Paper 311-2013

"How May I Help?" The SAS® Enterprise Guide® Analyze Program Feature

Ramya Purushothaman, Cognizant Technology Solutions, Teaneck, NJ

ABSRACT:

Have you ever been looking into a lengthy and complex SAS® code, may it be inherited or your own old program, and wished you could understand what is happening inside without having to go through every line? Left without any associated documentation and wondered where to start from? Felt that it would be better to have a process flow representation of what the code does, quickly? Then the Analyze Program feature that SAS Enterprise Guide offers might work for you! This paper discusses what to expect of this feature and what not to with example analyses from the Life Sciences industry.

INTRODUCTION:

SAS® Enterprise Guide® has been looked at as the Integrated Development Environment (IDE) for SAS which can be used by business users even with minimal SAS programming experience to enjoy the benefits of SAS. However there are certain less explored features that EG hosts which can save you hours of time if utilized correctly. One such feature is the Analyze program.

If your responsibilities include maintaining large legacy SAS programs, then you probably wish you had time to bring some of those programs up-to-date. Even if the program was written a long time ago, as long as it is functioning fine, usually it is considered too costly and time consuming to spend time looking into the code and making manual changes.

Analyze SAS Program feature in SAS Enterprise Guide 4.3 gives the ability to view a complex SAS program as a set of logical nodes within a flow.

A FEW WORDS ABOUT THE PROCEDURE:

In the background SAS EG uses the SAS Code Analyzer, introduced with SAS 9.2, is a useful tool that can help you maintain and modify these legacy programs. The SAS Code Analyzer, or SCAPROC procedure, is a Base SAS® procedure. It executes an existing SAS program and while the program runs, it collects and analyzes information related to the SAS steps, input and output data, and any dependencies. It records information that can be used to enhance the manageability and efficiency of the program.

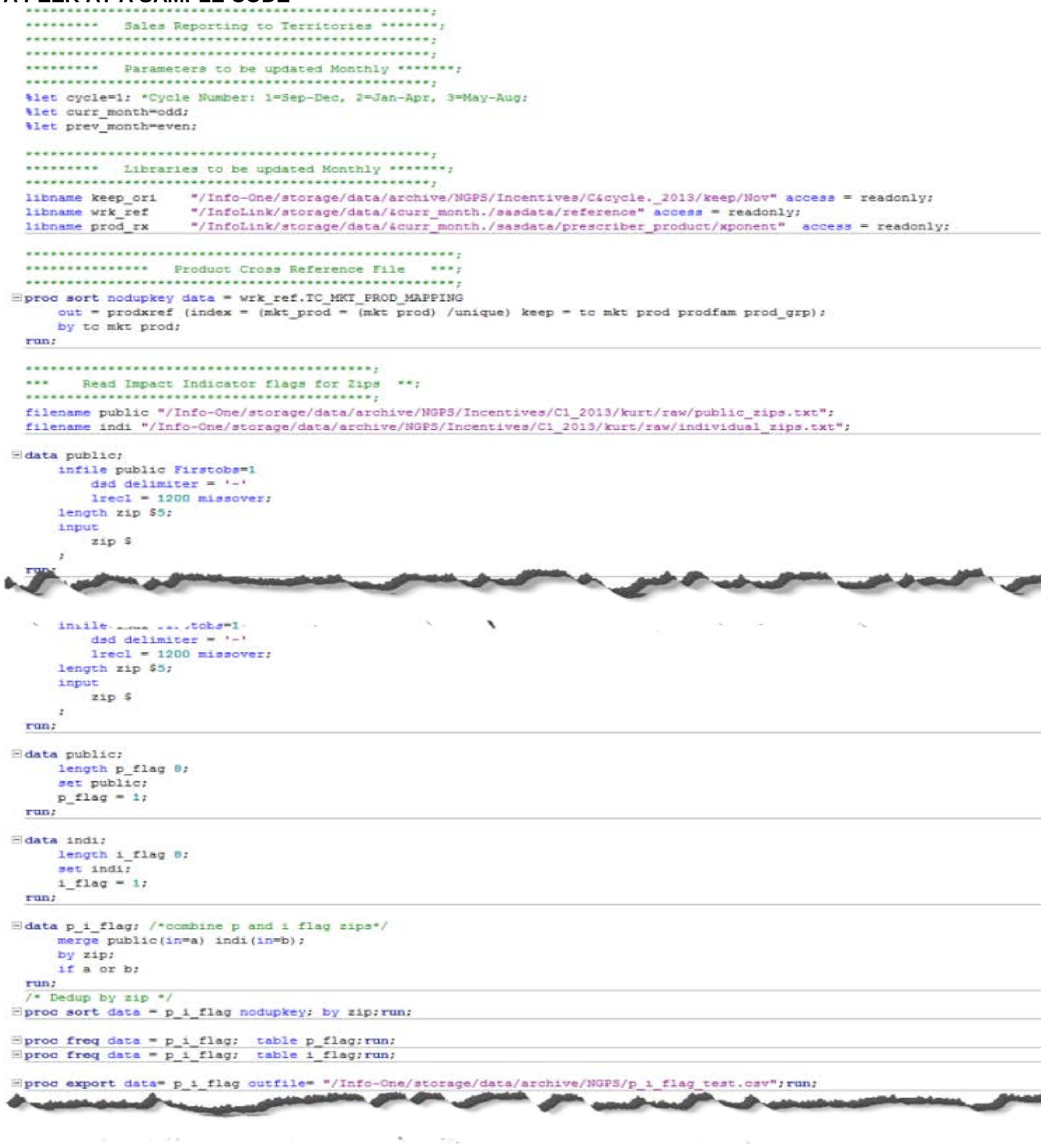
The SCAPROC procedure compiles and runs a SAS program and then reports on the details of program's effect, including which procedure and DATA steps are run and the file and data sources that are referenced and created. SAS Enterprise Guide then interprets the detailed report and creates a process flow representation of your SAS program. The result will be a new process flow with program nodes for each step, plus data nodes for each referenced data source or result. And they will be connected with arrows so that you can see how it all flows together.

When analyzing a program, things to be remembered are:

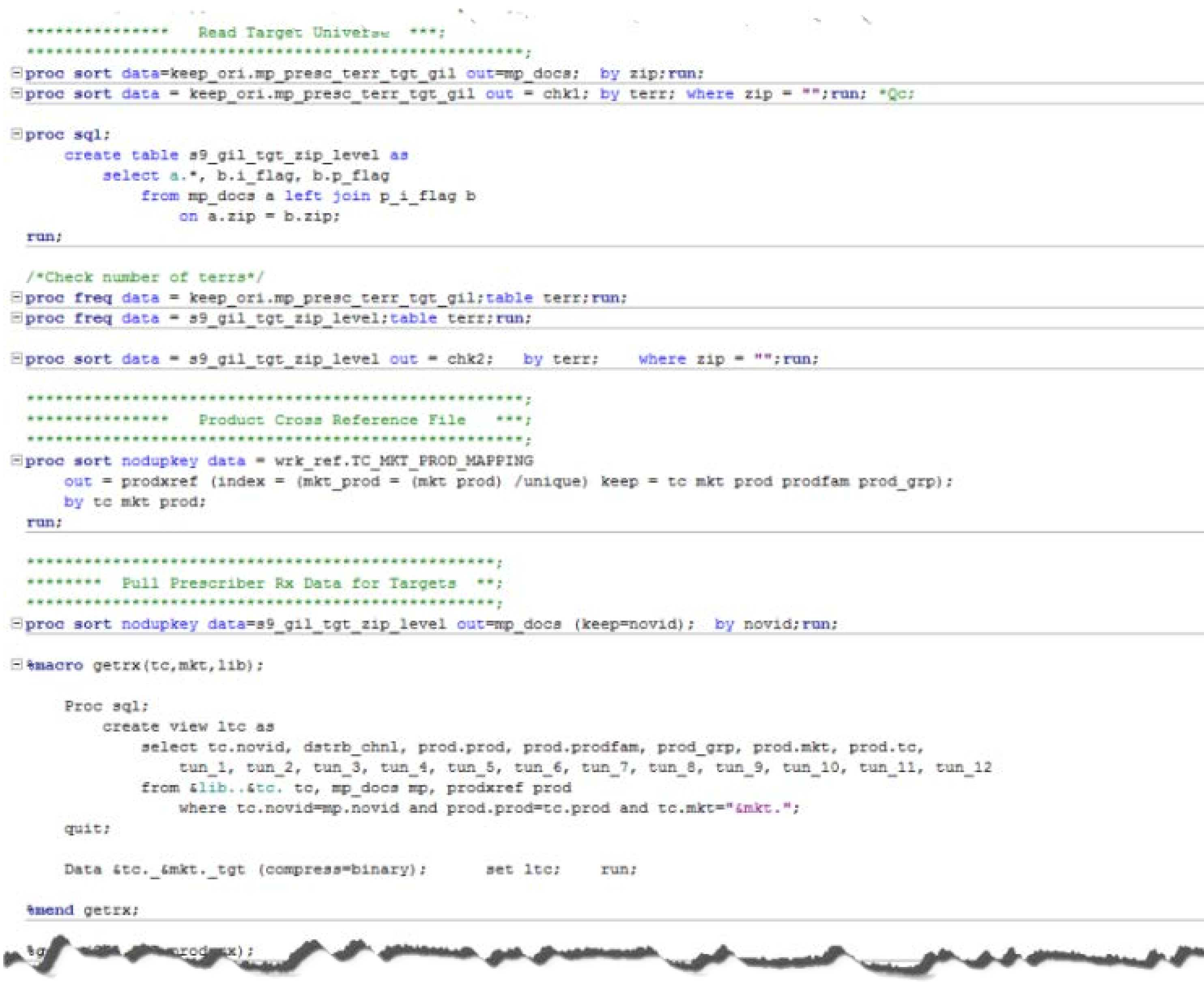
- SAS actually runs your program to analyze it, so the time it takes to complete the analysis is similar to how long it would take to run the program.
- If there are SAS errors that occur during analysis, you'll see this indicated in the analysis log. The analysis might not be completely accurate when errors are encountered.

In addition to creating a process flow from your program, SAS Enterprise Guide can help you to optimize a SAS program for use on the SAS® Grid Computing environment where instead of creating a new process flow, the result is a modified copy of the program that has special grid-computing statements inserted at key locations to tune the performance through parallel processing. As grid computing is out of scope of this paper, I leave it at that.

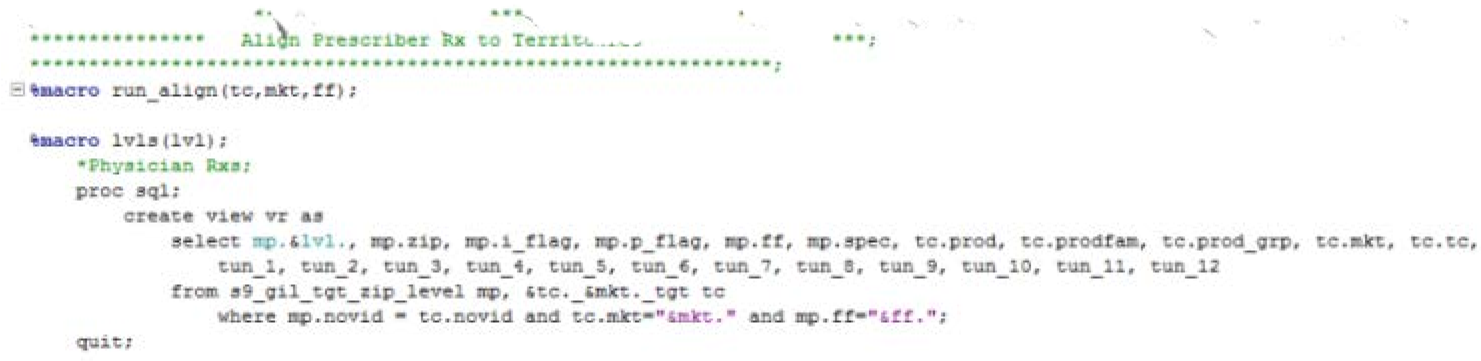
A PEEK AT A SAMPLE CODE



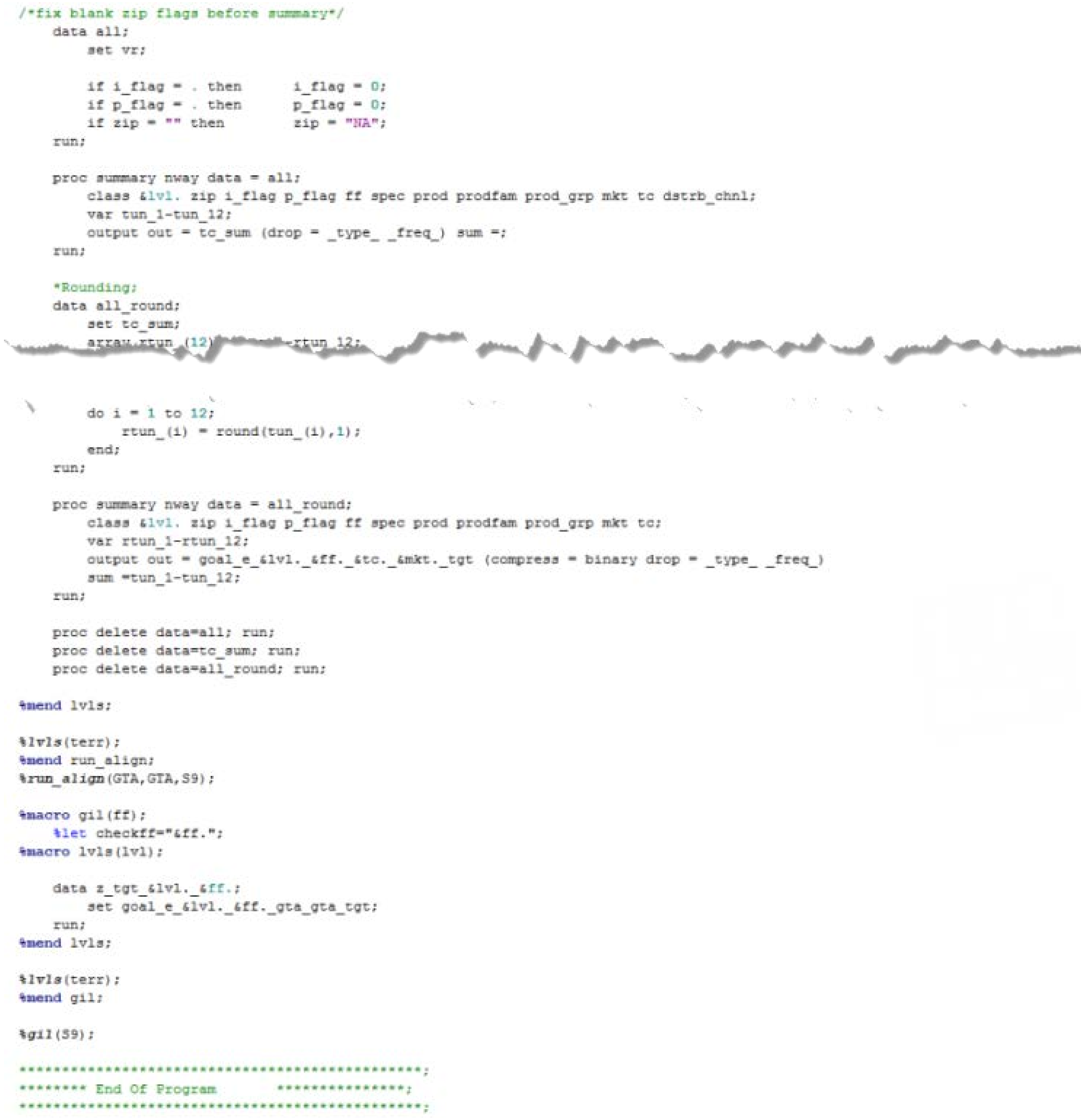
```
**************************************************;
*********   Sales Reporting to Territories *******;
**************************************************;
**************************************************;
*********   Parameters to be updated Monthly *******;
**************************************************;
%let cycle=1; *Cycle Number: 1=Sep-Dec, 2=Jan-Apr, 3=May-Aug;
%let curr_month=odd;
%let prev_month=even;

**************************************************;
*********   Libraries to be updated Monthly *******;
**************************************************;
libname keep_ori    "/Info-One/storage/data/archive/NGPS/Incentives/C&cycle._2013/keep/Nov" access = readonly;
libname wrk_ref     "/InfoLink/storage/data/&curr_month./sasdata/reference" access = readonly;
libname prod_rx     "/InfoLink/storage/data/&curr_month./sasdata/prescriber_product/xponent"  access = readonly;

*****************************************************;
***************    Product Cross Reference File    ***;
*****************************************************;
proc sort nodupkey data = wrk_ref.TC_MKT_PROD_MAPPING
    out = prodxref (index = (mkt_prod = (mkt prod) /unique) keep = tc mkt prod prodfam prod_grp);
    by tc mkt prod;
run;

*******************************************;
***    Read Impact Indicator flags for Zips  **;
*******************************************;
filename public "/Info-One/storage/data/archive/NGPS/Incentives/C1_2013/kurt/raw/public_zips.txt";
filename indi "/Info-One/storage/data/archive/NGPS/Incentives/C1_2013/kurt/raw/individual_zips.txt";

data public;
    infile public Firstobs=1
        dsd delimiter = '~'
        lrecl = 1200 missover;
    length zip $5;
    input
        zip $
    ;
run;

    in.ile [illegible] .tobs=1
        dsd delimiter = '~'
        lrecl = 1200 missover;
    length zip $5;
    input
        zip $
    ;
run;

data public;
    length p_flag 8;
    set public;
    p_flag = 1;
run;

data indi;
    length i_flag 8;
    set indi;
    i_flag = 1;
run;

data p_i_flag; /*combine p and i flag zips*/
    merge public(in=a) indi(in=b);
    by zip;
    if a or b;
run;
/* Dedup by zip */
proc sort data = p_i_flag nodupkey; by zip;run;

proc freq data = p_i_flag;  table p_flag;run;
proc freq data = p_i_flag;  table i_flag;run;

proc export data= p_i_flag outfile= "/Info-One/storage/data/archive/NGPS/p_i_flag_test.csv";run;
```



```
****************   Read Target Universe  ***;
*****************************************************;
proc sort data=keep_ori.mp_presc_terr_tgt_gil out=mp_docs;  by zip;run;
proc sort data = keep_ori.mp_presc_terr_tgt_gil out = chk1; by terr; where zip = "";run; *Qc;

proc sql;
    create table s9_gil_tgt_zip_level as
        select a.*, b.i_flag, b.p_flag
            from mp_docs a left join p_i_flag b
                on a.zip = b.zip;
run;

/*Check number of terrs*/
proc freq data = keep_ori.mp_presc_terr_tgt_gil;table terr;run;
proc freq data = s9_gil_tgt_zip_level;table terr;run;

proc sort data = s9_gil_tgt_zip_level out = chk2;   by terr;    where zip = "";run;

*****************************************************;
****************   Product Cross Reference File   ***;
*****************************************************;
proc sort nodupkey data = wrk_ref.TC_MKT_PROD_MAPPING
    out = prodxref (index = (mkt_prod = (mkt prod) /unique) keep = tc mkt prod prodfam prod_grp);
    by tc mkt prod;
run;

**************************************************;
********  Pull Prescriber Rx Data for Targets  **;
**************************************************;
proc sort nodupkey data=s9_gil_tgt_zip_level out=mp_docs (keep=novid);  by novid;run;

%macro getrx(tc,mkt,lib);

    Proc sql;
        create view ltc as
            select tc.novid, dstrb_chnl, prod.prod, prod.prodfam, prod_grp, prod.mkt, prod.tc,
                tun_1, tun_2, tun_3, tun_4, tun_5, tun_6, tun_7, tun_8, tun_9, tun_10, tun_11, tun_12
            from &lib..&tc. tc, mp_docs mp, prodxref prod
                where tc.novid=mp.novid and prod.prod=tc.prod and tc.mkt="&mkt.";
    quit;

    Data &tc._&mkt._tgt (compress=binary);      set ltc;    run;

%mend getrx;

%g[illegible]rod[illegible]x);
```



```
****************   Align Prescriber Rx to Territ[illegible]           ***;
***************************************************************;
%macro run_align(tc,mkt,ff);

%macro lvls(lvl);
    *Physician Rxs;
    proc sql;
        create view vr as
            select mp.&lvl., mp.zip, mp.i_flag, mp.p_flag, mp.ff, mp.spec, tc.prod, tc.prodfam, tc.prod_grp, tc.mkt, tc.tc,
                tun_1, tun_2, tun_3, tun_4, tun_5, tun_6, tun_7, tun_8, tun_9, tun_10, tun_11, tun_12
            from s9_gil_tgt_zip_level mp, &tc._&mkt._tgt tc
                where mp.novid = tc.novid and tc.mkt="&mkt." and mp.ff="&ff.";
    quit;
```

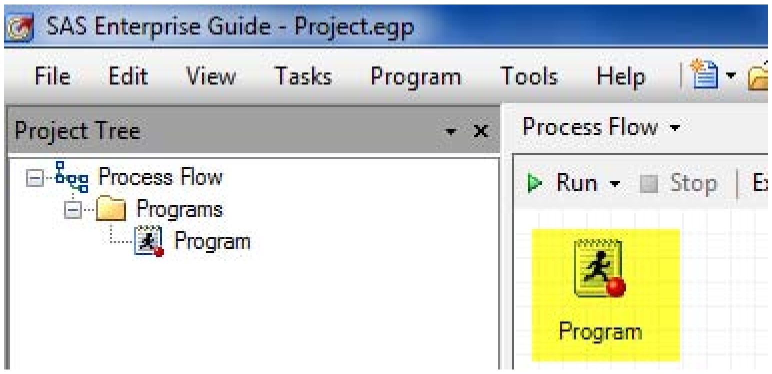


```
/*fix blank zip flags before summary*/
    data all;
        set vr;

        if i_flag = . then      i_flag = 0;
        if p_flag = . then      p_flag = 0;
        if zip = "" then        zip = "NA";
    run;

    proc summary nway data = all;
        class &lvl. zip i_flag p_flag ff spec prod prodfam prod_grp mkt tc dstrb_chnl;
        var tun_1-tun_12;
        output out = tc_sum (drop = _type_ _freq_) sum =;
    run;

    *Rounding;
    data all_round;
        set tc_sum;
        array rtun_(12) [illegible]-rtun_12;


        do i = 1 to 12;
            rtun_(i) = round(tun_(i),1);
        end;
    run;

    proc summary nway data = all_round;
        class &lvl. zip i_flag p_flag ff spec prod prodfam prod_grp mkt tc;
        var rtun_1-rtun_12;
        output out = goal_e_&lvl._&ff._&tc._&mkt._tgt (compress = binary drop = _type_ _freq_)
        sum =tun_1-tun_12;
    run;

    proc delete data=all; run;
    proc delete data=tc_sum; run;
    proc delete data=all_round; run;

%mend lvls;

%lvls(terr);
%mend run_align;
%run_align(GTA,GTA,S9);

%macro gil(ff);
    %let checkff="&ff.";
%macro lvls(lvl);

    data z_tgt_&lvl._&ff.;
        set goal_e_&lvl._&ff._gta_gta_tgt;
    run;
%mend lvls;

%lvls(terr);
%mend gil;

%gil(S9);

************************************************;
******** End Of Program       ***************;
************************************************;
```

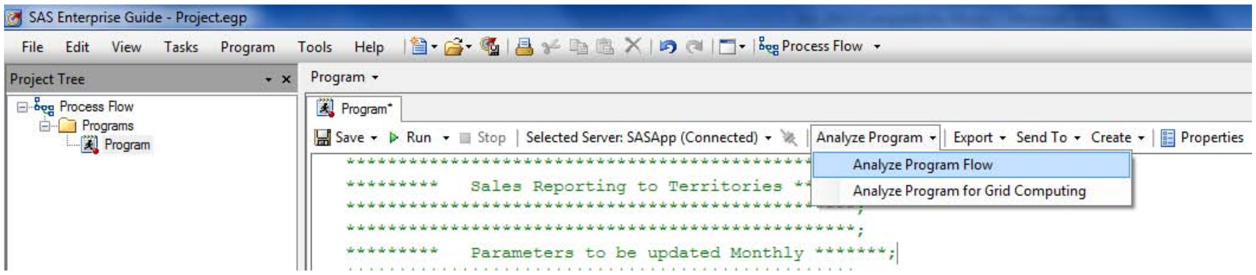
HOW ANALYZE PROGRAM CAN HELP HERE?

Let us see how to analyze the program in SAS Enterprise guide step-by-step.

Step 1: Bring the program into SAS Enterprise Guide and open it.

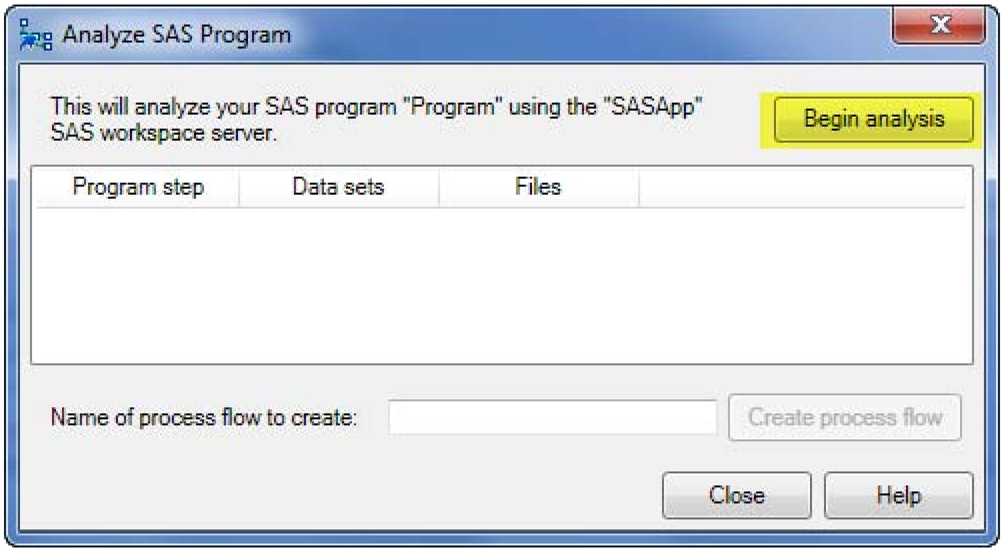


Step 2: Click Analyze Program ➔ Analyxe program Flow



```
**********************************************
*********   Sales Reporting to Territories **
**********************************************
**************************************************;
*********   Parameters to be updated Monthly *******;
```

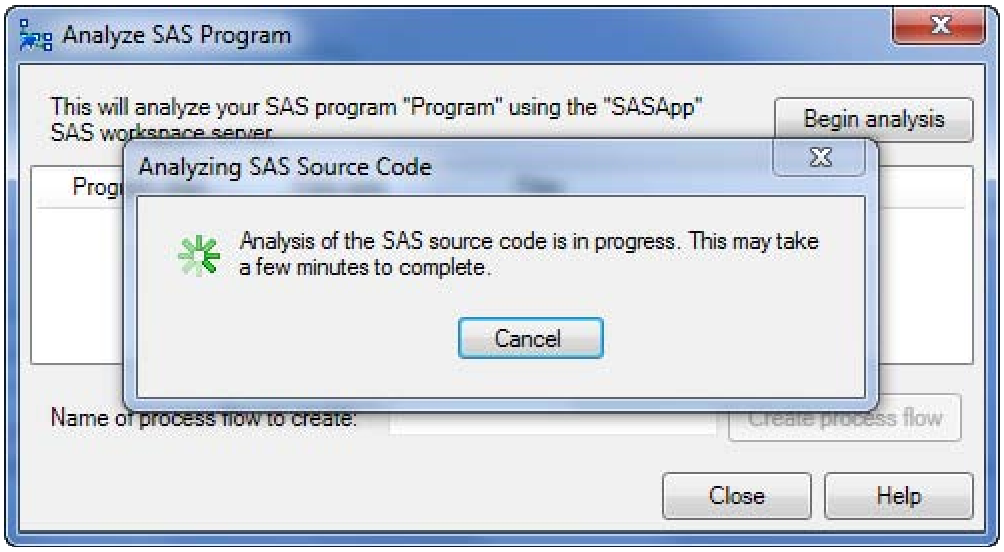
Step 3: In the pop-up window click Begin Analysis



This will analyze your SAS program "Program" using the "SASApp" SAS workspace server.

Name of process flow to create:

Step 4: SAS is performing its analysis of the code.

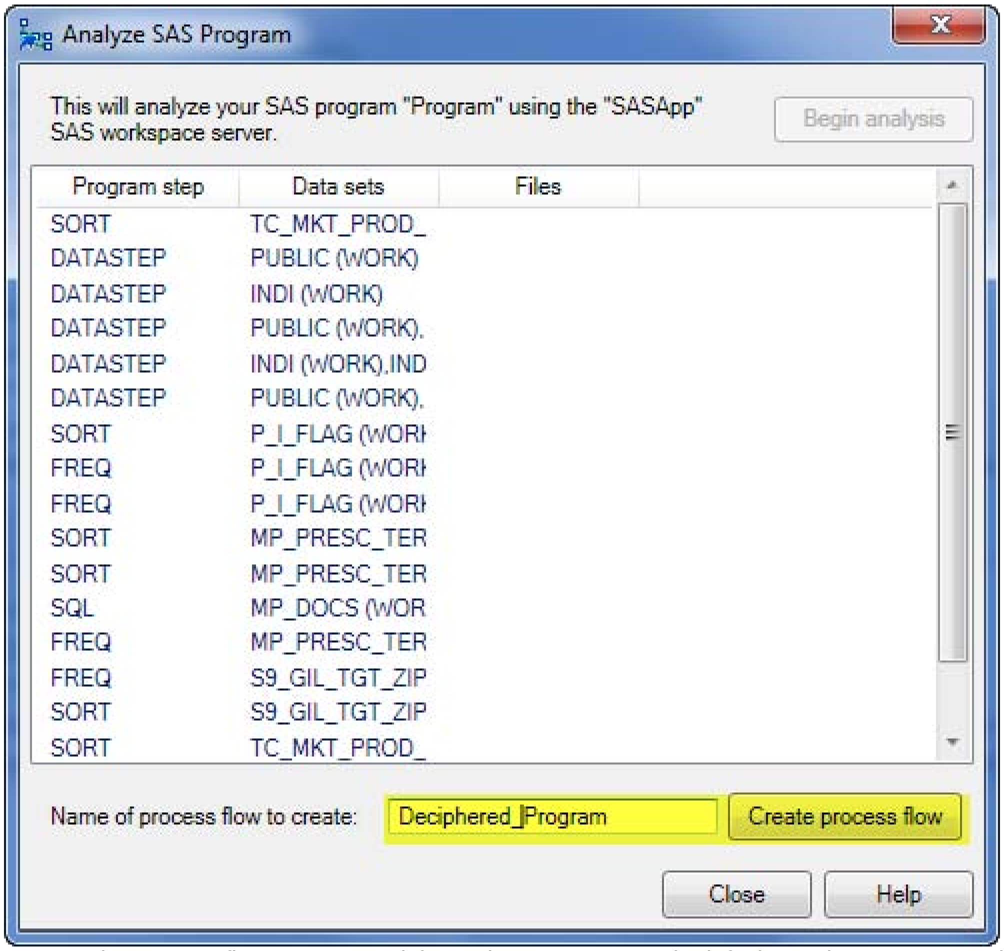


This will analyze your SAS program "Program" using the "SASApp" SAS workspace server.

Analysis of the SAS source code is in progress. This may take a few minutes to complete.

Name of process flow to create:

Step 4: Next step is to create the process flow.

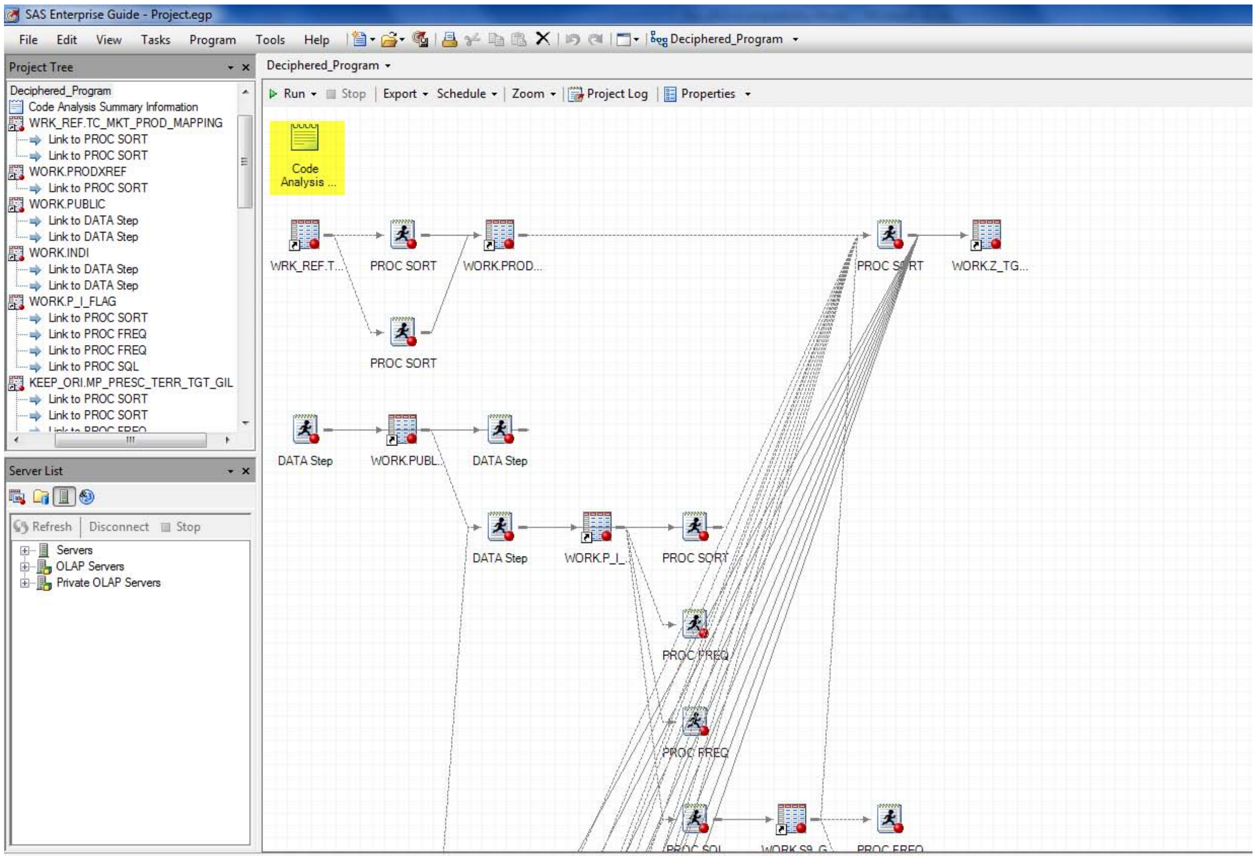


This will analyze your SAS program "Program" using the "SASApp" SAS workspace server.

Program step	Data sets	Files
SORT	TC_MKT_PROD_	
DATASTEP	PUBLIC (WORK)	
DATASTEP	INDI (WORK)	
DATASTEP	PUBLIC (WORK),	
DATASTEP	INDI (WORK),IND	
DATASTEP	PUBLIC (WORK),	
SORT	P_I_FLAG (WORI	
FREQ	P_I_FLAG (WORI	
FREQ	P_I_FLAG (WORI	
SORT	MP_PRESC_TER	
SORT	MP_PRESC_TER	
SQL	MP_DOCS (WOR	
FREQ	MP_PRESC_TER	
FREQ	S9_GIL_TGT_ZIP	
SORT	S9_GIL_TGT_ZIP	
SORT	TC_MKT_PROD_	

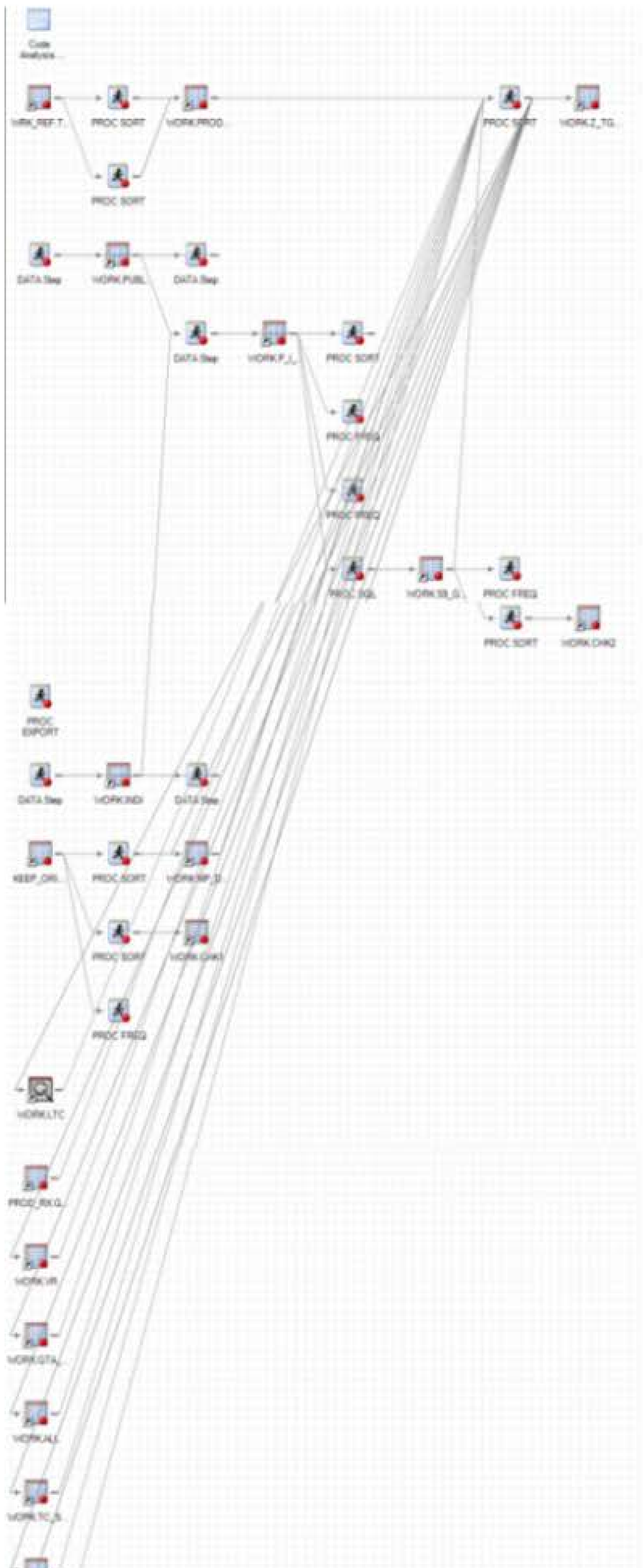
Name of process flow to create: Deciphered_Program

Step 5: The process flow pops up and the project tree pane to the left shows the steps created in the flow.



The code Analysis document seen above will have analysis of each step in your program, counts of libraries and datasets used in each step, PROC CONTENTS of the datasets used containing variable details etc.

What does the complete process flow look like?


Code Analysis ...
WRK_REF_T...
PROC SORT
WORK.PROD...
PROC SORT
WORK.Z_TG...
PROC SORT
DATA Step
WORK.PUBL
DATA Step
DATA Step
WORK.P_L...
PROC SORT
PROC FREQ
PROC FREQ
PROC SQL
WORK.SB_G...
PROC FREQ
PROC SORT
WORK.CHK2
PROC EXPORT
DATA Step
WORK.INDI
DATA Step
KEEP_ORI...
PROC SORT
WORK.RP_D...
PROC SORT
WORK.CHK1
PROC FREQ
WORK.LTC
PROD_RX.G...
WORK.VR
WORK.GTA...
WORK.ALL
WORK.TC_S...

Why is it cool?

1) End to end process flow in minutes of simple to moderately long codes
2) It is run time so missing source datasets can be identified while analyzing.
3) After Analysis if there are any other errors, this feature points them out like Syntax errors, Real- time error detection if variable name not found in table etc
4) Analyzing in GRID mode can help you in optimization

WHAT ARE NOT GOOD CANDIDATES FOR ANALYZE PROGRAM TO ACT UPON?

1) Very Huge programs that will take hours to run may take longer for the Analysis to complete and will spit a process flow that is difficult to understand.
2) Not so well-written programs and those that create a lot of intermediate datasets will have a clumsy looking Process flow
3) Like said before, the code is executed while analyzing so be cautious about overwriting someone else's file or as a best practice the libraries need to be changed to work directory.
4) Unlike when analyzing manually, program cannot be run step by step to complete the flow. It is analyzed fully.
5) PROC EXPORT step is deciphered and the output file is created as specified in the program but still EG throws an error and the flow is not completed. This seems to be a known issue in this feature in EG 4.3. Only workaround at this point is to create a separate program node just for export steps and link them to relevant datasets so they will run in sequence. (SAS support mentioned that this problem is fixed in Enterprise Guide 5.1)
6) PROC FREQ is not run automatically during the analysis but it has to be run manually to see the results window.
7) SAS MACROS are not resolved into separate constituent steps but are bundled with the previous procedure into one step in the process flow which is misleading and not complete.

CONCLUSION

The analyze program feature can help to some extent and it is for you to explore and leverage this cool feature to your benefit. Hope this paper will help you save a few hours to savor other interesting SAS magic.

ACKNOWLEDGMENTS

The author would like to thank her colleagues, customers and family for their support and encouragement. She also would like to thank SAS Global Forum and Cognizant Technology Solutions for this opportunity to present the paper.

RECOMMENDED READING

http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a003199742.htm

http://support.sas.com/resources/papers/proceedings10/313-2010.pdf

CONTACT INFORMATION

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

Ramya Purushothaman
Cognizant Technology Solutions
500 Frank W.Burr Blvd.
Teaneck, NJ 07666
Ramya.Purushothaman@Cognizant.com