Paper SM02

Automation of Processes for Programming Study Leads in SAS

Fred Ross, Parexel, Stockport, United Kingdom

ABSTRACT

During a study a programmer taking a leading role may need to communicate a complex idea, status of certain tasks, or some other information that would usually take a long time to write up manually before sharing with the team. To assist such an individual several SAS procedures and features can be utilized to automate parts or the entirety of this process. This paper will look at doing the following in SAS: automating programming lead tasks, sending team or individual emails, creating relevant study driven tables and visualizing information that will be used to express anything from concepts and ideas to timelines and important dates. In using SAS macros given in this paper it is expected that a programmer in a lead role will be able to reduce their time needed in project management which will free up time to spend elsewhere such as on programming tasks.

INTRODUCTION

The main goal of this paper and the code found within is to automate some processes and thus save time for programming study leads. It is written bearing in mind the variety of environments in which a study lead may work. It does presume common concepts are in use on a study such as programming plans and issue logs. The paper is not meant to provide an all-encompassing suite of macros to automate any study out of the box. It is instead a general selection of macros and concepts which if adapted and applied to a study can provide additional tools to a programming leads coding toolbox.

Please see appendix for complete code, the body of the paper contains mainly email and dataset examples.

SETUP FOR AUTOMATED EMAILS ON A STUDY

There are three prerequisite for the code found in this paper. These are:

1. Logging onto email account securely through SAS.
2. The TEAM dataset.
3. External files to be read into SAS.

LOGGING INTO EMAIL THROUGH SAS

It is important to note that logging into an email account securely through SAS is not within the scope of this paper. This is due to the many SAS environments that can be used, for example SAS Studio, Enterprise Guide and PC SAS; the many ways email accounts may be setup; and the different SOPs that might govern how and if a user can log in outside of the email client. The code in this paper was tested using a dummy Gmail account. Please take care when logging into an email account through SAS.

TEAM DATASET

This dataset will include all the information needed to send emails to team members. Each row will represent an individual team member and will require: a unique ID, email address and the name used to refer to each person. This could be stored as an external file such as Excel so that it can be easily updated with new team member details and have the email programs read in each time. A quicker method would be to generate and store this dataset wholly within SAS an example of this method can be seen in the Appendix.

TABLE 1 – TEAM DATASET SPECIFICATION

ID	A unique ID is relatively easy as all users will have unique login credentials so simply using usernames of team members is the obvious choice here. A nice bonus is the TEAM dataset can then be used to look up each members login for granting permissions to certain areas of a study.
EMAIL	Email addresses are the most obvious example needed to be able to send emails to the team.
NAME	The name is used to address recipients of emails. Unlike the unique ID this does not need to be unique as it will mainly be used for creating text strings for email body text. For example, “Dear Jay, Fred…”.

Note: Additional roles and subgroups can be added simply by creating new variables. In the TEAM dataset used in this paper Blinded and Role variables are used as examples for additional variables that can be included.

TABLE 2 TEAM DATASET EXAMPLE

id	name	email	blind	role
Bob1	Bob	phuse2020emailtest@gmail.com	blinded	statistician
Greg1	Greg	phuse2020emailtest@gmail.com	unblinded	data_manager
Brian1	Brian	phuse2020emailtest@gmail.com	unblinded	programmer
Greg2	Gregory	phuse2020emailtest@gmail.com	blinded	data_manager

Note: For the purposes of demonstration the email addresses are the same for each team member, though the code has been tested with unique email addresses.

READING IN EXTERNAL FILES

To make use of the information found within various study documents said files need to bread into SAS. The files made use of in the code are the programming plan and issue log. Once these are read in, authors, QCers and task assignees can be cross referenced using the TEAM dataset to generate an emails and their recipients.

One important post processing step after reading in a study file will be to align the Unique ID found within the TEAM dataset. For example, my unique ID in the TEAM dataset might be FRED1 but in the Issue Log I am referred to by my initials FR. Simply a single data step can be used to generate a Unique ID column that matches TEAM dataset.

TABLE 3 – PROGRAMMING PLAN IN SAS

dataset_name	*prog_name*	*first_line_id*	*qc_id*	*first_line_ran*	*qc_ran*
DM	sdtm_dm_prog.sas	Bob1	Greg1	2020-04-25	2020-04-26
ADSL	adsl_prog.sas	Bob1	Greg1	2020-04-25	2020-04-26
ADAE	adae_prog.sas	Brian1	Greg1	2020-04-25	2020-04-26

BUILDING BLOCK & SEND_EMAIL_TEMPLATE MACROS

The macros found in this paper will make use of several building block macros where possible as is best practice when creating a macro and especially when creating a suite of macros. (Appendix B)

%RECIPIENT MACRO

The RECIPIENT macro will generate a list of email addresses which will be put into the macro variable EMAIL_TO for sending the email with EMAIL. In addition, it will create a text string list of all recipients into NAME_TO which can then be used in the text of the email.

%EMAIL MACRO

Simply at the end of program EMAIL will use the list of email addresses in EMAIL_TO generated in RECIPIENT to send the email created in the macro between the two building block macros.

%SEND_EMAIL_TEMPLATE MACRO

SEND_EMAIL_TEMPLATE is only an example of a structure for the final macro created and run on a study. The following example SEND_EMAIL makes use of this structure directly. Starting with the RECIPIENT macro which leads onto the FILENAME call and ODS HTML options with email contents and finishing with EMAIL.

```
%macro send_email_template(id=,subject=);
%recipient; %* Requires ID macro variable input;
%debug; %* Prints to log what macro variables resolve to;

filename emailcon "%sysfunc(pathname(work))\send_email_template.html"; %* FILENAME
call;
ods html file=emailcon options(pagebreak='no'); %* ODS HTML to create email contents;

%* Email content goes here;
```

```
ods html close;

%email; %* %EMAIL requires macro variable EMAIL_TO generated by %recipient and macro
variable SUBJECT;
%mend send_email_template;
```

SEND_EMAIL MACRO

This is a simple email that can be sent using SAS, it is the basic example of sending an email and though this example may not be an email a study lead would send in practice, it does showcase some nice features such as side by side tables and multiple recipients.

One simple practical adaptation of this code would be to create and send a study induction email for new team members joining a study. This would also serve to test that the new programmers' details have been correctly added into the TEAM dataset. The macro call is as follows:

```
%send_email(id=greg1 brian1, subject=PHUSE Paper SM02 Test Email);
```

TABLE 4 – %SEND_EMAIL EMAIL

PHUSE Paper SM02 Test Email Inbox ×

phuse2020emailtest@gmail.com
to me

Hi Greg, Brian,

This is an example email for my PHUSE Paper - Automation of Processes for Programming Study Leads and has been sent using SAS!

The first table below is an example of a programming plan you may re ad into this program
The second table below is an example of a TEAM dataset

Kind regards,
Fred

dataset_name	prog_name	first_line_id	qc_id	first_line_ran	qc_ran
DM	sdtm_dm_prog.sas	Bob1	Greg1	2020-04-25	2020-04-26
ADSL	adsl_prog.sas	Bob1	Greg1	2020-04-25	2020-04-26
ADAE	adae_prog.sas	Brian1	Greg1	2020-04-25	2020-04-26

id	name	email	blind	role
Bob1	Bob	phuse2020emailtest@gmail.com	blinded	statistician
Greg1	Greg	phuse2020emailtest@gmail.com	unblinded	programmer
Brian1	Brian	phuse2020emailtest@gmail.com	unblinded	data_manager
Greg2	Gregory	phuse2020emailtest@gmail.com	blinded	programmer

Note: Code can be found in Appendix C.

OUTSTANDING_ISSUES_EMAIL MACRO

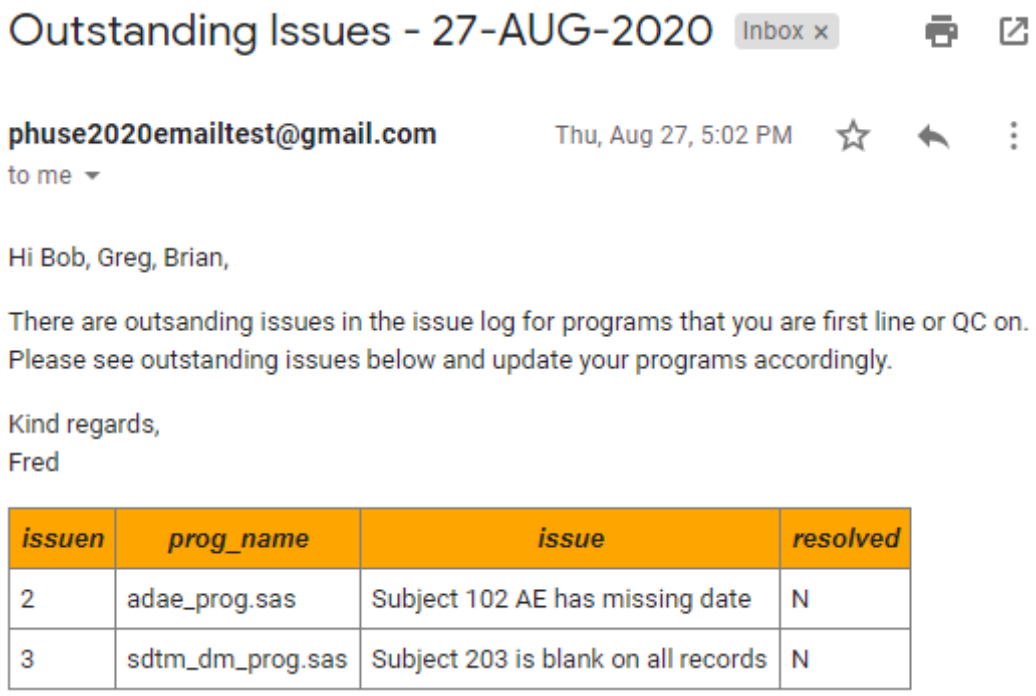
A common time draining task faced by programming leads can be following up unresolved issues in an issue log. Having to cross reference both an issue log and programming plan to work out who the responsible programmers are for a given program can soon add up to a lot of time. This is the ideal example of something that can be automated with macros.

Looking at the email below there is a dynamic first line created in the macro to pick out each name of authors/front line programmers and QCers uniquely. Following this there is a generic body of text and finally ending on a printout of the outstanding issues. This may not be practical on some studies to email every responsible programmer for every outstanding issue in a single email, but the macro is easily customisable. It is simple to create an email per program or per programmer with small changes to the code. Dynamic subjects for emails created using macros are important to differentiate each email sent, an example of which can already be seen in the email subject where todays date has been added to the email subject.

Though this is a practical example the macro leans more towards showcasing features than an absolute be all and end all Outstanding Issue email macros to use on any study. It is heavily encouraged to adapt any macro found in this paper.

```
%outstanding_issues_email(subject=Outstanding Issues - &todaysdate.);
```

TABLE 5 – OUTSTANDING ISSUES EMAIL



Outstanding Issues - 27-AUG-2020 Inbox ×

phuse2020emailtest@gmail.com Thu, Aug 27, 5:02 PM

to me

Hi Bob, Greg, Brian,

There are outsanding issues in the issue log for programs that you are first line or QC on.
Please see outstanding issues below and update your programs accordingly.

Kind regards,
Fred

issuen	prog_name	issue	resolved
2	adae_prog.sas	Subject 102 AE has missing date	N
3	sdtm_dm_prog.sas	Subject 203 is blank on all records	N

Note: Code in Appendix D.

LOG_ISSUES_EMAIL MACRO

During the review of programs on a study, checking the logs are clean is an important step. Log checker macros are frequently available to any programmer in some form. Time consuming programming lead tasks stemming from this are that an issue is identified in a program log and that then authors need to be notified to clean.

LOG_ISSUES_EMAIL macro requires that SAS program name is entered into the program option. From this it will use a simplistic log checker that looks for occurrences of “ERROR” and “WARNING” and print these into a dataset. This will then be included in the email that is sent. This is only slightly modified version of the OUTSTANDING_ISSUES_EMAIL macro. Additional features included in this are more dynamic elements in the email subject and body text.

When using this macro practically it is recommended to use a full log checker. It could be further automated to run on every SAS program name found in the programming plan.

```
%log_issues_email(program=adsl_prog);
```

TABLE 6 – LOG ISSUES EMAIL

Log Issues in ADSL - 23-OCT-2020 Inbox ×

phuse2020emailtest@gmail.com
to me

Hi Bob, Greg,

There are outsanding issues in adsl_prog.sas log. Please see outstanding issues below and clean adsl_prog.sas accordingly.

Kind regards,
Fred

ERROR: Dummy Error
WARNING: Dummy Warning

Note: Code in Appendix F.

ADDITIONAL FEATURES & IDEAS

Below are additional features that with some straightforward adaptations of code can be created using the macros showcased here. Due to how similar they would be to programs found already in the paper they have not been programmed.

TABLE 7 – ADDITIONAL FEATURES AND POTENTIAL MODIFICATIONS	
Ready for QC/ Not QC passed/ QC Failed	A program to be run at end of the day or start of day. For example, "Ready for QC": 1) Check if program is set to QC ready in the programming plan 2) Send emails to QCer to notify
Status of Datasets/ Status of TFLs/ Status of Tasks	A program that could be run daily/weekly/fortnightly. 1) Take Dataset/TFL/Task Name and Status columns from programming plan 2) Send to relevant team members for example other managers on the project
Team Availability with Outstanding Tasks	1) Other projects and holidays cross referenced with number of tasks outstanding in project. 2) Could be setup to be sent to resourcing department/project management to secure additional resources or identify potential upcoming resource unavailability.

DO IT ALL AGAIN IN R?

SAS is not the only language in which emails can be sent. All the concepts raised and explored in this paper can easily be implemented with R. As an example, using packages such as gmailR to send emails in Gmail as demonstrated in the macros.

CONCLUSION

As a programming study lead time is a precious commodity and automation is a great tool to utilise programming knowledge into making better use of this time. The code found in this paper is to showcase features and ideas for email automation using SAS. Hopefully this will act as inspiration for developing your own macros with many more expanded features above and beyond explored in this paper.

REFERENCES

SAS EMAILS

Sending E-mail from the DATA step Erik W. Tilanus, consultant, Driebergen, the Netherlands
https://support.sas.com/resources/papers/proceedings/pdfs/sgf2008/038-2008.pdf

Snail Mail to Emails: Generating Auto-Emails from SAS® with Attachments Crystal Carel, MPH, Baylor Scott & White Health, Dallas, TX https://support.sas.com/resources/papers/proceedings17/1060-2017.pdf

SMTP E-Mail Access Method: Hints, Tips, and Tricks Chuck Hunley, SAS Institute Inc., Cary, NC
https://support.sas.com/resources/papers/proceedings10/060-2010.pdf

R EMAILS

How to Send an Email in R by Atur Hebda https://blog.mailtrap.io/r-send-email/

ACKNOWLEDGMENTS

Thanks to; Pradeep Kumar, Jessica Whittaker-Dixon & Ravi Adla for sharing their experiences as study leads; Ridda Mahmood for her initial research into SAS emails; Mr B.P. Cat for all his help and support.

APPENDIX A: SETUP OF EMAIL ADDRESS AND DEMONSTRATION FILES

```
/** SM02 PHUSE Paper Program Start **/
proc datasets library=WORK kill;
    run;
quit;

ods html close;
*** Email login setup to send emails from ;
options emailhost=("smtp.gmail.com" port=587 STARTTLS auth=plain

/* Gmail address */
id="phuse2020emailtest@gmail.com"

/* Can encode PW with PROC PWENCODE */
pw="");

*** Team dataset for team details;

data team;
    length id $10. name $20. email $50. blind $10. role $50.;
    input id $ name $ email $ blind $ role $;
    cards;
Bob1 Bob phuse2020emailtest@gmail.com blinded statistician
Greg1 Greg phuse2020emailtest@gmail.com unblinded programmer
Brian1 Brian phuse2020emailtest@gmail.com unblinded data_manager
Greg2 Gregory phuse2020emailtest@gmail.com blinded programmer
;
run;

*** Programming plan dataset to contain details of program names program authors and
QCers;

data prog_plan;
    length dataset_name prog_name $50. first_line_id qc_id $10.;
    input dataset_name $ prog_name $ first_line_id $ first_line_ran :yymmdd10. qc_id $
qc_ran :yymmdd10.;
    format first_line_ran qc_ran yymmddd10.;
    cards;
DM sdtm_dm_prog.sas Bob1 2020-04-25 Greg1 2020-04-26
ADSL adsl_prog.sas Bob1 2020-04-25 Greg1 2020-04-26
ADAE adae_prog.sas Brian1 2020-04-25 Greg1 2020-04-26
;
run;

*** Issue log dataset containing issues, datasets affected.;

data issue_log;
    infile cards delimiter=',';
    length issuen 8. affected $50. issue $200. resolved $1.;
    input issuen affected $ issue $ resolved $;
    cards;
1,adsl_prog.sas,Subject 101 looks to have incorrect age,Y
2,adae_prog.sas,Subject 102 AE has missing date,N
3,sdtm_dm_prog.sas,Subject 203 is blank on all records,N
4,sdtm_dm_prog.sas adsl.sas,Subject 203 is blank on all records,Y
```

```
;
run;
```

APPENDIX B: BUILDING BLOCK MACROS

```
*** Scope of macro variables;
%global name_to email_to email_list where;
*** Macro that will send email to &subject and ;
* Requires EMAIL_TO macro variable. Can be generated by recipient macro;

%macro email;
    filename myemail EMAIL to=(&email_to.) subject="&subject."
        content_type="text/html";

    data _null_;
        file myemail;
        infile emailcon;
        input;
        put _infile_;
    run;

    filename emailcon clear;
    filename myemail clear;
%mend email;

*** Selection from TEAM dataset for name and email addresses for where to send email;
* Requires ID to be an assigned macro variable. If more than one ID then each ID
should be seperated by a space;

%macro recipient;
    %* Upcase ID for processing;
    %let up_id=%upcase(%quote(%sysfunc(catt(&id.))));
    %* If ID is equal to the following keywords whole team or whole subsection of
        team members will be selected;

    %if &up_id. eq WHOLE_TEAM %then
        %let where=  ;
    %else %if &up_id. eq BLINDED %then
        %let where=where catt(upcase(blind))=catt(upcase("&id."));
    %else %if &up_id. eq UNBLINDED %then
        %let where=where catt(upcase(blind))=catt(upcase("&id."));
    %else %if &up_id. eq STATISTICIAN %then
        %let where=where catt(upcase(role))=catt(upcase("&id."));
    %else %if &up_id. eq PROGRAMMER %then
        %let where=where catt(upcase(role))=catt(upcase("&id."));
    %else %if &up_id. eq DATA_MANAGER %then
        %let where=where catt(upcase(role))=catt(upcase("&id."));
    %* If one ID is listed then select only that ID;
    %else %if %sysfunc(countw(&up_id.)) eq 1 %then
        %let where=where catt(upcase(id))=catt(upcase("&id."));
    %* If more than one ID is listed and it is not a keyword then create EMAIL_LIST
macro variable of ;
    %else %if %sysfunc(countw(&up_id.)) gt 1 %then
        %do;

            data _null_;
                length temp $2000;
                temp=cat('"', tranwrd(catt(upcase("&id.")), " ", '" "'), '"');
                call symput('email_list', temp);
            run;

            %let where=where catt(upcase(id)) in(&email_list.);
        %end;
```

```
    %* Print to log if invalid options for ID;
    %else
        %put %str(WARN)ING: Invalid value for ID option;
    %* Create NAME_TO and EMAIL_TO macro variables for use in email text and for
determining recipients of generated email;

    proc sql noprint;
        select name into :name_to separated by ", " from team
          &where.;
        select quote(catt(email)) into :email_to separated by " " from team
          &where.;
    quit;

%mend recipient;


%macro debug;
    %* Debug Macro to print macro variables to log;

    %if %symexist(email_list) %then
        %put WARNING: email_list = &email_list.;

    %if %symexist(id) %then
        %put WARNING: UP_ID resolves to %upcase(%quote(%sysfunc(catt(&id.))));

    %if %symexist(name_to) %then
        %put WARNING: Name_to resolves to: &name_to.;

    %if %symexist(email_to) %then
        %put WARNING: Email_to resolves to: &email_to.;

    %if %symexist(where) %then
        %put WARNING: Where = &where.;
%mend debug;
```

APPENDIX C: SEND_EMAIL MACRO

```
%macro send_email(id=, subject=);
    %recipient;
    %debug;
    filename emailcon "%sysfunc(pathname(work))\send_email.html";
    ods html file=emailcon options(pagebreak='no');
    ods html text="<html><p>Hi &name_to.,</p> <p>This is an example email for my
PHUSE Paper - Automation of Processes for Programming Study Leads
and has been sent using SAS!
<br><br>The first table below is an example of a programming plan
you may read into this program. <br>The second table below is an
example of a TEAM dataset</p>
<p>Kind regards,<br>Fred</p></html>";
    ods layout gridded columns=2;
    ods region;

    proc print data=prog_plan style(table)={just=l}
  style(header)=[font_style=italic background=dark green]
  noobs;
    run;

    ods region;

    proc print data=team style(table)={just=l}
  style(header)=[font_style=italic background=dark green]
  noobs;
    run;
```

```
    ods layout end;
    ods html close;
    %email;
%mend send_email;

%send_email(id=greg1 brian1, subject=PHUSE Paper SM02 Test Email);
```

APPENDIX D: OUTSTANDING_ISSUES_EMAIL MACRO

```
%macro outstanding_issues_email(subject=);
    %* Resolve ID based on outstanding issues;

    data outstanding;
        set issue_log;
        where catt(resolved)='N';
        rename affected=prog_name;
    run;

    proc sort data=outstanding out=outstanding_sort;
        by prog_name;
    run;

    proc sort data=prog_plan out=prog_plan_sort;
        by prog_name;
    run;

    %* Merge PROG_PLAN with outstanding issues in log to determine authors of
        affected programs;

    data authors;
        merge prog_plan_sort outstanding_sort(in=o);
        by prog_name;

        if o;
    run;

    %* Select distinct First Line and QC programmers and read into ID macro
        variable for RECIPIENT macro;

    proc sql ;
        select distinct z into :id separated by " " from ((select distinct
            first_line_id as z from authors) union (select distinct qc_id as z from
            authors));
    quit;

    %recipient;
    %debug;
    filename emailcon "%sysfunc(pathname(work))\outstanding_issues_email.html";
    ods html file=emailcon options(pagebreak='no');
    ods html text="<html><p>Hi &name_to.,</p>
<p>There are outsanding issues in the issue log for programs that you are first line
or QC on.
Please see outstanding issues below and update your programs accordingly.</p>
<p>Kind regards,
<br>Fred</p></html>";

    proc print data=outstanding style(table)={just=l}
  style(header)=[font_style=italic background=orange]
  noobs;
    run;

    ods html close;
    %let todaysDate = %sysfunc(today(), date11.);
```

```
    %email;
%mend outstanding_issues_email;

%outstanding_issues_email(subject=Outstanding Issues - &todaysdate.);
```

APPENDIX E: SEND_EMAIL_TEMPLATE MACRO

```
%macro send_email_template(id=, subject=);
    %recipient;
    %* Requires ID macro variable input;
    %debug;
    %* Prints to log what macro variables resolve to;
    filename emailcon "%sysfunc(pathname(work))\send_email_template.html";
    %* Simple FILENAME call;
    ods html file=emailcon options(pagebreak='no');
    %* ODS HTML to create email contents;
    %* Email content goes here;
    ods html close;
    %email;
    %* %EMAIL requires macro variable EMAIL_TO generated by %recipient and macro
        varible SUBJECT;
%mend send_email_template;
```

APPENDIX F: LOG_ISSUES_EMAIL MACRO

```
%macro log_issues_email(program=);
    %* Create prog_output as the character string preceding the first underscore
        specified in program=option;
    %let prog_output=%upcase(%scan(&program., 1, '_'));
    %* Point towards location of study logs;
    filename proglog "%sysfunc(pathname(work))\&program..log";
    ;
    %*Create Dummy Log file for us to scan;

    data _null_;
        file proglog;
        put "This is a dummy &prog_output. LOG for demonstration purposes.";
        put "ERROR: Dummy Error";
        put "WARNING: Dummy Warning";
    run;

    %* Scanning only for ERROR and WARNING here for simplicity.;
    %* It is recommended to use a log checker macro and print flagged lines to LOG
        dataset;

    data log;
        infile proglog missover pad;
        input line $50.;

        if index(upcase(line), "ERROR") or index(upcase(line), "WARNING");
    run;

    %* Pick out Authors for Firt Line Program and QC from prog_plan;

    data authors;
        set prog_plan;

        if prog_name="&program..sas";
    run;

    %* Select distinct First Line and QC programmers and read into ID macro
        variable for RECIPIENT macro;
```

```
    proc sql ;
        select distinct z into :id separated by " " from ((select distinct
            first_line_id as z from authors) union (select distinct qc_id as z from
            authors));
    quit;

    %recipient;
    %debug;
    filename emailcon "%sysfunc(pathname(work))\log_issues_email.html";
    ods html file=emailcon options(pagebreak='no');
    ods html text="<html><p>Hi &name_to.,</p>
<p>There are outsanding issues in &program..sas log.
Please see outstanding issues below and clean &program..sas accordingly.</p>
<p>Kind regards,
<br>Fred</p></html>";

    proc report data=log noheader nocenter;
    run;

    ods html close;
    %let todaysDate = %sysfunc(today(), date11.);
    %let subject=Log Issues in &prog_output. - &todaysDate.;
    %email;
%mend log_issues_email;

%log_issues_email(program=adsl_prog);
```

APPENDIX G: TABLES

TABLE 1 – TEAM DATASET SPECIFICATION	
ID	A unique ID is relatively easy as all users will have unique login credentials so simply using usernames of team members is the obvious choice here. A nice bonus is the TEAM dataset can then be used to look up each members login for granting permissions to certain areas of a study.
EMAIL	Email addresses are the most obvious example needed to be able to send emails to the team.
NAME	The name is used to address recipients of emails. Unlike the unique ID this does not need to be unique as it will mainly be used for creating text strings for email body text. For example, “Dear Jay, Fred…”.

TABLE 2 TEAM DATASET EXAMPLE

id	name	email	blind	role
Bob1	Bob	phuse2020emailtest@gmail.com	blinded	statistician
Greg1	Greg	phuse2020emailtest@gmail.com	unblinded	data_manager
Brian1	Brian	phuse2020emailtest@gmail.com	unblinded	programmer
Greg2	Gregory	phuse2020emailtest@gmail.com	blinded	data_manager

TABLE 3 – PROGRAMMING PLAN IN SAS

dataset_name	*prog_name*	*first_line_id*	*qc_id*	*first_line_ran*	*qc_ran*
DM	sdtm_dm_prog.sas	Bob1	Greg1	2020-04-25	2020-04-26
ADSL	adsl_prog.sas	Bob1	Greg1	2020-04-25	2020-04-26
ADAE	adae_prog.sas	Brian1	Greg1	2020-04-25	2020-04-26

TABLE 4 – %SEND_EMAIL EMAIL

PHUSE Paper SM02 Test Email Inbox ×

phuse2020emailtest@gmail.com
to me

Hi Greg, Brian,

This is an example email for my PHUSE Paper - Automation of Processes for Programming Study Leads and has been sent using SAS!

The first table below is an example of a programming plan you may re ad into this program
The second table below is an example of a TEAM dataset

Kind regards,
Fred

dataset_name	*prog_name*	*first_line_id*	*qc_id*	*first_line_ran*	*qc_ran*
DM	sdtm_dm_prog.sas	Bob1	Greg1	2020-04-25	2020-04-26
ADSL	adsl_prog.sas	Bob1	Greg1	2020-04-25	2020-04-26
ADAE	adae_prog.sas	Brian1	Greg1	2020-04-25	2020-04-26

id	*name*	*email*	*blind*	*role*
Bob1	Bob	phuse2020emailtest@gmail.com	blinded	statistician
Greg1	Greg	phuse2020emailtest@gmail.com	unblinded	programmer
Brian1	Brian	phuse2020emailtest@gmail.com	unblinded	data_manager
Greg2	Gregory	phuse2020emailtest@gmail.com	blinded	programmer

TABLE 5 – OUTSTANDING ISSUES EMAIL

Outstanding Issues - 27-AUG-2020 Inbox ×

phuse2020emailtest@gmail.com Thu, Aug 27, 5:02 PM

to me

Hi Bob, Greg, Brian,

There are outsanding issues in the issue log for programs that you are first line or QC on. Please see outstanding issues below and update your programs accordingly.

Kind regards,
Fred

issuen	*prog_name*	*issue*	*resolved*
2	adae_prog.sas	Subject 102 AE has missing date	N
3	sdtm_dm_prog.sas	Subject 203 is blank on all records	N

TABLE 6 – LOG ISSUES EMAIL

Log Issues in ADSL - 23-OCT-2020 Inbox ×

phuse2020emailtest@gmail.com

to me

Hi Bob, Greg,

There are outsanding issues in adsl_prog.sas log. Please see outstanding issues below and clean adsl_prog.sas accordingly.

Kind regards,
Fred

ERROR: Dummy Error
WARNING: Dummy Warning

TABLE 7 – ADDITIONAL FEATURES AND POTENTIAL MODIFICATIONS	
Ready for QC/ Not QC passed/ QC Failed	A program to be run at end of the day or start of day. For example, “Ready for QC”: 3) Check if program is set to QC ready in the programming plan 4) Send emails to QCer to notify
Status of Datasets/ Status of TFLs/ Status of Tasks	A program that could be run daily/weekly/fortnightly. 3) Take Dataset/TFL/Task Name and Status columns from programming plan 4) Send to relevant team members for example other managers on the project
Team Availability with Outstanding Tasks	3) Other projects and holidays cross referenced with number of tasks outstanding in project. 4) Could be setup to be sent to resourcing department/project management to secure additional resources or identify potential upcoming resource unavailability.