PharmaSUG 2023 - Paper MM-040

Gear up the Metadata – Automating Patient Profile Generation, a Metadata Driven Programming Approach

Tanmay Khole, Aman Sharma, Lili Li, PTC Therapeutics, Inc.

Durga Prasad Chinthapalli, TechObserver

ABSTRACT

Patient profiles are individual reports of subject's clinical data and provide a great benefit to clinical or medical teams when performing an on-going review in a clinical trial project. These reports are customized per reviewer's requests when safety, efficacy, and other key significant data points vary for each study design. The programming challenges for generating patient profile reports include the accommodation of variances in data source, structure, mapping, quality, and the customization in report outputs and formats. The application featured in this paper is an MS Excel VBA based utility featuring a user-friendly interface that uses SAS® macros in the backend with a robust design to analyze all clinical trials data in CDSIC SDTM or ADaM format and generate patient profiles by GUI. The SAS® macros are designed by implementing metadata-driven programming approach where the users only need to use the metadata specification and the VBA based utility to generate patient profiles without the core programming conventionally required.

INTRODUCTION

Patient profiles are individualized displays of key datapoints for each patient. They are sometimes equated to case report form tabulations (Ritter, 2011). However, they are, and should be, more than case report form tabulations. Patient profiles are more customizable to include safety, efficacy, and other key significant data points of clinical interests. With this more focused scope, patient profiles usually include, but not limited to, only key SDTM domains (i.e., LB, AE, VS, CM, MH) and critical ADaM data sets. Moreover, patient profiles serve a wider range of purposes for different teams. For instance, data management team can use the patient profile reports to monitor the data integrity; safety team can utilize the patient profile reports to oversee the safety of ongoing treatments; medical writing team can refer to patient profile reports to generate narratives; or, the clinical team or medical monitors can get a general overview of the ongoing study from data points reported in the patient profiles (Conover, 2011; Fahmy, 2006). Last but not the least, patient profiles can be more illustrative than simple case report form tabulations as patient profiles can include graphs to visualize the patterns and trends.

While patient profiles are beneficial as discussed above, the generating of patient profile reports can be challenging and time-consuming. The first challenge is that the statistical programmers need to take the variances of data sources, structures, mapping, and data quality into account when they produce the reports. For instance, some most commonly used data sets for patient profiles include DM, AE, CM, VS, and LB domains from SDTM model. Among these domains, some of them are simple by-subject structure (i.e., DM), while others are more complex per measurement per time point per visit per subject structure (i.e., VS, LB). Moreover, when ADaM data sets or even EDC are requested into patient profile reports (Desai & Collins, 2015), it introduces even more complexity into the programming logics and techniques. This will require the production programming to be holistic yet flexible enough to preserve multidimensional

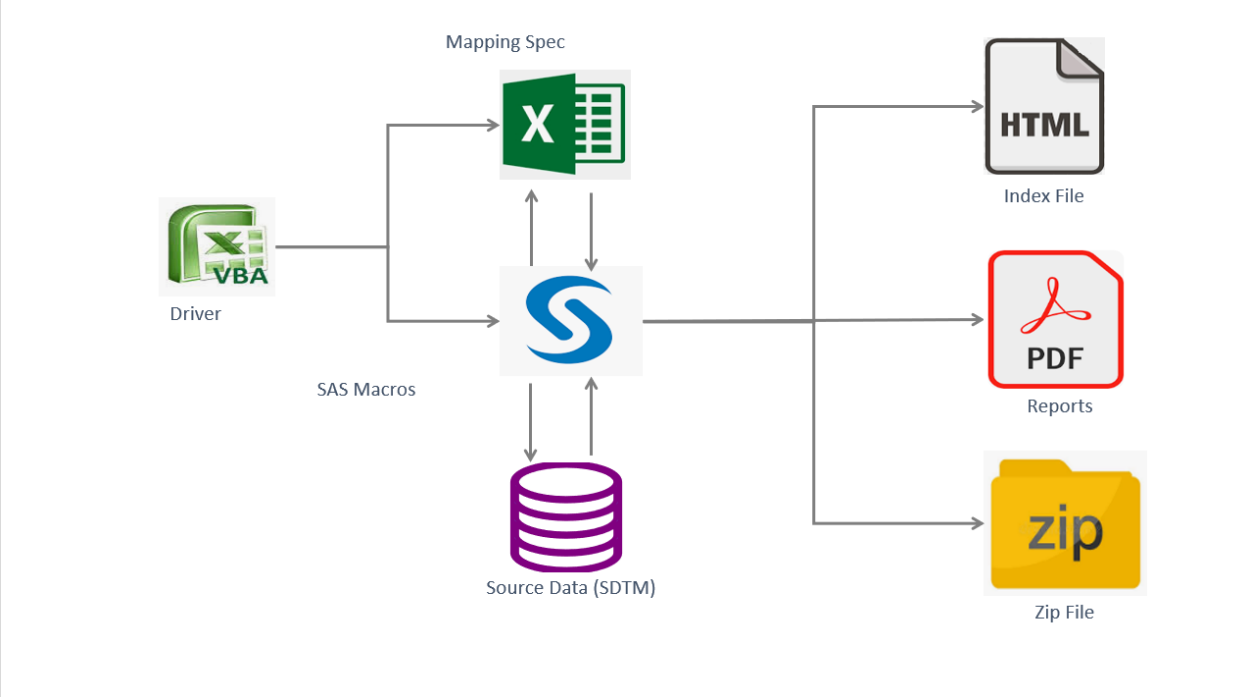
data from different data sources for each subject. The second challenge in patient profile programming is to make reports more readable and effective in delivering the information in various output formats. For instance, highlights of changes from the last data extraction are preferred. While various sorting orders is another common request to make it more effective from different teams who want to check different patterns of their interest.

This paper will introduce an easy-to-use application that can smoothly tackle the challenges mentioned above, providing a one-click solution for patient profile production using metadata. By utilizing a GUI based on MS Excel VBA, the SAS macros in the application will automatically run in the backend when the corresponding patient profile specification is properly set up and the driver calls the macro to execute. In another words, instead of requiring advanced SAS programming knowledge and techniques, this application requires minimal knowledge of SAS programming and maintenance from the end-users and can be adopted by a much broader team of users.

OVERVIEW OF THE APPLICATION

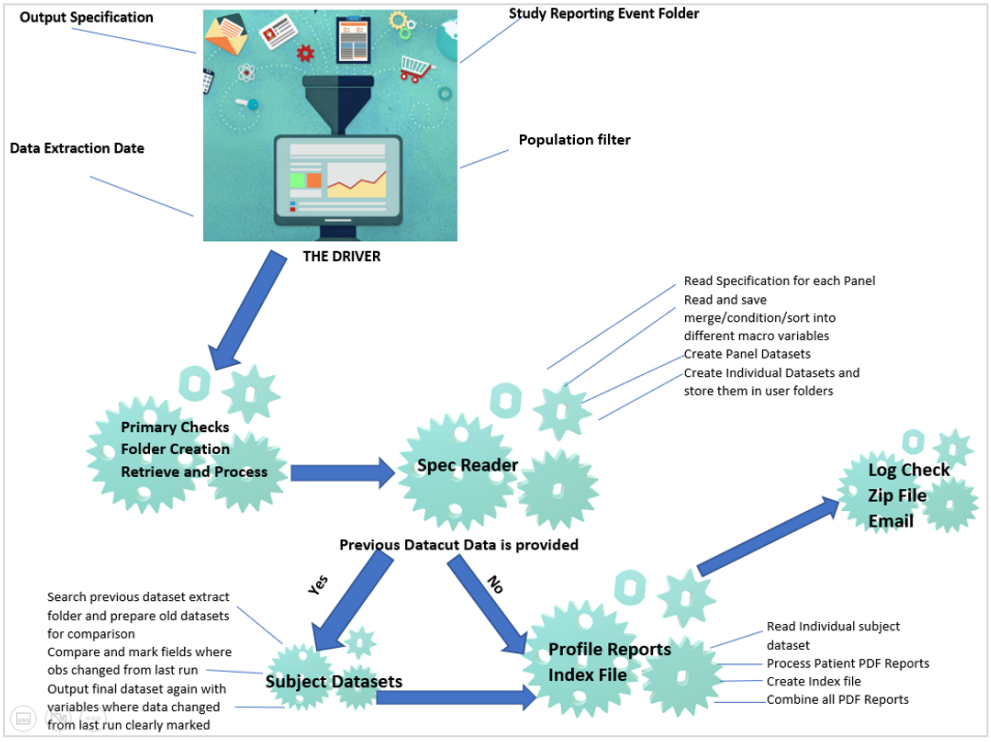
The production of patient profiles consists of multiple working blocks in the application. It starts with specification file on the MS Excel Macro-Enabled Workbook file (XLSM file) where the VBA driver is also located. The VBA driver serves to call the SAS macros in the program folder to be executed.

As the working block diagram shown in Display 1, the driver (GUI) is the central control that corelates the mapping specification and the SAS macros where the specification file locates and annotates what data points to be included from the source data set (i.e. SDTM data) into the final output that the SAS macro need to produce. The SAS macro will generate the individual reports in PDF format for each subject. In addition, an index file with hyperlink to each individual file and a zip file with combined reports will also be generated automatically. In addition to reports & zip file generation, automatic notification emails to inform the start and end of execution will also be sent to the user of this application.



Display 1. Software Used in Development

Display 2 provides information for data processing flow through various modules in this application. In the initiating stage at the Driver, output specification, data extraction date, population filter, and the working folder location are read in and passed along to the preparation module where primary check of the dataset location and folder creation/confirmation is performed. When the primary information is ready, the spec reader macro module parses and identifies panel information and data points to produce the subject datasets and PDF outputs before the last stage where those outputs are combined into the final reports package with an index file in a zipped file.



Display 2. Process Flow

SPECIFICATION FILE

The specification file is embedded in an MS Excel Macro-Enabled Workbook file (XLSM file). The end-users, including users without extensive knowledge of SAS, can select their desired data points and set up the panels directly in the XLSM file, which later can be readily imported by the programmers into SAS to be processed without any special transcription or editing. In this process, both end-user side and programmer side can track and maintain the project in one place, thus having better traceability.

For the specification file component, there are two separate sheets for different purposes: the panel specifications, and the header/footnotes specifications.

To set up the panel specifications it follows a simple formatting rule. Take the example below as an illustration. The first column is the panel information column – Panel 1 for subject stratification information. In this panel, five data points will be reported, in green font. The first row is the output panel label row. The second row is the input source data row. In this example, panel 1 source data is from SDTM DM domain, while the prior therapy lines data point is referring to PRIORLIN variable in the DM domain. Similarly, the rest of the data points are mapped from DM domain with the respective variables. The output of panel 1 can be seen in display 3.

Panel 1 – Stratification Information	Number of Prior Therapy Lines	Diagnosis Cancer Type	Number of Prior Systemic Therapies	ECOG PS Score	Histological Tumor Type
SDTM.DM	DM.PRIORLIN	DM.CANTYP	DM.STRAT1	DM.STRAT2	DM.STRAT3

Table 1. Sample Panel Specification

Panel 1 - Stratification Information				
Number of Prior Therapy Lines	Diagnosis Cancer Type	Number of Prior Systemic Therapies	ECOG PS Score	Histological Tumor Type
2	Other	1	0	Nonuterine LMS

Display 3. Sample Panel Output

While for the header specifications, it follows a fixed template where the requesters only need to enter what labels and corresponding variable names are needed. For example, to produce the following header format (Display 4), the specifications following the sample are displayed in Table 2.

Display 4. Sample Report Header

In the header specification template, there are configuration columns for left, center, and right sections respectively. For each section, there are three rows of content can be configured. Take the sample below as an illustration: The Subject ID will be displayed in the upper right corner, with the displayed label “Subject ID:” and the corresponding SDTM variable “DM.SUBJID”. Similarly in the middle row (title_num = 3), left, center, and right sections will respectively display this subject’s country and site ID information, the study protocol information, and the basic demographic information as chosen by the request (in this case, age, sex, and race). Of course, if you want to include more information in the header section, you can utilize the lower row (title_num = 4) for additional reference data points.

Title_Num	Text_Left	Value_Left	Text_Center	Value_Center	Text_Right	Value_Right
2					Subject ID:	SDTM.DM. SUBJID
3	Country/Site ID:	SDTM.DM.COUNTRY/SDTM.DM.SITEID	Study ID:	SDTM.DM. STUDYID	Age/Sex/Race:	SDTM.DM.AGE|/|SDTM.DM.SEX|/|SDTM.DM.RACE
4						

Table 2. Sample Header Specification

STANDARDIZED DATA MAPPING RULES

The example above is the most basic type where the data points are from a source data set with no filter or merging needed. When it comes to more customization, the mapping rules for the patient profile specification are also simple and straightforward. There are only four mapping key words needed in the specification that can be identified and executed by the SAS macros later:

KEY WORD	PURPOSE
~where	Filter out information needed
~mergevar	Denote variables to be used as merging keys
~sortvar	Customize sorting order (can have more than one sorting variable in one panel)
|	Concatenate multiple data points in one cell

Table 3. Mapping rules in Specification

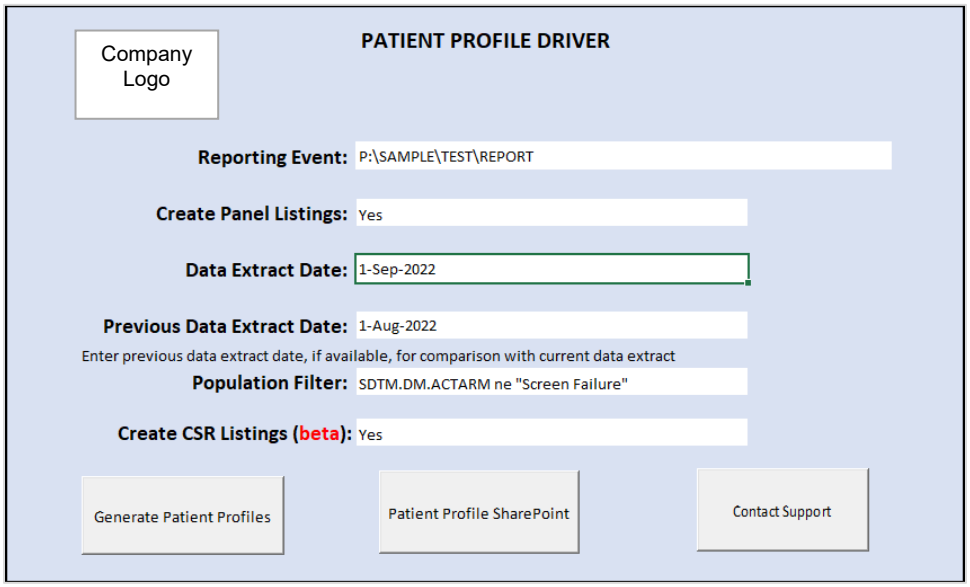
With these four customization key words, more complex data manipulation can be accommodated. Below is an implementation example where an occurrence data set is converted from vertical structure into horizontal data structure while keeping only the reported data points in the output and sorted by the exam visit date as requester desired.

Panel 1 - ECG	Visit	ECG Date	Overall Interpretation	PR Interval (msec)
SDTM.EG	EG.VISIT ~mergevar	EG.EGDTC ~sortvar1~mergevar	EG.EGSTRESC ~where EGTESTCD="INTP"	EG.EGSTRESC ~where EGTESTCD="EGPR"

Table 4. Mapping Customization Example

PATIENT PROFILE DRIVER

The Patient Profile Driver is a GUI based on the MS Excel VBA for the end-users. As shown in the screenshot below, the Driver includes a form format designed to capture necessary input from users to generate the patient profile reports, and the click-button section to execute and operate the utility.



Display 5. VBA Interface as the Patient Profile Driver

Details of the form component and the functions of button options are presented in Table 5:

Type/Category	Description	Required
Reporting Event	Field to point the location of study programming area for a deliverable, example CSR or IA.	Yes
Create Panel Listings	Option to create panel data in excel format for all subjects. The excel file will contain	No
Data Extract Date	Data extraction date of EDC or Cutoff date	Yes
Previous Data Extract Date	Prior data extraction date use for comparison purpose to highlight data changes/updates	No
Population Filter	Filter to apply conditions to select subject population	No
Create CSR Listings	Converts panels into subject listings per CSR format	No

Table 5. Driver Form Details

Button	Description
Generate Patient Profile	Button to start utility
Patient Profile SharePoint	Button to open the SharePoint webpage consisting of information regarding patient profile utility, user documentation, FAQs, etc.
Contact Support	Button to open a message box listing out contact details of SME group

Table 6. Driver Button Details

With this information, the output is automatically saved in designated sub-folders by default. The “Data Extraction Date” is required not only for annotation purpose in the report outputs, but also for the application to automatically detect any immediate recent data set in this same report folder and generate a comprehensive comparison in the output where any changes or newly added data values are highlighted, thus, making it easier for the reviewers to detect changes if they need to compare from previous report.

The other four menu items on the driver are all optional. For the “Create Panel Listings” and “Create (CSR) Listings” options, there are drop-down list of Yes/No to choose from. The panel listings are individual listings for each designated panel for each subject. They are SAS data set format, and mostly for QC or programmer review purpose. The CSR Listings, on the other hand, are combined listings of all the subjects per each panel in the rtf format that can readily be used as CSR appendix items. The “Previous Data Extract Date” is for the comparison not with most immediate recent data cut. For instance, one might want to compare the current data cut to a milestone data cut in the further past instead of the most immediate recent data cut. Lastly, the option of “Population Filter” is to provide the flexibility of including only desired sub-set of the study population. For instance, as shown above, the screen failure subjects will be excluded from the report.

Once the configuration is completed, by clicking the “Generate Patient Profiles” button on the driver as shown above, a well-organized patient profiles report is generated.

VBA Code for GUI

VBA coding syntax is different from SAS programming language, but the code below is easy to follow. For instance, there are five major functions to achieve at the VBA module here. First task is to properly identify the deliverable folders:

```
RE_FolderName = Worksheets("Driver").Range("D8").Value
PP_FolderName = RE_FolderName & "\" & "patient_profile"
PP_FolderExists = Dir(PP_FolderName, vbDirectory)
```

Once the deliverable folders are identified, key working folder locations and reporting structure is configured:

```
Sub MakeMyFolder(orgFld As String)

       Dim fdObj As Object
       Application.ScreenUpdating = False
       Set fdObj = CreateObject("Scripting.FileSystemObject")

       If fdObj.FolderExists(orgFld) Then
           'MsgBox "Found it.", vbInformation
       Else
           fdObj.CreateFolder (orgFld)
           'MsgBox "It has been created.", vbInformation
       End If

       Application.ScreenUpdating = True
End Sub

MakeMyFolder (PP_FolderName)
MakeMyFolder (PP_FolderName & "\data")
MakeMyFolder (PP_FolderName & "\data\adam")
MakeMyFolder (PP_FolderName & "\data\sdtm")
```

```
MakeMyFolder (PP_FolderName & "\data\sdtmplus")
MakeMyFolder (PP_FolderName & "\output")
MakeMyFolder (PP_FolderName & "\pgm")
MakeMyFolder (PP_FolderName & "\spec")
```

When the infrastructure configuration is set up as shown above, the SAS macro modules to execute the actual production of profile reports are called, and the final batch-run is performed: Below is the VBA code to call subroutines to create SAS programs and execute them in batch mode.

```
Call Createcallpp(batflnm:=PP_FolderName & "\pgm\call_pp.sas",
folderLoc:=PP_FolderName & "\pgm")
Call Createcallmchklog(batflnm:=PP_FolderName & "\pgm\call_mchklog.sas",
folderLoc:=PP_FolderName & "\pgm")
Call CreateAfile(batflnm:=PP_FolderName & "\pgm\_runall.bat",
folderLoc:=PP_FolderName & "\pgm")
execBat (PP_FolderName & "\pgm")
```

In the following section, the key SAS macro modules that execute the actual production of profile reports will be discussed in more detail.

PRIMARY PROGRAMMATIC CHECKS

As shown in Display 2, primary check is the first step in preparation stage. These are the checks which prevent the running of entire code if an error occurred because of the user entries in the specifications. Such checks save time and inform the user of the correction(s) needed for successful execution of the macro.

The code section below checks that the library and the dataset specified exist for code processing. We use two functions to realize the purpose here: The LIBREF function checks for the library and the EXIST function checks for the dataset.

```
%macro chk_lib(Lib=,DSN=);
  %let LIBEXISTS=0;
  %IF %SYSFUNC(LIBREF(&LIB)) = 0 %THEN %Let LIBEXISTS=1; ;

  %Let DSNEXISTS=0;
  %IF %SYSFUNC(EXIST(&DSN)) > 0 %THEN %Let DSNEXISTS=1;
  %Put 1=YES, 0=NO;
  %Put &LIB. EXISTS? &libexists;
  %Put &DSN. EXISTS? &dsnexists;
%mend chk_lib;
```

The next check is the panel numbering. Every panel should have a unique number for it and is a positive integer in sequential order beginning from 1. Below is the code section to check for the duplicate panel numbering and aborts the run if the same panel number is repeated more than once: The data set *pp_spec* is the imported data set based off the specification spreadsheet introduced above. The SCAN function extracts the panel number from the panel header and the FREQ procedure finds the count of each panel number. The count > 1 indicates a repetition of panel number and triggers an error message to log and abort the program execution.

```
data pp_spec1;
  set pp_spec(rename=(&_rnm_vars.));
  where &col1_name. ne '';
  &col1_name. = strip(&col1_name.);
  if index(upcase(&col1_name.), "PANEL") > 0 then
   grpnum =  scan(&col1_name.,2);
run;


proc freq data=pp_spec1 noprint ;
table grpnum / out=chk_panel(where=(count>1));
  where grpnum ne '';
run;

proc sql noprint;
  select max(grpnum) into: grpnum_chk
  from chk_panel
  ;
quit;
```

Below section of code checks for the sequence of the panel number that must start from 1. A loop is run starting from 1 to the maximum of the panel number. As the loop runs, it checks for the panel number corresponding to the index variable value and assigns a macro variable *gnum_err*. If the index variable value does not match with the current panel number, it means the numbers are not in sequence and the macro variable *gnum_err* is set to missing which is a trigger to abort the macro.

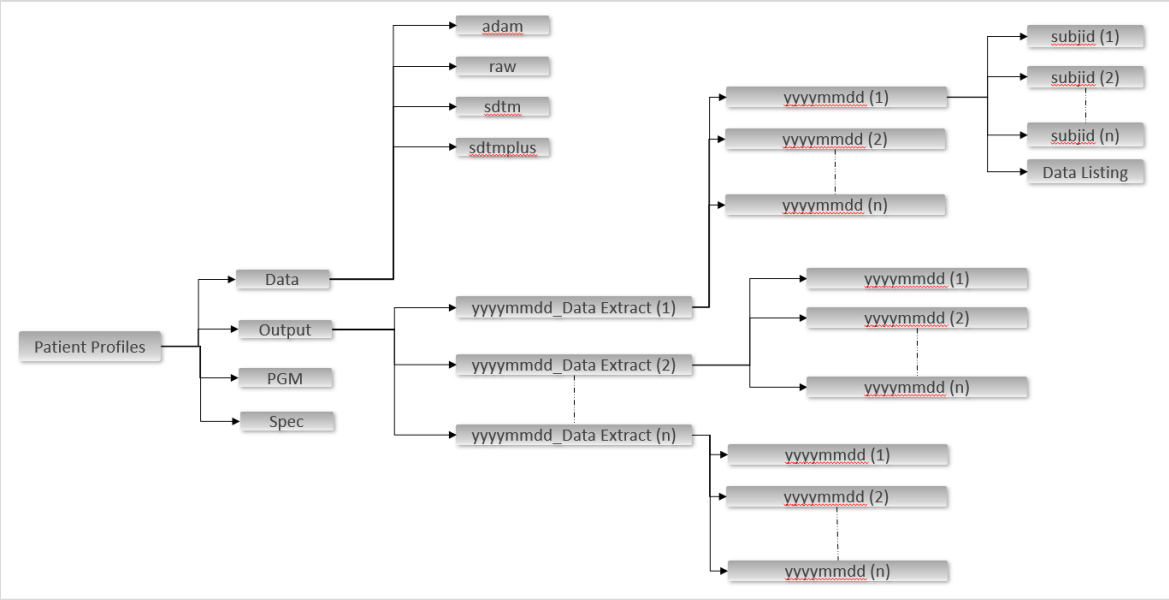
```
%do j =1 %to &max_gnum. ;
%put J iteration = &j. ;
data panel&j.;
set pp_spec2;
  where gnum = &j.;
  rownum=_n_;
  if rownum=2 then
    lib = substr(&col1_name., 1, index(&col1_name., ".")-1);
run;

proc sql noprint; select max(gnum) into: gnum_err from panel&j.; quit;

%if &gnum_err < 1 %then %do;
  %put ERROR: User has not specified the panel numbers in sequential order
from 1.;
  %put User Note: Update spec to sequential panel number from 1.;
  %put User Note: Macro will now EXIT.;
  %goto exit;
%end;
%end;
```

AUTOMATED FOLDER CREATION

This module of the tool focuses on creation of the multiple folders when the tool is initiated. When the user clicks on “Generate Patient Profiles” button and “Reporting Event” path is specified in the Patient Profile Driver GUI, the module works by detecting the path from the working folder to determine if a “Patient Profile” folder exists, VBA code snippet and subroutine are provided in VBA Code for GUI and appendix section respectively. If the folder does not exist, the application will create one with proper sub-folders included. If the “Patient Profile” folder is detected, but no dated output sub-folder exists under the deliverable folder path, this application will also automatically create a sub-folder using the data extraction date. Towards the end when the tool is ready to output the Individual subject profile data, individual folders for each subject will also be created under the dated output sub-folder.



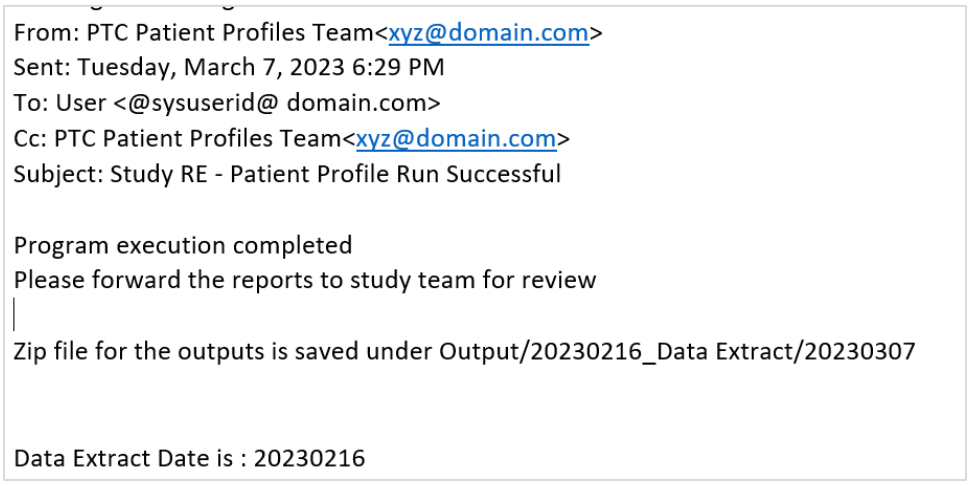
Display 6. Automated Folder Structure Generated by the Application

```
%macro your_macro(dir);
   %let rc = %sysfunc(filename(fileref, &dir.));
   %if %sysfunc(fexist(&fileref)) %then %do;
      options dlcreatedir;
      libname crtdir "&projloc.\output\&currexdt._Data Extract\&fdate";
   %end;
   %else %do;
      options dlcreatedir;
      libname mandir "&projloc.\output\&currexdt._Data Extract";
      libname crtdir "&projloc.\output\&currexdt._Data Extract\&fdate";
    %end;
  %put syserr = &syserr.;
%mend your_macro;
```

EMAIL FUNCTIONALITY

Patient profile application has a feature to send out emails to user when the utility starts and ends. This helps to avoid the need of continuous monitoring while the application is processing the data, automatic email notifications will be sent to the users and identified key stakeholders: one at the initiating timepoint to notify the users with beginning of patient profiles production, while the other at the end of the execution to report possible errors that came up during the process or to notify successful production of a patient profiles report for a request. Sample codes for producing automatic email notification with log error identified as attachment are shared below:

```
proc options group=email; run;
  options
  emailsys=smtp
  emailhost=mail.domain.com;
  filename msg email
  to="&sysuserid@domain.com"
  from="&sysuserid@xyz.com"
  subject = "&projfold - Patient Profile Run Started";
  data _null_;
    file msg;
    put "Waiting on Program execution completion.”
    Put “Notification email of completion will follow.";
    put "Data Extract Date is: &currexdt";
    %if %length(&prevexdt) > 0 %then %do;
       put "Previous Data Extract Date is: &prevexdt";
    %end;
run;
```


From: PTC Patient Profiles Team<xyz@domain.com>
Sent: Tuesday, March 7, 2023 6:29 PM
To: User <@sysuserid@ domain.com>
Cc: PTC Patient Profiles Team<xyz@domain.com>
Subject: Study RE - Patient Profile Run Successful

Program execution completed
Please forward the reports to study team for review

Zip file for the outputs is saved under Output/20230216_Data Extract/20230307

Data Extract Date is : 20230216

Display 7. Sample Email Notification of Successful Run

RETRIEVE AND PRE-PROCESS DATA

Though non-standardized data gives more flexibility, it comes with a cost of additional data processing as they vary from study to study and sometimes even the EDC system changes the process. Due to these limitations, standardized data format i.e., SDTM and ADaM are preferred. Using standardized datasets, the same application can be employed at different studies or even in different therapeutics areas. Therefore, this approach significantly cuts the time and resources needed to produce the set of patient profile reports for any request if the standardized data sets are available.

This module accesses the SDTM and ADaM datasets stored in the project folders and copy them into the deliverable folder where a series of steps are performed for datasets to be ready for later use in the process. Moreover, the utility combines all the SDTM supplemental domains with the parent domain and converts all numeric values across SDTM and ADaM datasets into Character variables to ease data formatting and creation of the outputs.

```
%macro Convert_SDTM_Num_2_Char;
%let i=1;
%do %while(%scan(&_ds,&i) ne );
  data _null_;
  set sashelp.vcolumn(keep=libname memname name type where=(libname='SDTM'
and memname="%upcase(%scan(&_ds,&i))")) end=last;
  if _n_ eq 1 then call execute("proc sql; create table
sdtm_w."||strip(memname)||" as select ");
  if type='char' then call execute(name);
  else call execute('put('||strip(name)||',best32. -l) as '||strip(name));
  if not last then call execute(',');
  else call execute("from SDTM."||strip(memname)||";quit;");
run;
%let i=%eval(&i+1);
%end;
%mend;
```

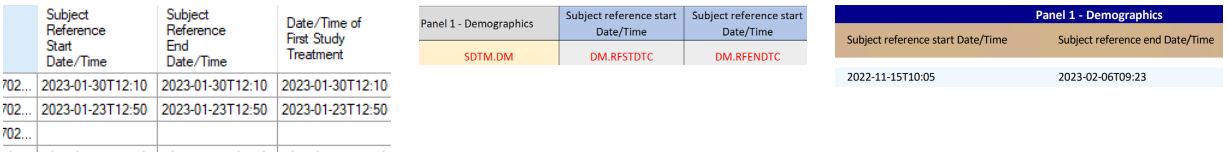
SPEC READER MACRO

Spec reader macro is designed to read in the mapping specifications from an excel file. The specification follows standardized mapping rules, mentioned in table 3., and to identify and parse the specifications per each panel, the spec reader macro categorizes all panels into four types based on the panel contents and mapping keywords. Some of these mapping keywords include special words such as “Merge/Sort/Where” which are read from the specifications and stored into the SAS macro variables. These 3 macro variables and the rules surrounding these 3 variables help us identify the Panels and process/display the data accordingly. There are also features like concatenation of fields (identified using special reserved characters for concatenation), multiple where statements, sorting of a variable using numeric equivalent and sort by ascending/descending features that have also been implemented for display needs. Below sections describe in detail how each type of Panel is processed along with screenshots on how the data for each is mapped and displayed by the tool. The table below shows the keywords and conditions that are checked for each panel before the data is processed.

Panel Types	Conditions to Identify
Panel 1	WHERE and MERGE clauses in the specification for this Panel type should not exist
Panel 2	Where Clause can exist and there should be no MERGE Clause in the specification for this Panel type. All the variables declared for this Panel should have a WHERE clause
Panel 3	Where Clause can exist and there should be no MERGE Clause in the specification for this Panel type. Not all the variables declared for this Panel should have a WHERE clause
Panel 4	Both MERGE and WHERE should be present in the specification for this Panel Type

Table 7 – Conditions to Identify Panel Types

Panel Type 1: The data displayed in panel type 1 is a one record panel and has exact structure as source SDTM or ADaM data. The data points displayed in the panel do not have any keywords specified for merging or any conditions. Below is an example of data for Panel 1 example Demographics; where the information in source data is presented as One Record per Subject and will not need any Merge/Sort conditions. When the panel type is identified the module works with storing the Display label for the whole panel and its variables and then retrieves the source data from the specified SDTM/ADaM.



	Subject Reference Start Date/Time	Subject Reference End Date/Time	Date/Time of First Study Treatment
702...	2023-01-30T12:10	2023-01-30T12:10	2023-01-30T12:10
702...	2023-01-23T12:50	2023-01-23T12:50	2023-01-23T12:50
702...			

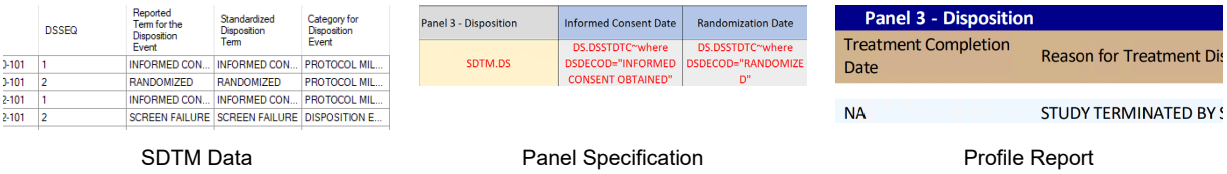
Panel 1 - Demographics	Subject reference start Date/Time	Subject reference start Date/Time
SDTM.DM	DM.RFSTDTC	DM.RFENDTC

Panel 1 - Demographics	
Subject reference start Date/Time	Subject reference end Date/Time
2022-11-15T10:05	2023-02-06T09:23

SDTM Data | Panel Specification | Profile Report

Display 8. Panel Type 1

Panel Type 2: The data displayed in panel type 2 is a single record panel; but the source data structure is vertical and converted to horizontal layout to display the data points required for the panel. Each data point has a specific condition to select the required value. The data points are selected per specified conditions and USUBJID is used as a default variable to merge all the data points together to form a panel dataset. Example: Disposition Status, where SDTM.DS has a vertical structure per standard SDTM rules and panel layout designed for this application has a horizontal layout. The identification for this panel occurs when the specification is processed and WHERE identified for all columns required for the respective panel. When the panel type is identified the module works with storing the Display label for the whole panel and its variables and then retrieves the source data from the specified SDTM/ADaM



	DSSEQ	Reported Term for the Disposition Event	Standardized Disposition Term	Category for Disposition Event
0-101	1	INFORMED CON...	INFORMED CON...	PROTOCOL MIL...
0-101	2	RANDOMIZED	RANDOMIZED	PROTOCOL MIL...
2-101	1	INFORMED CON...	INFORMED CON...	PROTOCOL MIL...
2-101	2	SCREEN FAILURE	SCREEN FAILURE	DISPOSITION E...

Panel 3 - Disposition	Informed Consent Date	Randomization Date
SDTM.DS	DS.DSSTDTC~where DSDECOD="INFORMED CONSENT OBTAINED"	DS.DSSTDTC~where DSDECOD="RANDOMIZED"

Panel 3 - Disposition	
Treatment Completion Date	Reason for Treatment Dis
NA	STUDY TERMINATED BY S

SDTM Data | Panel Specification | Profile Report

Display 9. Panel Type 2

Panel Type 3: The data displayed in panel type 3 is a multiple record panel and has exact structure as the source SDTM or ADaM data. The difference between panel 1 and panel 3 is the number of records displayed in the panel and the sorting feature which can be implemented in panel type 3 to sort the panel by data points specified in the panel. Similar to Panel 2 and Panel 1 the Module first works with identifying the Panel type which is identified by source data containing more than one record per subject and by identifying the specification file for a Sort variable. A restriction placed on this module like Panel Type 1 is that there cannot be any merge variables assigned to this Panel Type.

	AESEQ	Reported Term for the Adverse Event	Lowest Level Term	AELLTCD	Dictionary-Derived Term	AEPTCD
103	1	FOOD POISONI...	Food poisoning	10016952	Food poisoning	10016952
101	1	MILD LOW BAC...	Post procedural p...	10049749	Procedural pain	10064882
101	1	DIARRHEA	Diarrhea	10012727	Diarrhoea	10012735
101	2	RHINO/SINUSIT...	Rhinosinusitis	10052106	Sinusitis	10040753
102	1	HEADACHE	Headache	10019211	Headache	10019211
102	2	FLU	Flu	10016790	Influenza	10022000
102	3	NECK PAIN	Neck pain	10028836	Neck pain	10028836

Panel 2 - Adverse Events	Adverse Event ID	Reported Term	Dictionary Term
SDTM.AE	AE.AESEQ	AE.AETERM	AE.AEDECOD

Panel 2 - Adverse Events	
Dictionary Term	Start Date/End Date/Causality
Diarrhoea	2022-11-04 / 2022-11-09 / NOT RELATED
Sinusitis	2023-01-05 / 2023-01-06 / NOT RELATED

SDTM Data Panel Specification Profile Report

Display 10. Panel Type 3

Panel Type 4: Panel Type 4 requires Merge variables besides USUBJID to be included in panel. Each subsequent columns/variables (with or without conditions) will be merged by the specified variables. USUBJID is considered as a default MERGE variable. The spec reader macro starts with identifying the merge variables to enter the processing for this Panel Type and creates independent interim datasets for each column consisting of merge variables and column contents with any condition specified, if any. These interim independent datasets are merged to form panel 4 datasets which get processed by the reporting macro. The example below shows how a multiple record per subject Vitals dataset is mapped and how the final output is displayed.

	VSSEQ	Vital Signs Test Short Name	Vital Signs Test Name	Result or Finding in Original Units	Original Units
101	1	DIABP	Diastolic Blood Pr...	65	mmHg
101	2	OXYSAT	Oxygen Saturation	97	%
101	3	PULSE	Pulse Rate	72	beats/min
101	4	RESP	Respiratory Rate	20	breaths/min
101	5	SYSBP	Systolic Blood Pr...	111	mmHg

Panel 4 - Vital Signs	Visit Date	Visit	Height (cm)
SDTM.VS	VS.VSDTC~mergevar	VS.VISIT~mergevar	VS.VSSTRESC~where VSTESTCD="HEIGHT"

Panel 4 - Vital Signs		
BMI (kg/m2)	Systolic BP (mmHg)	Diastolic BP (mmHg)
NA	110	74
NA	NA	NA
NA	121	71

SDTM Data Panel Specification Profile Report

Display 11. Panel Type 4

SUBJECT DATASETS

After the Spec Reader Macro is executed, this module outputs individual subject datasets for each panel and stores them in a sub- folder created for each individual subject with the subject ID number as the sub-folder name. The data stored in these sub-folders is imported for the creation of the final outputs; in the meantime, they also serve as future reference purposes for comparison or review, if needed. The codes below are executed when the user entered Previous Data Extract Date; the module searches for a folder under the output directory with the previous extract date and checks if datasets exist. If the datasets exist under the folder user mentioned in the Driver, the tool prepares those datasets for comparison with the datasets created using Spec Reader Macro. The code below shows how to read contents of all the files and folders within a directory.

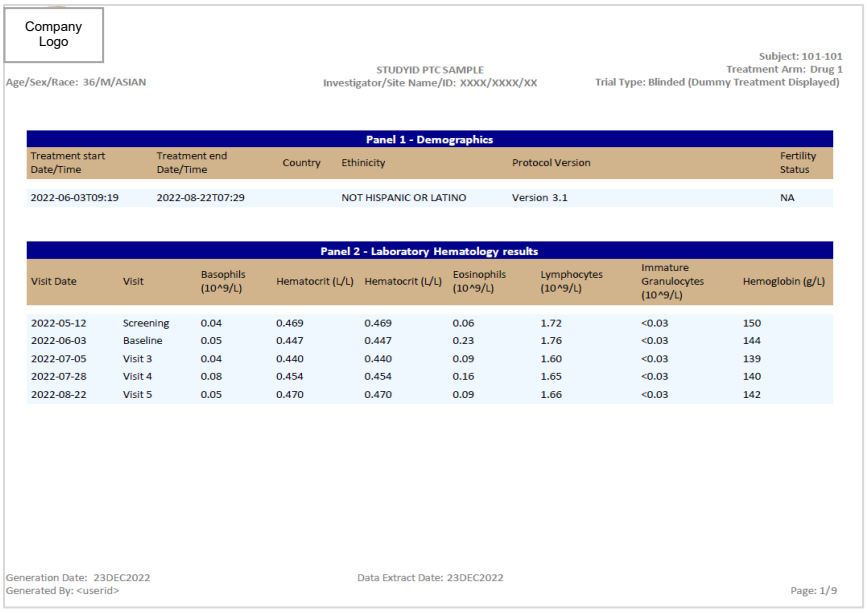
```
%macro getds();
data dirs_found  files_found (compress=no);
  rc = filename(fref, path);
  if rc = 0 then do;
    did = dopen(fref);
```

```
    rc = filename(fref);
  end;
  else do;
    length msg $200.;
    msg = sysmsg();
    putlog msg=;
    did =.;
  end;
  if did <= 0 then do;
    putlog 'ERR' 'OR: Unable to open ' Path=;
    return;
  end;
  dnum = dnum(did);
    do i = 1 to dnum;
    filename = dread(did, i);
    fid = mopen(did, filename);
   FileType = prxchange('s/.*\.{1,1}(.*)/$1/', 1, filename);
    if filename = filetype then filetype = ' ';
      output files_found;
    end;
    else do;
      root = catt(path, "\", filename);
     output dirs_found;
    end;
  end;
  rc = dclose(did);
run;
%mend;
%getds;
```

Once the combined and individual datasets are stored in the output folders, this SAS module proceeds to check the configuration set at the driver if the user has entered the previous data extract date. If an input is made at the driver, it scans for a folder under the name of previous data extract date, and, if found, combine the datasets from the previous data folder for each panel. It would then identify the sort order of the current datasets created for the creation of Patient Profile, and search if those sorting variables exist in the reports of the designated previous data cut. If similar sorting variables are found from both reports, automatic comparison and highlight in the current patient profiles reports dataset will be annotated for any identified changes or addition of new data points.

PROFILE REPORTS

This SAS module is to output the final reports in PDF format for individual subjects. There are two key steps in the process. One, proper header information is read from the annotations sheet in the XLSM file that stores all the patient profile specification and related information. See more details about the header spec as in Table 2 above and the sample output for header section as in Display 4. Step two, patient profiles in PDF output are produced for each subject. To achieve this step, the PDF production macro reads panel datasets in a loop for each subject. The macro checks whether the reports are being generated for the first time. If the reports are available for previous run, the macro checks for the difference with previous extract which are generated during the preparation of subject datasets and highlights the value-level changes, see Display 13 for the value-level changes identified. Sample code how the value level changes are color codes is shared in the appendix.



Company Logo

Age/Sex/Race: 36/M/ASIAN

STUDYID PTC SAMPLE
Investigator/Site Name/ID: XXXX/XXXX/XX

Subject: 101-101
Treatment Arm: Drug 1
Trial Type: Blinded (Dummy Treatment Displayed)

Panel 1 - Demographics

Treatment start Date/Time	Treatment end Date/Time	Country	Ethinicity	Protocol Version	Fertility Status
2022-06-03T09:19	2022-08-22T07:29		NOT HISPANIC OR LATINO	Version 3.1	NA

Panel 2 - Laboratory Hematology results

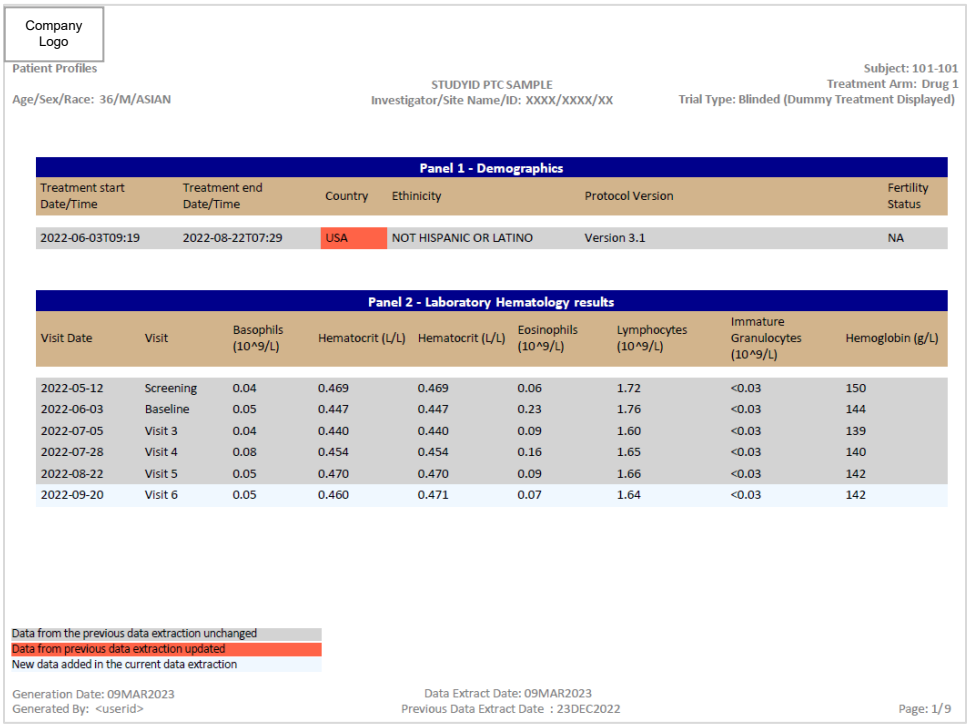
Visit Date	Visit	Basophils (10^9/L)	Hematocrit (L/L)	Hematocrit (L/L)	Eosinophils (10^9/L)	Lymphocytes (10^9/L)	Immature Granulocytes (10^9/L)	Hemoglobin (g/L)
2022-05-12	Screening	0.04	0.469	0.469	0.06	1.72	<0.03	150
2022-06-03	Baseline	0.05	0.447	0.447	0.23	1.76	<0.03	144
2022-07-05	Visit 3	0.04	0.440	0.440	0.09	1.60	<0.03	139
2022-07-28	Visit 4	0.08	0.454	0.454	0.16	1.65	<0.03	140
2022-08-22	Visit 5	0.05	0.470	0.470	0.09	1.66	<0.03	142

Generation Date: 23DEC2022
Generated By: <userid>

Data Extract Date: 23DEC2022

Page: 1/9

Display 12. Sample Patient Profile Output for first run



Company Logo

Patient Profiles

Age/Sex/Race: 36/M/ASIAN

STUDYID PTC SAMPLE
Investigator/Site Name/ID: XXXX/XXXX/XX

Subject: 101-101
Treatment Arm: Drug 1
Trial Type: Blinded (Dummy Treatment Displayed)

Panel 1 - Demographics

Treatment start Date/Time	Treatment end Date/Time	Country	Ethinicity	Protocol Version	Fertility Status
2022-06-03T09:19	2022-08-22T07:29	USA	NOT HISPANIC OR LATINO	Version 3.1	NA

Panel 2 - Laboratory Hematology results

Visit Date	Visit	Basophils (10^9/L)	Hematocrit (L/L)	Hematocrit (L/L)	Eosinophils (10^9/L)	Lymphocytes (10^9/L)	Immature Granulocytes (10^9/L)	Hemoglobin (g/L)
2022-05-12	Screening	0.04	0.469	0.469	0.06	1.72	<0.03	150
2022-06-03	Baseline	0.05	0.447	0.447	0.23	1.76	<0.03	144
2022-07-05	Visit 3	0.04	0.440	0.440	0.09	1.60	<0.03	139
2022-07-28	Visit 4	0.08	0.454	0.454	0.16	1.65	<0.03	140
2022-08-22	Visit 5	0.05	0.470	0.470	0.09	1.66	<0.03	142
2022-09-20	Visit 6	0.05	0.460	0.471	0.07	1.64	<0.03	142

Data from the previous data extraction unchanged
Data from previous data extraction updated
New data added in the current data extraction

Generation Date: 09MAR2023
Generated By: <userid>

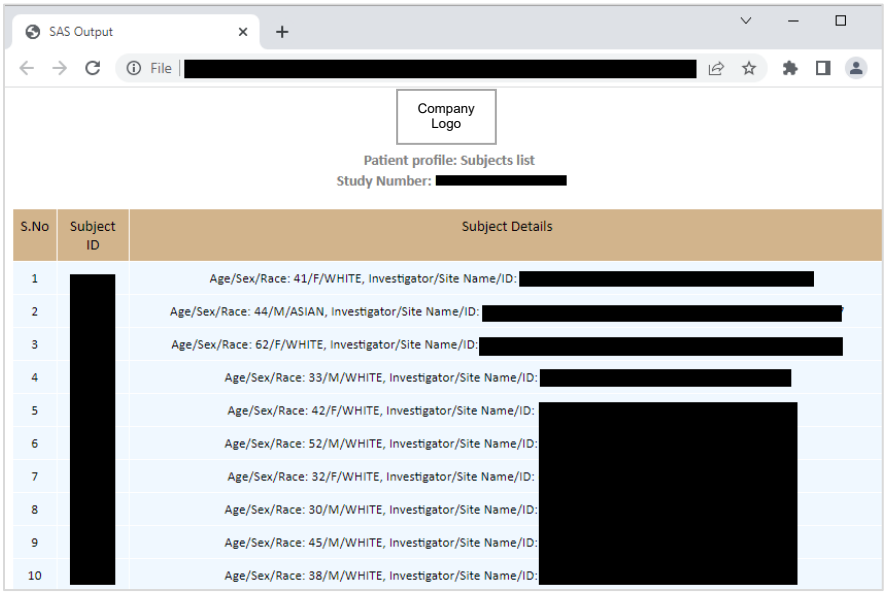
Data Extract Date: 09MAR2023
Previous Data Extract Date : 23DEC2022

Page: 1/9

Display 13. Sample Patient Profile Output with value level changes

INDEX FILE

An automatic index file in HTML format is created at the end of the output production stage. This HTML file serves quick reference with clickable hyperlink directing to each corresponding patient profiles PDF output for each subject (see display as demonstration below).



Display 14. Sample Index HTML File

To create this index file, a simple PROC REPORT will serve the purpose with a proper ODS HTML output set (use ODS HTML PATH to set up the output file path and name).

ZIP FILE CREATION

Once all the reports outputs and final output dataset are produced. The log checker macro is executed to review log for errors. With no errors during the tool run, all the reports in PDF format, index file in HTML format and mapping spec in XLSX format available in the current folder are compressed into zip file for ease of delivery to reviewers/stakeholders. The code below gives an overview of how the portion of the module works when the path of the files and the name is provided to the zip file creation.

```
%macro createZIP(path, archive_name, archive_path);
   %put *** Creating an archive (&archive_path\&archive_name) ***;
   ods package(newzip) open nopf;
   %readCatalog(&path)
   ods package(newzip) publish archive properties
     (archive_name="&archive_name"
      archive_path="&archive_path");
   ods package(newzip) close;
%mend createZIP;
```

CONCLUSION

Utilizing this application reduces the time & resource for programming efforts and provides the clinical/medical team the reports containing critical data points to facilitate data anomalies identification that are related to safety of the patient.

As automation is becoming critically important and the focus on removing redundancy from everyday tasks, this paper introduces a very user-friendly tool for patient profile generation that can save significant amounts of time and resources for such frequent request in any clinical trial. The tool leverages a user's experience in writing specifications and removes the need for the user to write different programs for such frequent requirements for different Clinical Trials. By innovatively utilizing the MS Excel VBA interface, robust SAS macros, mapping keywords, along with leveraging clinical trials data available in standardized format, minimal SAS programming knowledge and skill are required from the users, thus, making it a truly easy-to-use and one-click-away application for all parties who want to generate patient profile reports for various purposes and focuses of their interest.

REFERENCES

Conover, W. (2011). "Creating Hyperlinked PDF Graphical Patient Profiles with PROC REPORT", *PharmaSUG 2011*, Paper TU01. Nashville, TN; SAS Users Group. Available at www.lexjansen.com/pharmasug/2011/TU/PharmaSUG-2011-TU01.pdf

Desai, P. and Collins, R. (2015). "Patient Profile: A Menu-Driven System", *PharmaSUG 2015*, Paper AD13. Orlando, FL; SAS Users Group. Available at https://www.lexjansen.com/pharmasug/2015/AD/PharmaSUG-2015-AD13-SAS.pdf

Fahmy, A. (2006). "Patient Profile: A Menu-Driven System", *PharmaSUG 2006*, Paper PO19. Bonita Spring, FL; SAS Users Group. Available at https://www.lexjansen.com/pharmasug/2006/Posters/PO19.pdf

Ritter, A. (2011). "Creating Customized Patient Profiles using SAS ODS RTF and PROC TEMPLATE", *PharmaSUG 2011*, Paper TT03. Nashville, TN; SAS Users Group. Available at https://www.lexjansen.com/pharmasug/2011/TT/PharmaSUG-2011-TT03.pdf

Harrington (2013) "Beep, Beep, Beep, Back It Up! A Fool Proof Approach to Archiving with no Copying": PharmaSUG 2013 – CC02
https://www.pharmasug.org/proceedings/2013/CC/PharmaSUG-2013-CC02.pdf

ACKNOWLEDGMENTS

The authors on this paper would like to thank our manager Steven Huang of PTC Statistical Programming team for his constant support and helpful advice throughout this project and beyond.

CONTACT INFORMATION

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

Tanmay Khole
PTC Therapeutics
tkhole@ptcbio.com

Aman Sharma
PTC Therapeutics
amansharma@ptcbio.com

Durga Prasad
PTC Therapeutics
durga.prasad@ptcbio.com

Lili Li
PTC Therapeutics
lli@ptcbio.com

APPENDIX

APPENDIX A: VBA Subroutine to Create New Folders.

```
Sub MakeMyFolder(orgFld As String)
    Dim fdObj As Object
    Application.ScreenUpdating = False
    Set fdObj = CreateObject("Scripting.FileSystemObject")

    If fdObj.FolderExists(orgFld) Then
        'MsgBox "Found it.", vbInformation
    Else
        fdObj.CreateFolder (orgFld)
        'MsgBox "It has been created.", vbInformation
    End If

    Application.ScreenUpdating = True
End Sub
```

APPENDIX B: Code to show the color for value level changes in data compared to previous data extract. The keywords ‘update’ and ‘new’ are added to the panel data during the comparison with previous data extract.

```
%do l=1 %to &varcount.;
%let vark = %scan(&varlist.,&l,%str( ));
compute &vark.;
            if _&vark._  eq 'update' then do;
call define("&vark.", "style", "style=[background=red]");
end;
            if lowcase(_chg_) eq 'new’ then do;
call define(_row_, "style", "style=[background=lightgrey]");
end;
endcomp;
%end;
```

APPENDIX C: Report code to create HTML index file with hyperlinks.

```
ods html path = "<file location>" file = "<file name>.html" style = custom;

proc report data = subj_data nowd

      style(report)={frame=void rules = none just=center}

style(header) = {backgroundcolor = &table_header_color.};

column pg lnum _chg_ _subjid_var desc;

define _subjid_var. /order flow 'Subject ID' style(column) = {just = c}
style(header) = {just = c};

define lnum  /order 'S.No' style(column) = {just = c} style(header) = {just =
c};

define desc  /order 'Subject Details' flow style(column) = {just = c}
style(header) = {just = c};

define _chg_ / display noprint;

define pg / order noprint;


*** The compute block below turns the value of subjid into a HTML link.;

compute &_subjid_var.;

href="./"||trim(_subjid_var.)||".pdf";

call define(_col_, "URL", href);

            if lowcase(_chg_) = 'update' then do;

call define(_row_, "style", "style=[background=red]");

end;

else if _chg_ = 'new' then do;

call define(_row_, "style", "style=[background=lightgrey]");

end;

endcomp;

run;
```