PharmaSUG 2017 - Paper QT05

My bag of SAS® lifehacks

Dmytro Hasan, Experis Clinical, Kharkiv, Ukraine

ABSTRACT

Have you ever come across anything special in another person's code that made your brain freeze? If your answer is yes, then I hope this article will be of interest to you. Sometimes it can be very difficult to grab the message the programmer wanted to convey in their code lines. In this article, I would like to share my remarks about the issues that I find the most interesting. I have caught sight of some of them in other people's SAS programs or read up in different books and used them in my programs. We will consider a missing RUN statement at the end of the data step and calling procedure(s) without specifying the dataset, which decreases the length of the code. Some features of PROC SQL will be analyzed which may seem non-trivial for the programmers who are less experienced with this procedure. We will go back to the beginning and try to find out what COMMENT does and in what ways it can be helpful. Also I would like to draw the reader's attention to two functions, namely ifn() and input() and how they handle missing values, how byte() allows to make the output more refined. Moreover, I would like to outline the utility of -rcl options of put(). I will demonstrate how to make a musical notification for data issues, compilation errors and the end of compiling with sound().

INTRODUCTION

First of all, not to worry: this article is not going to be a trivial list of tips for some obscure data leaving reader with a feeling of confusion as to where they can be implemented. All the lifehacks will be related to working with data and creating tables comprising an integral part of Clinical SAS. We will use The Vaccine Adverse Event Reporting System (VAERS), which is a free access database, created by the Food and Drug Administration (FDA) and Centers for Disease Control and Prevention (CDC). You can download the datasets for the analysis in .csv format using this link: https://vaers.hhs.gov/data/index. For more details as to how the database is organized and of the variables in the datasets please see (VAERS data use guide).

We are going to look at the data which were reported from 2014.01.01 till 2017.01.13. Moreover, we will examine the list of vaccines included in Immunization Schedules for Infants and Children (for more details see Recommended Immunizations for Children (Birth through 6 years)). A macro program from Appendix is used to access the database and create the SAS datasets for our analysis. So, we suggest analyzing what lifehacks can be used to make working with this data a little easier.

LIFEHACKS

WHEN IFN() FUNCTION IS NOT THE SAME AS IF STATEMENT.

Have you ever seen this NOTE in the log (see Display 1)?

```
NOTE: Missing values were generated as a result of performing an operation on missing values.
      Each place is given by: (Number of times) at (Line):(Column).
      113007 at 1764:55
NOTE: There were 123626 observations read from the data set WORK.DATA2014_2017.
NOTE: The data set WORK.DATA2014_2017_AGE has 3072 observations and 20 variables.
NOTE: DATA statement used (Total process time):
      real time           0.22 seconds
      cpu time            0.12 seconds
```

Display 1. Missing values note

It appears when we make arithmetical operations on missing values. Consider an example when IFN() function generates this NOTE:

```
data example;
   set data2014_2017;
```

```
    CALCULATED_AGE= IFN(not missing(CAGE_YR), CAGE_YR+CAGE_MO, CAGE_MO);
run;
```

We use IFN() function to evaluate the expression and return a value depending if missing(CAGE_YR) is true or false. After checking the log, we will find out that the following line:

```
CALCULATED_AGE= IFN(not missing(CAGE_YR), CAGE_YR+CAGE_MO, CAGE_MO);
```

generates the NOTE. This happens because all return values for IFN() function are always evaluated by SAS, so CAGE_YR+CAGE_MO is evaluated even if CAGE_YR is missing.

Usually this “issue” is fixed using IF statement. If we replace the function call with IF clause with the same condition, we will not get this “issue”. It is really a great way to fix it.

Nevertheless, what if we do not want to change anything and keep using IFN()? That is where this lifehack may help you. Use SUM() function when you do arithmetical operations on values. Please note that if all the values in SUM() function are missing, you will still get the same NOTE. In this case, if it makes sense for your data, add 0 to make the result not missing. In our case, in Appendix, we wrote as follows:

```
CALCULATED_AGE= IFN(not missing(CAGE_YR), SUM(CAGE_YR,CAGE_MO,0), CAGE_MO);
```

As a result, CALCULATED_AGE will be 0 even if CAGE_YR and CAGE_MO are both missing, and this line will newer throw “Missing values” note.

CONVERTING WITH INPUT().

If we want to convert a character variable into the numeric one, INPUT() function is usually used with a specified informat. However, if we don’t specify all the values of our variable in the format for some reason, we will get this NOTE (Display 2):

```
NOTE: Invalid argument to function INPUT at line 1029 column 17.
```

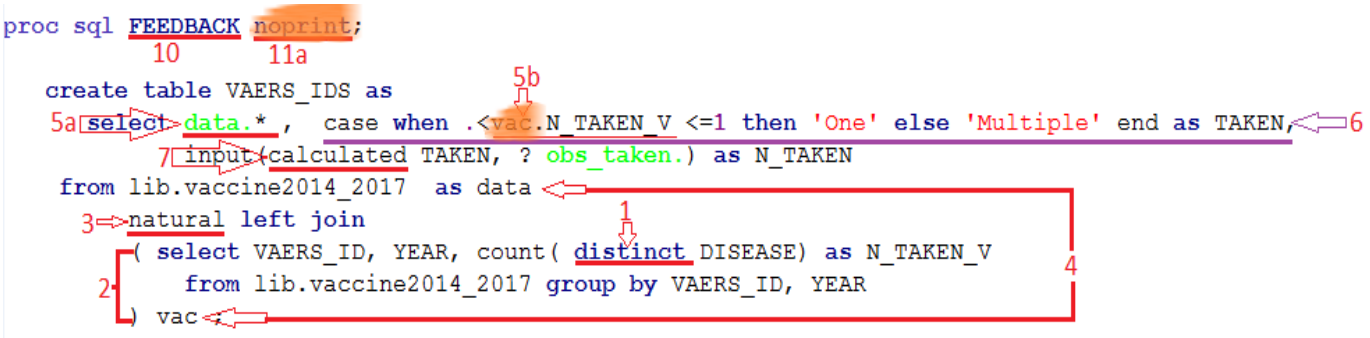
Display 2. Invalid values note

One of the ways to deal with it is to ask SAS to convert all the values not specified in the informat into missing ones. This is done by putting the optional question mark (?) or double question mark (??) before the informat name in INPUT() function (you can read more on the difference between them in SAS INPUT function). Please note that in PROC SQL input() function does not work with double question mark (??).

WHO IS AFRAID OF SQL?

Before we start talking about SQL procedure, note that all the datasets used in the example below were created by a macro from Appendix.

SQL is probably one of the most powerful and challenging procedures in SAS. We can use it in many different ways: to create datasets, variables and macro variables, calculate the number of records, join the datasets (akin to merging in SAS), etc. It can be really helpful, and sometimes one SQL query can fit the logics of many procedures and data steps and greatly simplify things. Take a look at the example code in Display 3:



```
    create table SE_TAB as
     select YEAR, EMERGENT , DISEASE,
            count(distinct case when TAKEN = 'One'  then  VAERS_ID else . end)
                 as ONE label = "ONE ",
            count(distinct case when TAKEN = 'Multiple' then  VAERS_ID else . end)
                 as MUL label = "MULTIPLE"
     from VAERS_IDS
     group by 1,2 ,3
     order by 1,2;

quit;

proc sql noprint;
   select distinct DISEASE into : All_VV separated by '" "' from SE_TAB where ONE>= 5 and MUL>= 5;
quit;
```


Display 3. SQL code

May the gurus of SQL forgive us, but here we are not going to call the variables columns and the observations rows to simplify things and draw an analogy with a DATA step.

1. DISTINCT keyword eliminates duplicate records. In our example it is specified in count() function which calculates the number of non-missing observations. As a result, we get the number of non-missing unique vaccinations by variables VAERS_ID and YEAR.
2. We can wrap the select statement in brackets — it's called "In-Line View". We can think of the result as a dataset.
3. Where are the by variables in the join statement? In this case we used a "natural" join. It merges two datasets by variables which are present in both of them, so we don't need to specify them. However, you must make sure that both datasets have only those variables to merge by in common.
4. Dataset aliases can be specified after the dataset name with or without using 'as'. Usually they are used to separate the same variables from different datasets.
5. a) The STAR of SQL. The alias "data" tells us that we will take variables from the corresponding dataset. The asterisk means that all the variables will be included in the final result.

   b) In this case it is not necessary to use alias before the variable because it exists in only one dataset.
6. The example of how conditions can be used in PROC SQL. We specify cases when a condition is true, and it returns a value. You may specify as many conditions as you want. In the end we write the name of the variable which contains the result.
7. We need to put "calculated" before we can use new variables defined in the same select clause, otherwise we will get an error.
8. The numbers after "group by" and "order by" clauses are the index numbers of the variables used in the select clause.
9. We can count the number of unique records which satisfy some conditions. For example, the variable ONE contains the number of unique VAE where a patient had only one vaccination.
10. FEEDBACK option helps us to understand what is hidden from the user. It shows which variables are selected by the asterisk and the natural join in the SAS log.
11. a) When we create the dataset with PROC SQL, it is not necessary to use noprint option because the result won't be printed anyway.

    b) In this case we need to suppress the report with noprint.

12. As we said at the beginning of this chapter, SQL can be useful for creating macro variables. Here we create macro variable All_VV representing a list of all the unique vaccinations separated by " " (double quote space double quote) administered at least 5 times with variable EMERGEN = "Y" and at least 5 times with "N".

Those were some very brief explanations of non-trivial things which may confuse some programmers. Please use (SAS SQL Procedure User's Guide) to find more details about proc SQL.

LIFEHACK 1.

Look at how we create SE_TAB dataset:

```
proc sql;
   create table SE_TAB as
     select YEAR, EMERGENT, DISEASE,
     count(distinct case when TAKEN = 'One' then VAERS_ID else . end)
     as ONE label = "ONE ",
     count(distinct case when TAKEN = 'Multiple' then VAERS_ID else . end)
     as MUL label = "MULTIPLE"
    from VAERS_IDS
    group by 1, 2, 3
    order by 1, 2;
quit;
```

What lifehack did we use? Here we calculate a number of unique vaccinations records by YEAR and EMERGENT flag and divide the result into two variables, which tell us if multiple vaccines or only one were taken. Without PROC SQL it would take us multiple calls of various procedures and data steps to derive these variables. It is a great example of how AE tables can be done very quickly.

LIFEHACK 2.

Macro variable All_VV contains a list of vaccines separated by " "(double quote space double quote). It is an unusual separator which is not used very often, is it? But it will help us easily select only the vaccines in &All_VV. Let us see how it works:

```
data MEET_VAC;
   set VAERS_IDS (where = (DISEASE in ("&All_VV")));
run;
```

We put &All_VV in double quotes and use it in IN() function. This lifehack allows you to keep the relevant records without typing them.

BACK TO THE FUTURE.

Just like Marty, we are going back in time to the first versions of SAS. We would like to share some things that are often forgotten but still have viability. Hopefully, you will discover something new for yourself.

COMMENT.

While writing a program, it is a good practice to make comments for other programmers or for yourself. Most of us use *message; or /*message*/. But there is another way to do it, coming from the past, and it still works on new versions of SAS. We are talking about COMMENT statement with semicolon at the end of the message. Let us see how it works in Display 4:

```
***** This is a comment *******;

Comment  - This is a comment;
```

Display 4. Comment

WHERE DID WE LOSE DATA= AND RUN?

Sometimes you may bump into a procedure step without data option specified. You may be wondering which dataset is used there? It is the dataset from the last DATA step. Let us have a look at the example:

```
data SAMPLE;
   set MEET_VAC;
COMMENT - Delete duplicated vaccines for one VAE;
proc sort dupout = DUP_VAC nodupkey;
   by DISEASE VAERS_ID;
proc freq noprint;
   by DISEASE;
   table N_TAKEN*EMERGENT_N /chisq relrisk;
   output out = TESTS chisq RELRISK;
run;
```

We are creating a dataset and calling procedures without data option. In this case, SAMPLE dataset is used in PROC SORT and FREQ. As you may notice, run statement is not used either because it is optional. In this case, proc statements also behave as run ones. In the first versions of SAS programs you also did not have to specify data option and run statement before calling a procedure. For more details on when and why statements and other features in SAS were created see (SAS Communications).

Note that not writing run statement and data option every time may help you avoid writing a few lines of code. They are nonetheless useful to understand the logics of the program, so do not overdo it!

OTHER LIFEHACKS

SOUND CHECK.

We are going to have a look at SOUND() function and where it can be used. You can find more details about this function in (SAS CALL SOUND function).

When program execution takes a lot of time, some people want to use this time doing something else. You could set up a sound notification which tells you that the program has finished running. It can be just a beep, or something more exquisite. If you would like to hear your favorite song, it is also possible! Only your imagination is the limit. If you are interested, the following link may be of use to you: http://www.lexjansen.com/nesug/nesug13/33_Final_Paper.pdf.

This is a SAS macro which can be run at the end of the program as a signal that it finished the execution.

```
COMMENT: Artist - Deep Purple, track - 'Smoke on the Water';
%macro SmokeOnTheWater;
   data _null_;
      do i = 1 to 2;
         CALL SOUND(311.127,300); CALL SOUND(349.228,280);
         CALL SOUND(391.995 ,400); CALL SOUND(311.127,280);
         CALL SOUND(349.228,200); CALL SOUND(415.305 ,150);
         CALL SOUND(391.995 ,500); CALL SOUND(311.127,300);
         CALL SOUND(349.228,280); CALL SOUND(391.995 ,400);
         CALL SOUND(349.228,280); CALL SOUND(311.127,850);
      end;
   run;
%mend SmokeOnTheWater;
```

It is not the only way how you can use SOUND(). For instance, in the previous chapter we created dataset DUP_VAC which has a duplicate record. It means that one vaccine was administered several times. Such cases are not considered to be normal and need to be investigated.

```
%macro check_dup_vac;
   proc sql noprint;
      select count(*) into : _data_issue from DUP_VAC;
```

```
    quit;
    %if &_data_issue %then %SmokeOnTheWater;
%mend check_dup_vac;
```

If DUP_VAC is not empty, our macro will play the song and you will know at once that you have a data issue even without looking at the log file. Also, it can be useful for determining compilation errors.

HOW BYTE() & PUT() CAN HELP TO MAKE AN OUTPUT.

We are going to report the results of PROC FREQ which were calculated in the previous chapter. Before we start, we will need to modify the TESTS dataset and create variables for the output.

```
data chisq_odds;
   set TESTS (keep = DISEASE _PCHI_ P_PCHI _CRAMV_ _RROR_ L_RROR U_RROR);
   length col1- col5 $100;
   col1 = strip(DISEASE); col2 = put(round(_PCHI_,.01), 8.2 -c);
   col3 = put(round(_CRAMV_,.01), 8.2 -c);
   col4 = put( _RROR_ , ODDSR8.3 -r)||' ('||put( L_RROR , ODDSR8.3 -
c)||','||put( U_RROR , ODDSR8.3 -c)||')' ;
   col5 = put(round(P_PCHI,.01), PVALUE6.4 -l);
   label col1 = "Vaccines" col2 = 'Chi-Square' col3 = "Cramer's
V%sysfunc(byte(178))" col4 = "Odds Ratio%sysfunc(byte(179)) ( 95% CI )"
col5 = "p-value%sysfunc(byte(185)) ";
run;
title1 "Association between emergent VAERS and number of taken
vaccinations.";
title2 "Populations infants in age 12-23 month.";
footnote1 "%sysfunc(byte(185))Corresponding p-value for Chi-Square
statistic.";
footnote2 "%sysfunc(byte(178))the strenght measure of the assosiations that
the Chi-Square test detected.";
footnote3 "%sysfunc(byte(179))the odds of emergent vaccination when it was
received multiple vaccines to one vaccine.";
proc print data = chisq_odds  L;
   var col1-col5;
run;
```

By using BYTE() function we are creating references specifying more detailed information for the column in the footnote. Usually the numbers in square brackets are used to make references but in this way they look more refined. Have a look at the Display 5:

Association between emergent VAERS and number of taken vaccinations.
Populations infants in age 12-23 month.

Obs	Vaccines	Chi-Square	Cramer's V²	Odds Ratio³ (95% CI)	p-value¹
1	Chickenpox	61.70	0.24	4.260 (2.905 , 6.248)	<.0001
2	Chickenpox,Measles,Mumps,Rubella	0.88	0.06	1.324 (0.735 , 2.385)	0.3500
3	Diphtheria,Haemophilus b,Pertussis,Polio,Tetanus	0.44	0.09	1.530 (0.434 , 5.401)	0.5100
4	Diphtheria,Hepatitis B,Pertussis,Polio,Tetanus	2.43	0.16	2.799 (0.737 , 10.626)	0.1200
5	Diphtheria,Pertussis,Polio,Tetanus	12.39	0.26	4.370 (1.835 , 10.407)	<.0001
6	Diphtheria,Pertussis,Tetanus	10.94	0.12	1.909 (1.296 , 2.812)	<.0001
7	Haemophilus b	57.97	0.27	13.428 (5.793 , 31.124)	<.0001
8	Hepatitis A	6.45	0.09	1.693 (1.124 , 2.550)	0.0100
9	Hepatitis B	4.44	0.24	4.833 (1.001 , 23.344)	0.0400
10	Influenza	22.61	0.21	2.871 (1.840 , 4.482)	<.0001
11	Measles,Mumps,Rubella	15.53	0.12	1.873 (1.367 , 2.567)	<.0001
12	Pneumococcal	11.41	0.12	2.511 (1.449 , 4.350)	<.0001

¹Corresponding p-value for Chi-Square statistic.
²the strenght measure of the assosiations that the Chi-Square test detected.
³the odds of emergent vaccination when it was received multiple vaccines to one vaccine.

Display 5. Results of PROC FREQ

Note that the numbers for 95% confidence intervals are centered. The -c option is used in PUT() function which aligns the values in the center. For more details about PUT() functions and -rcl options please see

(SAS PUT function). Let's see how this column will change depending on specified –r (Display 6) or –l (Display 7) options in PUT() function.

Odds Ratio³ (95% CI)
4.260 (2.905, 6.248)
1.324 (0.735, 2.385)
1.530 (0.434, 5.401)
2.799 (0.737, 10.626)
4.370 (1.835, 10.407)
1.909 (1.296, 2.812)
13.428 (5.793, 31.124)
1.693 (1.124, 2.550)
4.833 (1.001, 23.344)
2.871 (1.840, 4.482)
1.873 (1.367, 2.567)
2.511 (1.449, 4.350)

Display 6. -r option.

Odds Ratio³ (95% CI)
4.260 (2.905 ,6.248)
1.324 (0.735 ,2.385)
1.530 (0.434 ,5.401)
2.799 (0.737 ,10.626)
4.370 (1.835 ,10.407)
1.909 (1.296 ,2.812)
13.428 (5.793 ,31.124)
1.693 (1.124 ,2.550)
4.833 (1.001 ,23.344)
2.871 (1.840 ,4.482)
1.873 (1.367 ,2.567)
2.511 (1.449 ,4.350)

Display 7. -l option.

CONCLUSION

Every SAS programmer may have their own bag of lifehacks, and they are unique for everyone. This article is not a full list of the author's tips and tricks but features only the most interesting ones. We hope it will inspire more programmers to share their lifehacks with other people and find little things that make their code and work better and easier

REFERENCES

Vaccine Adverse Event Reporting System Data, URL: https://vaers.hhs.gov/data/index

Centers for Disease Control and Prevention/Food and Drug Administration Vaccine Adverse Event Reporting System (VAERS), 2016. *VAERS DATA USE GUIDE*. URL: https://vaers.hhs.gov/data/ReadMeDecember2016.pdf

U.S. Department of Health and Human Services Centers for Disease Control and Prevention, American Academy of Family Physicians, American Academy of Pediatrics, 2016. *Recommended Immunizations for Children (Birth through 6 years).* URL: https://www.cdc.gov/vaccines/parents/downloads/parent-ver-sch-0-6yrs.pdf

SAS support INPUT function, URL: https://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000180357.htm

SAS SQL Procedure User's Guide, URL: https://support.sas.com/documentation/cdl/en/sqlproc/63043/HTML/default/viewer.htm#titlepage.htm

SAS Communications, URL: http://www.codecraftersinc.com/pdf/SAS_Communications.pdf

SAS support CALL SOUND function, URL: https://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#win-callrout-sound.htm

Richard Dirmyer, *I'm Going to Pop Some Tags: Using SAS® CALL SOUND Routines*. URL: http://www.lexjansen.com/nesug/nesug13/33_Final_Paper.pdf

SAS support PUT function, URL: https://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000199354.htm

CONTACT INFORMATION

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

Name: Dmytro Hasan
Enterprise: Experis Clinical
E-mail: dmytro.hasan@intego-group.com, dimagmehanic@gmail.com

APPENDIX

```
%macro read_csv;
 %do i =4 %to 7;
   proc import datafile =
"&path_to_repository.\CSV_raw_data\201&i.VAERSData\201&i.VAERSDATA.csv"
             out= data201&i. dbms=csv  ;
    getnames=yes;
   run;
   data l_data201&i;
    set  data201&i %if &i = 5 %then (rename = (HOSPDAYs=HOSPDAY)) ;;
    %if &i = 5 %then HOSPDAYS = input(strip(HOSPDAY), ? 8.) ; ;
    YEAR = 201&i;
    keep VAERS_ID YEAR AGE_YRS CAGE_YR CAGE_MO DIED L_THREAT ER_VISIT
         HOSPITAL HOSPDAYS  ;
   run;
   proc import datafile =
"&path_to_repository.\CSV_raw_data\201&i.VAERSData\201&i.VAERSVAX.csv"
               out= vaccine201&i. dbms=csv  ;
    getnames=yes;
   run;
   %let Conc_Desease = %str(strip(Disease)||strip(ifc(not
missing(Disease),',','')));
   data l_vaccine201&i (keep=VAERS_ID YEAR Disease VACCINE);
    set  vaccine201&i ;
    length  VACCINE $20 Disease $200 ;
    YEAR = 201&i;
    VACCINE = strip(VAX_TYPE);
    if find(upcase(VACCINE),"MMRV") or find(upcase(VACCINE),"VARCEL") or
       find(upcase(VACCINE),"VARZOS") then
     Disease = &Conc_Desease||"Chickenpox";
    if find(upcase(VACCINE),"6VAX-F") or find(upcase(VACCINE),"DPIPV") or
       upcase(substr(VACCINE,1,2))="TD" or find(upcase(VACCINE),"DPP") or
       find(upcase(VACCINE),"DT") then
     Disease = &Conc_Desease||"Diphtheria";
    if index(upcase(VACCINE),"DTPPHI") or index(upcase(VACCINE),"HBPV") or
       index(upcase(VACCINE),"HBHEPB") or index(upcase(VACCINE),"HIB") or
       find(upcase(VACCINE),"6VAX-F") or upcase(VACCINE)="DTAPH" or
       find(upcase(VACCINE),"DTPIHI") then
     Disease = &Conc_Desease||"Haemophilus b";
    if index(upcase(VACCINE),"HEPA") or index(upcase(VACCINE),"HEPAB") then
     Disease = &Conc_Desease||"Hepatitis A";
    if upcase(VACCINE)="HEP" or find(upcase(VACCINE),"6VAX-F") or
       index(upcase(VACCINE),"HEPB") or index(upcase(VACCINE),"DTPHEP") or
     index(upcase(VACCINE),"DTAPHE") or index(upcase(VACCINE),"HEPAB") then
     Disease = &Conc_Desease||"Hepatitis B";
    if index(upcase(VACCINE),"FLU") or index(upcase(VACCINE),"H5N1") then
     Disease = &Conc_Desease||"Influenza";
    if find(upcase(VACCINE),"MEA") or find(upcase(VACCINE),"MER") or
       index(upcase(VACCINE),"MM") then
     Disease = &Conc_Desease||"Measles";
    if index(upcase(VACCINE),"MU") or index(upcase(VACCINE),"MM") then
     Disease = &Conc_Desease||"Mumps";
    if index(upcase(VACCINE),"DTAP") or find(upcase(VACCINE),"6VAX-F") or
       find(upcase(VACCINE),"DPIPV") or find(upcase(VACCINE),"DPP") or
       index(upcase(VACCINE),"DTP") or find(upcase(VACCINE),"PER") or
       find(upcase(VACCINE),"TDAP") then
```

```
     Disease = &Conc_Desease||"Pertussis";
    if find(upcase(VACCINE),"6VAX-F") or index(upcase(VACCINE),"DTAPHE") or
       find(upcase(VACCINE),"DTAPIP") or find(upcase(VACCINE),"DPP") or
       find(upcase(VACCINE),"DTAPHEPBIP") or find(upcase(VACCINE),"DTPPHI")
       or find(upcase(VACCINE),"DTPIHI") or index(upcase(VACCINE),"IPV") or
       find(upcase(VACCINE),"OPV") then
     Disease = &Conc_Desease||"Polio";
    if index(upcase(VACCINE),"PNC") or find(upcase(VACCINE),"PPV") then
     Disease = &Conc_Desease||"Pneumococcal";
    if upcase(substr(VACCINE,1,2)) = "RV" then
      Disease = &Conc_Desease||"Rotavirus";
    if upcase(substr(VACCINE,1,3))="MER" or find(upcase(VACCINE),"MMR") or
       upcase(substr(VACCINE,1,3))="MUR" or
       upcase(substr(VACCINE,1,3))="RUB" then
     Disease = &Conc_Desease||"Rubella";
    if find(upcase(VACCINE),"6VAX-F") or find(upcase(VACCINE),"DTPPHI") or
       find(upcase(VACCINE),"DTAPIP") or index(upcase(VACCINE),"DTAPHE") or
       (upcase(substr(VACCINE,1,2)) = "DT" and
        upcase(substr(VACCINE,1,4)) ne "DTOX") or
       find(upcase(VACCINE),"MNQHIB") or
       upcase(substr(VACCINE,1,2)) in( "TD",'TT') then
     Disease = &Conc_Desease||"Tetanus";
     if not missing(Disease);
  run;
%end;
data  data2014_2017;
  set l_data2014 l_data2015 l_data2016 l_data2017;
proc sort ;
  by VAERS_ID YEAR;
run;
data  vaccine2014_2017;
  set l_vaccine2014 l_vaccine2015 l_vaccine2016 l_vaccine2017;
proc sort ;
  by VAERS_ID YEAR;
run;
proc format;
  invalue yes_no
     "Y" = 1
     "N" = 2;
  invalue obs_taken
     'One'      = 2
     'Multiple' = 1;
run;
data data2014_2017_AGE;
  set data2014_2017;
  if not missing(DIED) or not missing(L_THREAT) or
     not missing(ER_VISIT) or not missing(HOSPITAL) or
     not missing (HOSPDAYS) then
   EMERGENT = "Y";
  else
   EMERGENT = "N";
  EMERGENT_N = input(EMERGENT, yes_no.);
/*Age of patient in years calculated by (vax_date - birthdate)*/
  CALCULATED_AGE=ifn(not missing(CAGE_YR),sum(CAGE_YR,CAGE_MO,0),CAGE_MO);
  if 1 < CALCULATED_AGE <= 1.75 or 1<AGE_YRS<=1.75 ;
run;
%mend read_csv;
```

```
%macro keep_VAERS;
/* We need to keep only VAERS in all datasets where VACCINE dataset
intersects with DATA dataset*/
 proc sql;
   create table keep_VAERS as
     select distinct a.VAERS_ID, a.YEAR
    from  data2014_2017_AGE natural inner join vaccine2014_2017 as a;
 quit;

 data lib.data2014_2017;
   merge data2014_2017_AGE(in = in1) keep_VAERS(in = in2);
   by VAERS_ID YEAR;
   if in1 & in2;
 run;

 data lib.vaccine2014_2017;
    merge vaccine2014_2017(in = in1) keep_VAERS(in = in2)
          lib.data2014_2017(keep = VAERS_ID  YEAR EMERGENT EMERGENT_N);
    by VAERS_ID YEAR;
    if in1 & in2;
 run;
%mend keep_VAERS;
```

If you would like to take a look or download all the source code files, please visit this link:
https://github.com/dimagmehanic/My-bag-of-SAS-lifehacks