

Paper BI03-2016
SAS AUTOMATION & SAS CODE IMPROVEMENT (MAKING CODES DYNAMIC)
Arjun K Shrestha, Centene Corporation

ABSTRACT

Process automation using dynamic SAS code will save time and money, giving your team more time to innovate rather than doing manual work. SAS automation is achieved by having a good architectural design, database design, robust IT infrastructure, and fluid work environment with some easy considerations and out of the box thinking, it is possible to build more dynamic SAS codes.

INTRODUCTION

This paper focuses on how architectural design, database design, IT infrastructure, fluid work environment, and dynamic SAS code can give rise to process automation. These five components play a crucial role in creating a product that is solid and innovative. Reducing code-dependency and database-dependency will save maintenance cost, time, and resources, allowing the team to focus on other products. Good database design will improve performance, for example table driven SAS code would be dynamic (see code example 1 and 3). Adding additional data in the table should not affect the code and vice versa.

In addition, this paper will focus on importance of a relationship between IT and SAS Analysts. Supporting and understanding the benefit of the SAS language by IT is crucial since in most cases, most of the components are IT related. Also, it is important to communicate and synchronize your ideas with other internal stakeholders such as the team and department; without approval from your team and department sometimes it is impossible to achieve what you want to achieve.

OVERVIEW OF THE SYSTEM COMPONENT/PLAYERS

The following graph shows different components that are important for successful process/system automation. Each of the components is discussed further.

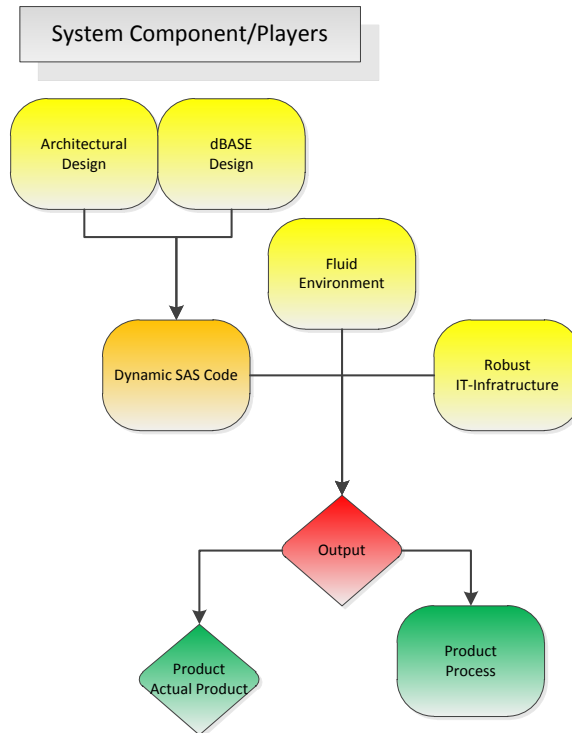


FIGURE 1

ARCHITECTURAL DESIGN

What is a good architectural design? In my opinion, good architectural design would have two main components - development design and documentation.


Good architectural design reduces code and database dependency. Making code and data less dependent will reduce maintenance cost (time and money) in the long run. That is, if we update code, there should not be a need for database change. Similarly, if we add certain data in a sourcing table that the SAS code uses, there should be no need or very little need for SAS code update. Build a table driven code structure so that you can control the output without changing the code. Refer to the mapping files below for a table driven SAS code output (See also code Example 1).

MAPPING FILE 1

order	visit_num	visit	active_ind
9	1	Nepal	y
10	2	Kansas	y
11	3	St. Louis	y
4	4	Troy	y
5	5	Collinsville	y
6	6	Edwardsville	y
7	7	New_York	y
8	8	DC	y
1	9	Nashville	y
2	10	Austin	y
3	1000	x_country	y

OUTPUT 1

FINAL2 -



The screenshot shows a SAS interface with a menu bar (Filter and Sort, Query Builder, Data, Describe, Graph, Analyze, Export, Send To) and a data table. The table has columns for 'key', 'value', and several location names: Nashville, Austin, x_country, Troy, Collinsville, Edwardsville, New_York, DC, Nepal, Kansas, and St. Louis. The 'key' column has a value of 1, and the 'value' column has a value of 'demo'. The location names are aligned with their respective values in the 'value' column.

key	value	Nashville	Austin	x_country	Troy	Collinsville	Edwardsville	New_York	DC	Nepal	Kansas	St. Louis
1	demo											

MAPPING FILE 2

order	visit_num	visit	active_ind
1	1	Nepal	y
2	2	Kansas	y
3	3	St. Louis	y
4	4	Troy	y
5	5	Collinsville	y
6	6	Edwardsville	y
7	7	New_York	y
8	8	DC	y
9	9	Nashville	y
10	10	Austin	y
11	1000	x_country	y

OUTPUT 2

key	value	Nepal	Kansas	St_Louis	Troy	Collineville	Edwardsville	New_York	DC	Nashville	Austin	x_country
1	demo	100	200	100	10	50	11	500	600	400	700	900

DATABASE DESIGN

It is important that the database design needs to align with SAS coding standards. For example, variable length, variable type, and database naming convention are some common considerations to take into account as you are designing the database.

```
TERADATA_21: Prepared: on connection 1
USING ("FACT_DIM_CK_FACT_DIM_CK_CK_FAC" INTEGER,"ACTIVE_IND" VARCHAR (1))INSERT INTO
test_table."variable_len_test_arjun" ("FACT_DIM_CK_FACT_DIM_CK_CK_FAC","ACTIVE_IND") VALUES
(:"FACT_DIM_CK_FACT_DIM_CK_CK_FAC";:"ACTIVE_IND")
```

```
TERADATA: trrlbk: ROLLBACK WORK
ERROR: Teradata insert: Column/Parameter
'test_table.variable_len_test_arjun.FACT_DIM_CK_FACT_DIM_CK_CK_FAC' does
not exist. SQL statement was: USING ("FACT_DIM_CK_FACT_DIM_CK_CK_FAC" INTEGER,"ACTIVE_IND"
VARCHAR (1))INSERT INTO
test_table."variable_len_test_arjun" ("FACT_DIM_CK_FACT_DIM_CK_CK_FAC","ACTIVE_IND") VALUES
(:"FACT_DIM_CK_FACT_DIM_CK_CK_FAC";:"ACTIVE_IND"). Insert substitution values are not shown.
ERROR: ROLLBACK issued due to errors for data set CONN.variable_len_test_arjun.DATA. Note:
FACT_DIM_CK_FACT_DIM_CK_CK_FAC is actually 30 char long
```

Error 2 generated while creating table with variable name longer than 32 character:

```
23 data test;
24 input t12345678912345678901234567890abcd v $1.;
ERROR: The variable named t12345678912345678901234567890abcd contains more than 32 characters.
25 cards;
```

Variable length - SAS does not support variable name longer than 32 characters. You will get the following error if more than 32 characters: Variable type - Another example is variable type such as *Bigint*, which is not supported by SAS. You need to cast the *Bigint* to integer before importing to SAS. Also, if the database has a *Bigint*-type column, you cannot directly insert integer value from SAS. Instead, the user needs to create and load data into a temporary table in the Teradata that has an integer column. Then copy data from temporary table to original table.

Database Naming Convention – It is important to keep a consistent name for the database. I have come across database naming convention where database name would change with each update. This will cause issues if you want to use encrypted password. Depending upon how often the database gets updated, the DSN database connection object needs to be configured as well.

```
LIBNAME SQL1 ODBC DSN=DSN_NAME USER=USER_NAME PW=' ENCRYPTED_PASSWORD' STRINGDATES=NO SCHEMA=DBO;
```

It is possible to connect dynamically to different database residing within a server using *init-string*; however, it does not support encrypted password.

```
%LET INIT_STRING = "PROVIDER=SQLOLEDB.1; User ID=user_name; password=password; PERSIST SECURITY INFO=TRUE;
INITIAL CATALOG = database_name ; DATA SOURCE=data_source_name";
LIBNAME SQLP OLEDB INIT_STRING=&INIT_STRING. SCHEMA=DBO;
```

While designing database tables, keep in mind how it may affect the SAS code. For example, table-driven SAS code would be more dynamic and scalable as opposed to a hard-coded SAS code. You will also need to keep in mind how you are going to save the error code. Sometimes you might need to do edit check and keep pass/fail records of each step of code execution. For example if it is ETL process, I prefer keeping before insert, after insert, and duplicate count, if any.

ROBUST IT-INFRASTRUCTURE

A good IT infrastructure is important for SAS support. Depending on the protocol of the company and department, IT may have a little or a bigger role to play.

For example, you may need IT approval to turn on shell command. It is important to have shell command capability where SAS needs to move files around. Ability to automatically send e-mail after the completion of SAS job to respective stakeholders is a nice feature to have. However, one needs IT approval to use e-mail feature in SAS. Finally, while scheduling of SAS jobs, you need IT support.

So it is important that IT work for SAS support. However, the structure may vary based on the organization goals. The IT support is either built around SAS, or SAS code needs to accommodate IT requirements, or both IT and SAS requirement are compromised for common goal.

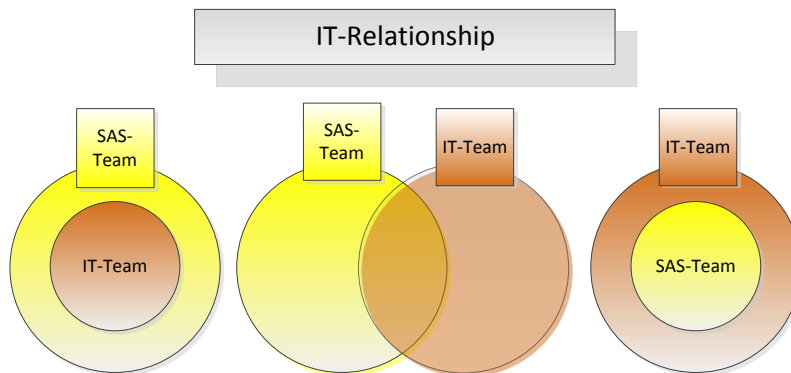


FIGURE 2

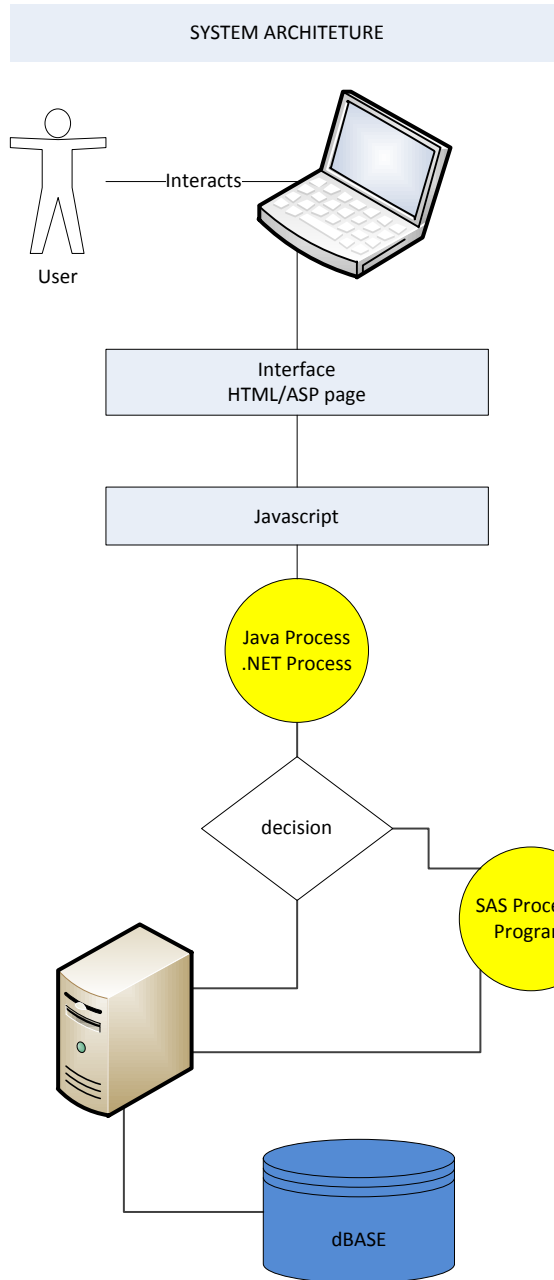
FLUID ENVIRONMENT

If you have a system where moving from Development to Test and Test to Production is fluid, then the product development and bug fixing will be very smooth, robust, and dynamic. I have seen structure where individuals are empowered and trained to use all three environments. There is hierarchical structure, where certain environment is run by specific stakeholders at a specific time. Both have pros and cons, however the former is much better than the later.

The culture to accept change is also equally important. For example, stakeholders get apprehensive to use a program, where they have little knowledge. It does not hurt to give it a try. Of course, you have to keep in mind the existing resources that are in your exposure. Your ideas need to follow your company policies and procedure.

A SIMPLE SYSTEM ARCHITECTURE

The figure below represents system architecture: a user select certain job to complete; the JavaScript captures the necessary parameter and passes them to Java or .NET process; depending on the task, Java or .NET process then calls SAS process and updates database.



EXAMPLE 1

Dynamic SAS Code

```

DATA MAIN;
  INPUT KEY VALUE $;
  CARDS;
  1 DEMO
  ;

```

RUN;

```
DATA SECONDARY (WHERE=(KEY IS NOT NULL));  
    INPUT KEY VISIT_NUM COST VISIT $20. ;  
    CARDS;  
    1 1 100 NEPAL  
    1 2 200 KANSAS  
    1 3 100 ST._LOUIS  
    1 4 10 TROY  
    1 5 50 COLLINSVILLE  
    1 6 11 EDWARDSVILLE  
    1 7 500 NEW_YORK  
    1 8 600 DC  
    1 9 400 NASHVILLE  
    1 10 700 AUSTIN  
    .. .. .  
    1 1000 900 X_COUNTRY  
    ;
```

RUN;

```
/*  
    METHOD 1  
*/
```

```
PROC SQL;  
    CREATE TABLE FINAL AS  
    SELECT A.KEY  
           , A.DEMO  
           , C1.COST AS NEPAL  
           , C2.COST AS KANSAS  
           ..  
           , C1000.COST AS X_COUNTRY  
    FROM MAIN AS A  
    LEFT JOIN SECONDARY AS C1  
        ON A.KEY = C1.KEY  
        AND VISIT_NUM=1  
    LEFT JOIN SECONDARY AS C2  
        ON A.KEY = C1.KEY  
        AND VISIT_NUM=2  
    ..  
    LEFT JOIN SECONDARY AS C1000  
        ON A.KEY = C1.KEY  
        AND VISIT_NUM=1000  
    ;  
QUIT;
```

```
/*  
    METHOD 2  
*/
```

```
PROC SQL;  
    CREATE TABLE FINAL AS  
    SELECT A.KEY  
           , A.VALUE  
           , COST.C1 AS NEPAL  
           , COST.C2 AS KANSAS  
           ..  
           , COST.C1000 AS X_COUNTRY  
    FROM MAIN AS A  
    LEFT JOIN (  
        SELECT DISTINCT KEY  
           , MAX(C1) AS C1  
           , MAX(C2) AS C2  
           ..  
           , MAX(C1000) AS C1000  
        FROM (  
            SELECT KEY  
           , CASE WHEN VISIT_NUM=1 THEN COST END AS C1  
           , CASE WHEN VISIT_NUM=2 THEN COST END AS C2  
           ..  
        )  
    )  
    ;
```

```

, CASE WHEN VISIT_NUM=1000 THEN COST END AS C1000
FROM SECONDARY
)AS INNER_A
) AS COST
ON A.KEY=COST.KEY
;
QUIT;

/*****
METHOD 3
*****/

PROC TRANSPOSE DATA=SECONDARY (WHERE=(VISIT_NUM IS NOT NULL)) OUT=SECONDARY_OUT (DROP=_NAME_)
;
*TRANSPOSE ON;
BY KEY;
*TRANSPOSE VARIABLE;
ID VISIT;
*TRANSPOSE VALUE;
VAR COST;

RUN;

PROC SQL;
CREATE TABLE FINAL AS
SELECT MAIN.KEY
, MAIN.VALUE
, SECONDARY.*
FROM MAIN AS MAIN
LEFT JOIN SECONDARY_OUT AS SECONDARY
ON MAIN.KEY =SECONDARY.KEY
;

QUIT;

/*****
METHOD 3
WHAT DO YOU DYNAMICALLY CHANGE THE COLUMN ORDER?
HOW DO YOU DYNAMICALLY CHANGE THE NAME OF THE FIELD?
ONE SOLUTION: CREATE MAP FILE OR TABLE
*****/

PROC IMPORT DATAFILE="\\MAP_FILE_LOCATION\MAP.XLSX"
DBMS=XLSX
OUT=MAP (WHERE=(UPCASE (ACTIVE_IND) = 'Y'));

RUN;

%LET DREF = %SYSFUNC (OPEN (MAP, IS));
%LET TOTAL_ROWS =%SYSFUNC (ATTRN (&DREF, NLOBS));
%LET DCLOSE = %SYSFUNC (CLOSE (&DREF.));

%PUT DREF=&DREF.;
%PUT TOTAL_ROWS=&TOTAL_ROWS.;
%PUT DCLOSE=&DCLOSE.;

PROC SQL NOPRINT;
SELECT ORDER
, VISIT_NUM
, COMPRESS (VISIT, '-&*')
INTO :ORDER1-:ORDER&TOTAL_ROWS.
, :VISIT_NUM1-:VISIT_NUM&TOTAL_ROWS.
, :VISIT1-:VISIT&TOTAL_ROWS.

FROM MAP
ORDER BY ORDER
;

QUIT;

%PUT ORDER = &ORDER1.;
%PUT VISIT_NUM = &VISIT_NUM1.;
%PUT VISIT = &VISIT1.;

```

```

OPTION MPRINT MLOGIC;
%MACRO DYNAMIC_TRANSPOSE(OUT_DATASET=);
  PROC SQL;
    CREATE TABLE &OUT_DATASET. AS
    SELECT A.KEY
           , A.VALUE
           %DO K=1 %TO &TOTAL_ROWS.;
           , COST.C&K. AS &&VISIT&K..
           %END;

    FROM MAIN AS A
    LEFT JOIN (
      SELECT DISTINCT KEY
             %DO J=1 %TO &TOTAL_ROWS.;
             , MAX(C&J.) AS C&J.
             %END;

      FROM (
        SELECT KEY
             %DO I = 1 %TO &TOTAL_ROWS.;
             , CASE WHEN VISIT_NUM=&&VISIT_NUM&I.. THEN COST END AS C&I.
             %END;

        FROM SECONDARY
        )AS INNER_A
      ) AS COST
    ON A.KEY=COST.KEY
  ;
QUIT;

%MEND DYNAMIC_TRANSPOSE;
%DYNAMIC_TRANSPOSE(OUT_DATASET=FINAL2);

```

MAPPING FILE 1

order	visit_num	visit	active_ind
9	1	Nepal	y
10	2	Kansas	y
11	3	St_Louis	y
4	4	Troy	y
5	5	Collinsville	y
6	6	Edwardsville	y
7	7	New_York	y
8	8	DC	y
1	9	Nashville	y
2	10	Austin	y
3	1000	x_country	y

OUTPUT 1

FINAL2 -

Filter and Sort Query Builder | Data Describe Graph Analyze | Export Send To

key	value	Nashville	Austin	x_country	Troy	Collinsville	Edwardsville	New_York	DC	Nepal	Kansas	St_Louis
1	demo	400	700	900	10	50	11	500	600	100	200	100

MAPPING FILE 2

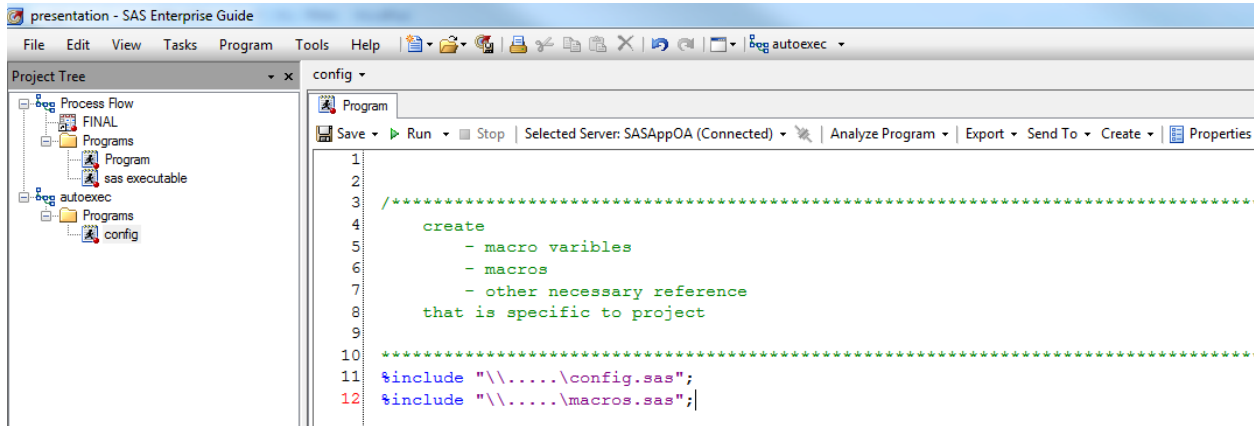
order	visit_num	visit	active_ind
1	1	Nepal	y
2	2	Kansas	y
3	3	St._Louis	y
4	4	Troy	y
5	5	Collinsville	y
6	6	Edwardsville	y
7	7	New_York	y
8	8	DC	y
9	9	Nashville	y
10	10	Austin	y
11	1000	x_country	y

OUTPUT 2

key	value	Nepal	Kansas	St_Louis	Troy	Collinsville	Edwardsville	New_York	DC	Nashville	Austin	x_country
1	demo	100	200	100	10	50	11	500	600	400	700	900

EXAMPLE 2

SAS EG autoexec node containing configuration SAS code:



Create Executable batch file using VBScript. Update the project location and save file as .VBS

```

OPTION EXPLICIT
DIM APP
CALL DOWORK
'SHUT DOWN THE APP
IF NOT (APP IS NOTHING) THEN
  APP.QUIT
  SET APP = NOTHING
END IF

SUB DOWORK()

  ' START UP ENTERPRISE GUIDE USING THE PROJECT NAME
  DIM PRJNAME
  DIM PRJOBJECT
  PRJNAME = "PROJECT_LOCATION\TEST_PROJECT.EGP"
  SET APP = CREATEOBJECT("SASEGOBJECTMODEL.APPLICATION.6.1")
  
```

```

IF CHECKERROR("CREATEOBJECT","CREATED OBJECT SUCCESSFULLY") = TRUE THEN
  EXIT SUB
END IF

' OPEN THE PROJECT
SET PRJOBject = APP.OPEN(PRJNAME,"")

IF CHECKERROR("APP.OPEN","EG APPLICATION OPENED SUCCESSFULLY" ) = TRUE THEN
  EXIT SUB
END IF

' RUN THE PROJECT
PRJOBject.RUN

IF CHECKERROR("PROJECT.RUN","PROJECT RAN SUCCESSFULLY") = TRUE THEN
  EXIT SUB
END IF

' SAVE THE NEW PROJECT
PRJOBject.SAVE

IF CHECKERROR("PROJECT.SAVE","PROJECT SAVED SUCCESSFULLY") = TRUE THEN
  EXIT SUB
END IF

PRJOBject.CLOSE
IF CHECKERROR("PROJECT.CLOSE","PROJECT CLOSED SUCCESSFULLY") = TRUE THEN
  EXIT SUB
END IF

      MSGBOX("EXECUTED SUCCESSFULLY")
END SUB

```

```

FUNCTION CHECKERROR(FNNAME,NUM)
  CHECKERROR = FALSE
  DIM STRMSG
  DIM ERRNUM
  IF ERR.NUMBER <> 0 THEN
    STRMSG = NUM & "ERROR #" & HEX(ERR.NUMBER) & VBCRLF & "IN FUNCTION " & _
      FNNAME & VBCRLF & ERR.DESCRPTION
    MSGBOX STRMSG
    CHECKERROR = TRUE
  ELSE
    'MSGBOX(NUM)
  END IF
END FUNCTION

```

EXAMPLE 3

```

/*****
  GET ALL THE VARIABLES FROM SOURCE DATA
  KEEP ONLY REQUESTED VARIABLES, FLAG IF NOT FOUND IN SOURCE TABLE
  *****/
*%LET ID=2;
DATA SOURCE_DATA;
  INFILE CARDS DLM=' ';
  INPUT ID  NAME $  A_NUM B_NUM C_NUM D_NUM E_NUM F_NUM G_NUM H_NUM I_NUM J_NUM K_NUM
         A_DEN B_DEN C_DEN D_DEN E_DEN F_DEN G_DEN H_DEN I_DEN J_DEN
K_DEN  ;
  CARDS;
  1,ROBERT,1,3,4,5,2,3,4,7,1,2,7,1,2,3,5,6,7,8,7,5,9,0
  2,HARRY,1,2,3,5,6,7,8,7,5,9,0,1,3,4,5,2,3,4,7,1,2,7
  ;
RUN;

/*****
  A=SCORE0
  *****/

```

```

B=SCORE1
C=SCORE2
D=SCORE3
E=SCORE4
F=SCORE5
G=SCORE6
H=SCORE7
I=SCORE8
J=SCORE9
K=SCORE10
*****/
DATA _MAP;
  INFILE CARDS DLM=' ';
  INPUT ID ORDER INPUT_VAR $ OUTPUT_VAR $;
  CARDS;
  1,1,A,SCORE0
  1,2,B,SCORE1
  1,3,C,SCORE2
  1,4,I,SCORE8
  2,1,A,SCORE0
  2,2,G,SCORE6
  2,3,B,SCORE1
  2,4,K,SCORE10
  2,5,L,SCORE11

RUN;

%MACRO RUN_ID(ID=);
  DATA _VARNAMES (KEEP=VNAME );
    LENGTH TYPE $4.;
    SET SOURCE_DATA (OBS=1) ;

    ARRAY N{*} NUMERIC ;
    DO I = 1 TO DIM(N) ;
      VNAME = VNAME(N{I}) ;
      TYPE='NUM';
    OUTPUT ;
    END ;

    ARRAY C{*} CHARACTER_ ;
    DO I = 1 TO DIM(C) ;
      VNAME = VNAME(C{I}) ;
      TYPE='CHAR';
    OUTPUT ;
    END ;
  RUN ;

  PROC SQL;
    CREATE TABLE _FINAL_VARNAMES AS
    SELECT DISTINCT *
    FROM
    (
      SELECT TRANWRD(VNAME, ' NUM', '') AS VAR_FINAL
      FROM _VARNAMES (WHERE= (INDEX (VNAME, '_NUM') >0))
      UNION
      SELECT TRANWRD(VNAME, ' DEN', '') AS VAR_FINAL
      FROM _VARNAMES (WHERE= (INDEX (VNAME, '_DEN') >0))
    )
    ;

  QUIT;

  PROC SQL NOPRINT;
    SELECT STRIP(VAR_FINAL) INTO :S_VAR_ALL SEPARATED BY ' '
    FROM _FINAL_VARNAMES
    ;
  QUIT;
%LET S_VAR_ALL=&S_VAR_ALL.;
%PUT S_VAR_ALL=&S_VAR_ALL.;

```

```

/*****
    GET THE REQUESTED VARIABLES FROM MAPPING
*****/
PROC SQL;
    CREATE TABLE _HVAR AS
    SELECT INPUT_VAR, OUTPUT_VAR
    FROM _MAP
    WHERE ID=&ID.
    ORDER BY ORDER
    ;

QUIT;
%LET IVARCOUNT=&SQLOBS.;
%PUT IVARCOUNT=&IVARCOUNT;

PROC SQL;
    SELECT INPUT_VAR
           , OUTPUT_VAR
    INTO :INPUTVAR1-:INPUTVAR&IVARCOUNT.
           , :OUTPUTVAR1-:OUTPUTVAR&IVARCOUNT.

    FROM _HVAR
    ;

QUIT;

PROC SQL NOPRINT;
    SELECT ''' || STRIP(INPUT_VAR) || ''' INTO :INPUT_VAR_ALL SEPARATED BY ','
    FROM _HVAR
    ;

QUIT;

%LET PVARCOUNT=&SQLOBS.;
%PUT PVARCOUNT=&PVARCOUNT;
%PUT INPUT_VAR_ALL=&INPUT_VAR_ALL;

/*****
    CREATE EXCLUDING MACRO VARIABLE
*****/
PROC SQL NOPRINT;
    SELECT STRIP(VAR_FINAL) || '_DEN'
           , STRIP(VAR_FINAL) || '_NUM'
    INTO :EXCLUDE_VAR_ALL_D SEPARATED BY ' '
           , :EXCLUDE_VAR_ALL_N SEPARATED BY ' '

    FROM _FINAL_VARNAMES
    WHERE STRIP(VAR_FINAL) NOT IN (&INPUT_VAR_ALL.)
    ;

QUIT;

%LET EVARCOUNT=&SQLOBS.;
%PUT EVARCOUNT=&EVARCOUNT;
%PUT EXCLUDE_VAR_ALL_D=&EXCLUDE_VAR_ALL_D;
%PUT EXCLUDE_VAR_ALL_N=&EXCLUDE_VAR_ALL_N;

/*****
    CREATE DATASET THAT CONTAINS REQUESTED VARIABLES. PUT -1 VALUE WHEN THE REQUESTED
    VARIABLE IS NOT FOUND
*****/
%MACRO LOOP;
    PROC SQL;
    CREATE TABLE DEMO_&ID. AS
    SELECT ID,
    NAME,
    %DO I=1 %TO &IVARCOUNT.;
        %IF %INDEX(&S_VAR_ALL.,&&INPUTVAR&I.) > 0 %THEN %DO;

            SUM(&&INPUTVAR&I._NUM) /
            SUM(&&INPUTVAR&I._DEN) AS &&OUTPUTVAR&I._RATE,
        %END;
        %ELSE %DO;
            -1 AS &&OUTPUTVAR&I._RATE,
        %END;
    %END;

```

```

%END;

%DO I=1 %TO &IVARCOUNT.;
  %IF &I NE &IVARCOUNT. %THEN %DO;
    %IF %INDEX(&S_VAR_ALL.,&&INPUTVAR&I.) > 0 %THEN %DO;
      SUM(&&INPUTVAR&I._DEN) AS &&OUTPUTVAR&I._DEN,
      SUM(&&INPUTVAR&I._NUM) AS &&OUTPUTVAR&I._NUM,

    %END;
    %ELSE %DO;
      -1 AS &&OUTPUTVAR&I._DEN,
      -1 AS &&OUTPUTVAR&I._NUM,

    %END;
  %END;
%ELSE %DO;
  %IF %INDEX(&S_VAR_ALL.,&&INPUTVAR&I.) > 0 %THEN %DO;
    SUM(&&INPUTVAR&I._DEN) AS &&OUTPUTVAR&I._DEN,
    SUM(&&INPUTVAR&I._NUM) AS &&OUTPUTVAR&I._NUM

  %END;
  %ELSE %DO;
    -1 AS &&OUTPUTVAR&I._DEN,
    -1 AS &&OUTPUTVAR&I._NUM

  %END;
%END;

%END;
FROM SOURCE_DATA (DROP=&EXCLUDE_VAR_ALL_D. &EXCLUDE_VAR_ALL_N.)
WHERE ID=&ID.
;
QUIT;

%MEND;
%LOOP;
%MEND RUN_ID;
%RUN_ID (ID=1);
%RUN_ID (ID=2);

```

CONCLUSION

SAS automation is achieved through good architectural design, database design, IT-Infrastructure, and most importantly the team, department, and company culture. One business process is linked to another, often overlapping. You need to foresee the need and scope of the project. Making SAS codes as dynamic and independent as possible, helps to produce an awesome product.

REFERENCES

<http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#a001328775.htm>

<https://support.sas.com/resources/papers/EffectivelyMovingSASDataintoTeradata.pdf>

<http://support.sas.com/kb/14/700.html>

<https://communities.sas.com/t5/SAS-Enterprise-Guide/Automation-in-EG-using-VBScript/td-p/189324>

<http://support.sas.com/documentation/cdl/en/lrcon/68089/HTML/default/viewer.htm#p0ji1unv6thm0dn1gp4t01a1u0g6.htm>

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Arjun K Shrestha
Centene Corporation
618-789-4833
shresthaarjun@yahoo.com