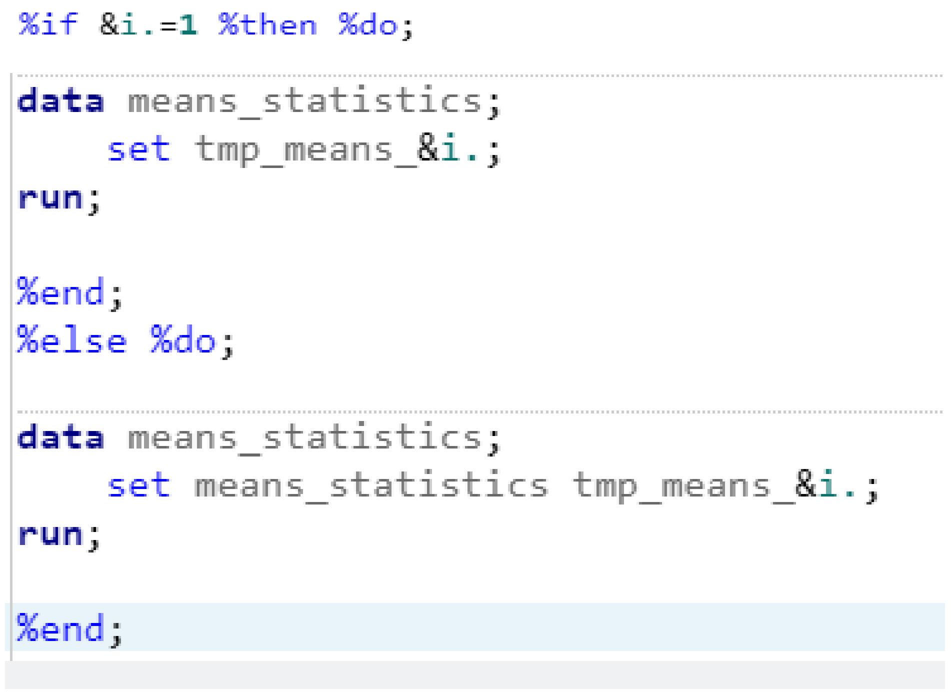
K2 Analytics
Building Skills, Building Individuals

## Write Append Logic



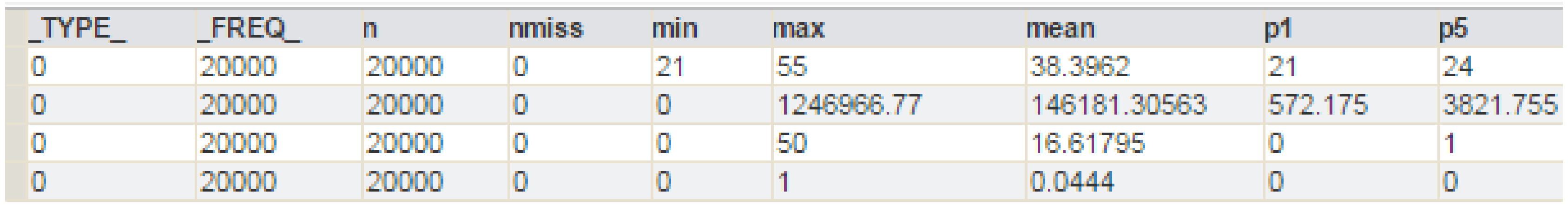
```
%do i = 1 %to &cnt_num_var.;
proc means data = &dst. noprint;
var &&col&i.;
output out = tmp_means_&i.
n=n nmiss=nmiss min=min max=max mean=mean
p1=p1 p5=p5 p10=p10 median =meadian
p75=p75 p90=p90 p95=p95 p99=p99;
run;

%end;
```

Write the APPEND
Logic here

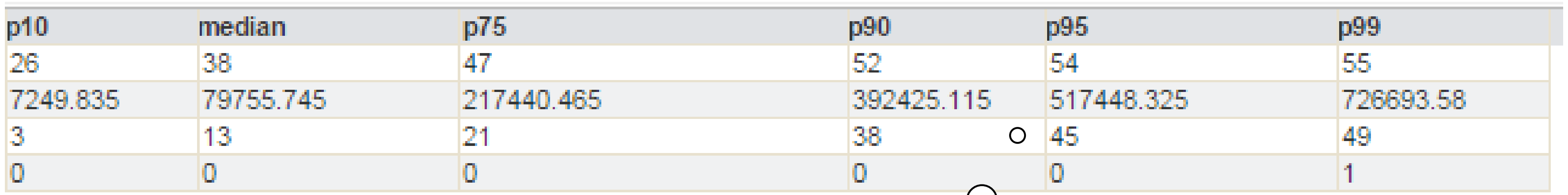


```
%if &i.=1 %then %do;
data means_statistics;
    set tmp_means_&i.;
run;

%end;
%else %do;
data means_statistics;
    set means_statistics tmp_means_&i.;
run;

%end;
```

# MEAN_STATISTICS Dataset Output



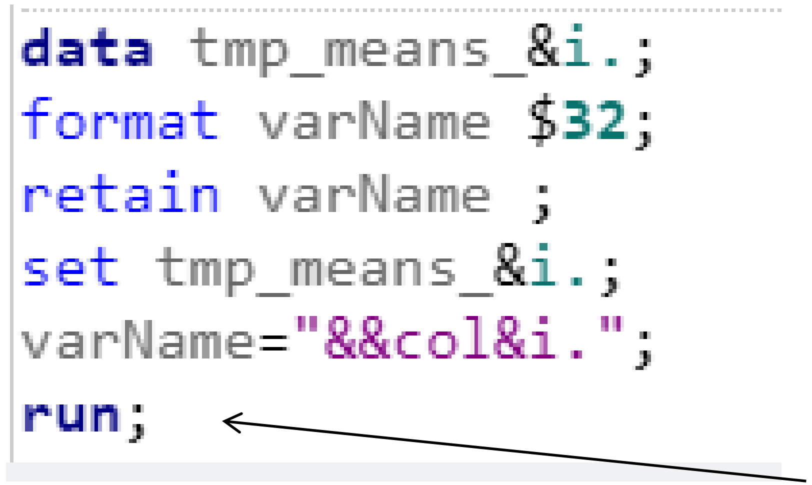
| _TYPE_ | _FREQ_ | n | nmiss | min | max | mean | p1 | p5 |
|---|---|---|---|---|---|---|---|---|
| 0 | 20000 | 20000 | 0 | 21 | 55 | 38.3962 | 21 | 24 |
| 0 | 20000 | 20000 | 0 | 0 | 1246966.77 | 146181.30563 | 572.175 | 3821.755 |
| 0 | 20000 | 20000 | 0 | 0 | 50 | 16.61795 | 0 | 1 |
| 0 | 20000 | 20000 | 0 | 0 | 1 | 0.0444 | 0 | 0 |



| p10 | median | p75 | p90 | p95 | p99 |
|---|---|---|---|---|---|
| 26 | 38 | 47 | 52 | 54 | 55 |
| 7249.835 | 79755.745 | 217440.465 | 392425.115 | 517448.325 | 726693.58 |
| 3 | 13 | 21 | 38 | 45 | 49 |
| 0 | 0 | 0 | 0 | 0 | 1 |

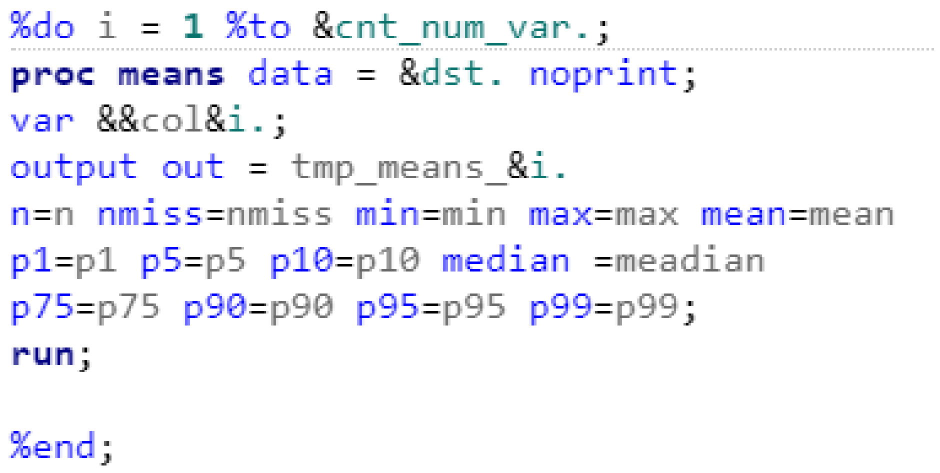
All the TMP_MEANS datasets are appended…. But I cannot figure out which row is for which field

# Adding Variable Name to Means Output



```
data tmp_means_&i.;
format varName $32;
retain varName ;
set tmp_means_&i.;
varName="&&col&i.";
run;
```

Write the above code before the **%do %end** block of code



```
%do i = 1 %to &cnt_num_var.;
proc means data = &dst. noprint;
var &&col&i.;
output out = tmp_means_&i.
n=n nmiss=nmiss min=min max=max mean=mean
p1=p1 p5=p5 p10=p10 median =meadian
p75=p75 p90=p90 p95=p95 p99=p99;
run;

%end;
```

## Exercise Time

WRITE THE EDA LOGIC FOR

DATE & CHARACTER

VARIABLES

K2 Analytics
Building Skills, Building Individuals

## Thank you

Name: Rajesh Jakhotia
Email : ar.jakhotia@k2analytics.co.in
Mobile: 89396 94874

Earning is in Learning
- Rajesh Jakhotia