SAS® Data Query and Edit Checks Web 2.0

Sy Truong, Meta-Xceed, Inc. (MXI), Fremont, CA
Carey Smoak, Roche Molecular Systems, Inc., Pleasanton, CA

ABSTRACT

Querying data used to require sophisticated algorithms and programming with SQL expertise or SAS data step techniques. Now, there are many user-friendly graphical tools to build expressions without having to program code to access relational databases. However, there are few tools for creating queries from a web interface upon SAS data. This paper describes a real world project where data managers who extract data from a relational database and transfer it to SAS data. Data managers need a user-friendly query tool to verify the cleanliness of the resulting SAS data. Since data managers are not SAS programmers, the web interface eases the learning curve allowing them to perform tasks such as: creating compound query expressions, deriving temporary imputed variables, saving and combining saved queries. These are some of the methods described in this paper as it explains how to use SAS data step, HTML and XML to accomplish an effective user experience. The initial target audience for this project was to empower data managers to perform tasks only SAS programmers were able to do. However, the user-friendly interface extends it to a larger audience of non-techie power users.

QUERY DATA IN AGE OF GOOGLE

Querying and cleaning data has traditionally been performed by data managers and analysts but as HTML5 and web based interfaces deliver more user friendly solutions; it is opening up data access to wider audience. The underlying querying technology such as SQL and SAS DATA STEP has remained the same for many years; but what has changed recently is how it is becoming easier to access. This paper describes the implementation of a query tool for clinical data and shows how it also represents a microcosm for the larger task of searching for information from a vast database. This is analogous to how Google changed the way we search for information. The search engine giant makes it easy to apply a search by providing dynamic and user friendly web interface on top of a sophisticated search engine. This paper will describe Syview which utilizes the same approach in identifying the most common query and data cleaning tasks and deliver this capability through a dynamic and user friendly web interface.

HTML5 standards are still being finalized by the standard body World Wide Web Consortium (W3C), but that does not stop developers from pushing the technology to provide dynamic web solutions today. Even proprietary technology vendors such as Adobe Flash and Microsoft Silverlight recognize the power of HTML5 and are developing solutions with HTML5. The software described in this paper implements some of these technologies but does it in a way that is transparent to the user. Users that are accustomed to performing searches on Google, buying a product on Amazon, or sharing with friends on Facebook do not need to know the underlying technology. Rather, they use these websites and services with ease. It is this same approach that this paper will show how these technologies can empower users with tools to view and query SAS data.

FUNCTIONAL REQUIREMENTS

The specific project which drove the development of Syview was established when the manager at a multi-national Pharmaceutical company noticed that SAS programmers were spending a lot of time writing one-off SAS scripts to query SAS datasets after it has been exported from the relational database. The request came from different sources including data managers who are skilled with SQL and tools used with the relational database, but are not SAS programmers. This presented an opportunity to have a solution for users who are SAS novices, yet allowing them to perform the same queries through a user friendly interface. Some of the functionality that was requested by these

users includes the following:

1. **Browser Access –** The ability to access the data and perform queries from a web browser such as Internet Explorer. No other software is needed to be loaded on the user's machine.
2. **Searching and Viewing Data –** Being able to type a word or a few key words used to search the data. This is analogous to Google with a single text entry for applying search criteria.
3. **Query Expressions –** Being able to create query expressions by selecting columns in the data and applying logical operators against sample values without having to know scripting languages of SQL or SAS.
4. **Compounded Queries –** Multiple queries can be compounded with an "AND" operator requiring each individual query to be true before the results are return. In addition, an "OR" operator can be applied between multiple queries which only require one query to be true in order for the results to be returned.
5. **Derived Variables** - A new derived variable can be defined based upon an existing column in the original source data. This new derived variable can then be selected in building new query expressions.
6. **Joining Data –** Source data can be joined or merged by key fields prior to having a query applied. The join can have options for inner, outer, left and right joins.
7. **Frequency and Means –** Simple statistics such as frequency counts can be applied for categorical variables or aggregate statistics such as mean or median can be applied to continuous numeric variables. This provides a quick way of identifying discrepancies between datasets or outliers in the data.

In addition to the list of functional requirements requested by the user, administration tasks to control user access and user account information are also needed to be easily managed through the user friendly web interface. Other solutions were considered such as employing Enterprise Guide or the DataFlux tools from SAS. These tools require some SAS skills and a multi-tiered computer architecture system including SAS BI Server implementation that needed a dedicated administrator to manage the software. Alternatively, this project required a light weight software approach simplifying the IT resource requirements, while still delivering a user friendly thin client solution to end users.

SOFTWARE VALIDATION STEPS

Due to the nature of the clinical data which this software will be used to view; formal validation and related documentation are needed. The following process was applied with formal documentation that accompanied each step. This process can viewed as administrative in nature but in practice, it does ensure the integrity of the development and implementation of the system within a validated environment.

STEP 1 – Change Request – A formal change control request is documented to identify what is being changed and what impact it has on other systems and processes. This can be a simple document but a four page template was used in this case. The following is a partial example of the form.

<table>
<tr><th colspan="3">Initiation of Project Change Request</th></tr>
<tr><td colspan="3">Change Title Syview Software Installation</td></tr>
<tr><td colspan="2">Date Request Submitted (dd/mm/yy)
03/12/2010</td><td>Signature</td></tr>
<tr><td>Change Priority
☐ Urgent
☒ Medium
☐ Low</td><td colspan="2">This installation of Syview will enable data managers to query SAS data more effectively without requiring users to make requests to SAS programmers or statisticians. This will enhance capabilities of data managers to perform tasks such as simple edit checks or data inquiries using automated software rather than having a programmer develop reporting scripts.</td></tr>
<tr><td>Request Type
☐ Requirement
☒ Enhancement
☐ Other</td><td colspan="2">This will enhance the productivity of data managers and freeing up time and resources for programmers and statistician to be more efficient with other analysis needs.</td></tr>
<tr><td colspan="3">Description of Problem or Change</td></tr>
</table>

Description of Problem or change requested: The challenge is currently, the analysis and reporting is done in SAS and therefore the data provided to the programmers are in SAS data format. However, data managers who prepare the data, do not necessarily use SAS and thus not familiar with using SAS tools to query and review the data. This change will enable data mangers to perform queries upon SAS data without the need for SAS programming knowledge since it provides a user interface to perform the necessary queries and joins.
Analysis of Alternatives (Name and describe potential alternatives and explain, why they have been rejected): The alternative is to have queries developed by SAS programmers or statisticians which require significant amount of time generating scripts that is used for these reports upon each request.

This form is a useful tool for communicating to both users and regulatory members of the team who would review to identify any potential issues from a global perspective prior to full implementation.

STEP 2 – Installation Qualification (IQ) – The test script instructions are formally documented in MSWord, but the accompanying test script is a SAS program. This script accesses all the files of the software system stored in a benchmark subfolder. It will then loop through each file and perform a binary comparison using a CHKSUM facility between installed files compared with the benchmark files. It then identifies any files that may be different. The code segment that performs this is shown here:

```
*** Create the command to perform CHKSUM ***;
command =  ' "' || md5tool || '" "benchmarks' || curslash  || curfile || '" "' ||
inpath || curslash || curfile
                  || '" > "' || repfile || '" ';
x '&command';

*** Create the Dataset Documenting Differences ***;
data diff;
   set xdiff;
   attrib note label="File Compared" length=$200;
   attrib comment label="Results from Comparison using CHKSUM" length=$200;
   note = "&curfile";
   comment = "The file is not identical compared to benchmark file.";
   output;
run;

*** Generate a Report of the Differences ***;
title2 height=&title_height "Summary Results from Comparing each file against
benchmarks";
proc freq data = sum (keep=comment);
run;
```

The resulting report from the IQ program is a PDF file which is produced using SAS ODS. The report contains a detail listing clarifying each file that is different along with a summary generated from the PROC FREQ. The following example report segment shows the summary portion.

Syview Installation Qualification (IQ) Results
Summary Results from Comparing each file against benchmarks

Results from Comparison using CHKSUM				
comment	Frequency	Percent	Cumulative Frequency	Cumulative Percent
The file exist in benchmark but can not be found.	1	0.04	2453	100.00
The file is OK.	2367	96.49	2367	96.49
The file is not identical compared to benchmark file.	1	0.04	2453	100.00

Some files were identified to be different due to server name and configuration differences. The binary comparison help verify that the installation process did not alter any of the files to ensure proper installation.

STEP 3 – Operational Qualification (OQ) – The operational qualification protocol is written in MSWord walking the tester through common operational steps, showing how to test each functional areas of the software. The instructions were developed with formal test script format which is tested with formal signatures.

5.4 Syview OQ - Query Data

Step	Instruction	Expected Result	Does Actual Result Match Expected Result?	Result
1	1. Open a web browser and navigate to the location of the Syview installation. An example is: http://sasserver 2. Enter the following value for the user name - password such as "admin - admin". 3. Click on the "Login" button. 4. Verify that the main screen "SAS Data Management" is displayed.	Verified that the main screen "SAS Data Management" is displayed.	☐ Yes ☐ No	☐ Pass ☐ Fail ☐ Deviation Initial/Date
2	1. Select on CM dataset from the hierarchy and then click on the View button on the right panel. 2. Verify that the "View Data" screen is opened, note the total observations this dataset has.	Verified that the "View Data" screen is opened. Noted the total observations this dataset has.	☐ Yes ☐ No	☐ Pass ☐ Fail ☐ Deviation Initial/Date
3	1. Click on the Back button on the bottom. 2. Navigate to the menu Tools > Query. 3. Verify that the Query SAS Dataset screen is displayed.	Verified that the Query SAS Dataset screen is displayed.	☐ Yes ☐ No	☐ Pass ☐ Fail ☐ Deviation Initial/Date

The test script is written to function with sample data that comes with the installation. This allows the testing to verify specific functionality with the ability to verify expected results. This can be time consuming so the project manger needs to allocate resources and sufficient time to ensure for complete and comprehensive testing, ensuring the operations of all key functional area.

STEP 4 – Performance Qualification (PQ) – The performance qualification evaluates how the system performs on large datasets with multiple simultaneous users. It simulates a stress test of the system including three users accessing a large set of data. A portion of the PQ test script is shown here:

3 Syview 4.0 PQ – Performance Qualification

The PQ testing will be performed on a dataset that is considered one of the largest data that users will encounter. It will also simulate three simultaneous users to push the system to see how well it performs. When processing large files between multiple users, it should not take more than two minutes for each request. There will be three users with the first one being the administrator. The second and third users of this test will be assigned at the time of testing and is referred to in this document as "user 2" and "user 3".

It is recommended that the script is written with general descriptions and user roles since the actual testing may require changes and specific user assignments. This test is based upon the OQ so some instructions can be reused. The main differences is that the PQ test scripts are applied with multiple users on large data.

Validation of software is a standard process that can be applied in a similar way for different implementations. What makes this validation project unique is that there are more graphical user interface elements compared to the traditional batch processing of SAS. Thus, this process requires more testing with interactive user interactions via a web browser. The testing included detailed instructions for the navigation and selection of graphical elements that are more visual compared to a pure scripting test script.

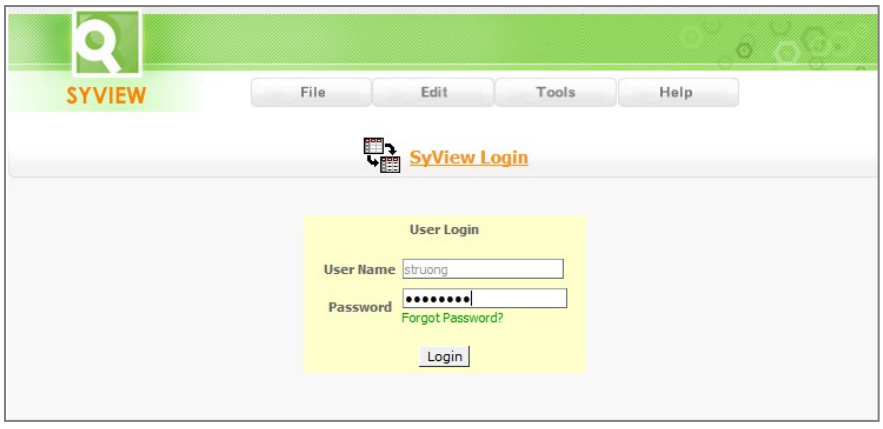
SOFTWARE IMPLEMENTATION

Syview can be implemented for data managers so that they can query SAS datasets without having to write SAS scripts. Access to the SAS datasets can be centrally managed by an administrator. Therefore, Syview provides a managed approach to SAS data access, while also extending to data managers a way of easily querying SAS datasets. Moreover, data managers cannot use Syview to write-over existing SAS datasets. They can only query SAS datasets and are not allowed to edit or update the same data. This allows for Syview to comply with 21 CFR Part 11 regulations from a change control perspective.

In this particular implementation, the software was deployed on a Windows 2003 server running SAS 9.2. The core technology is SAS but since this is delivered to a thin client, a CGI program was used and that was developed in C. The remaining logic on the server was SAS code used to generate HTML, which will described in the following steps.

STEP 1 – User Access

For secured data access, the first screen that is presented is a user login screen.



The SAS program on the server is invoked to generate the HTML code for the login screen. This HTML code contains only a couple of user input fields.

```
<table border="0" width="230" cellspacing="4">
  <tr>
    <td width="77" align="right"><b>User Name</b>
    </td>
    <td width="150"><input type="text" tabindex="1" name="Txtusername" size="23"
    onkeydown="return Login(event)"></td>
```

```
  </tr>
  <tr>
    <td width="77" align="right"><b>Password </b></td>
    <td width="150"><input type="password" tabindex="2" name="Txtpassword" size="20"
    onkeydown="return Login(event)"><br>
    <a href="#forgot" onclick="ForgotPassword();">Forgot Password?</a>
    </td>
  </tr>
</table>
```

The HTML also contains calls to a couple of JavaScript functions to perform error checking. This method of checking is more responsive for the user since the software does not need to connect to the server to validate missing credentials. The JavaScript is inserted along with the HTML so it is delivered to the browser at the same time.

```
function CheckLogin()
{
      var str = waitingBox("Verify Login....",'Verify Login',"");
      // Perform error checking
      var usrname=document.myForm.Txtusername.value;
      var password=document.myForm.Txtpassword.value;
      var userlibrary="sviewusr";
      // Invalid extension
      if (usrname == "")
      {
         popupDlg(null, "", "<b>WARNING</b> - User Name is missing.",
         "",null,"WARNING","");
         return;
      }
      if (password == "")
      {
         popupDlg(null, "", "<b>WARNING</b> - The required password is missing.", "",
         null, "WARNING","");
         return;
      }
...
```

The use of HTML combined with JavaScript and AJAX is a powerful combination for a rich user experience through the web interfaces. This is dynamically generated with a SAS program on the server. This can also be generated with the use of traditional DATA _NULL_. Alternatively, we find it more efficient to use a SUBMIT call with in SAS SCL to generate the HTML code.

```
   *** capture the current libname and catalog name ***;
   curcat = scan(screenname(), 1, '.') || "." || scan(screenname(), 2, '.');
   broker = "syview.exe";
   imgpath = "../syview/images";
   jspath="../syview/js";

   length root $200;
   call display('_getrootlib.scl',root);

   submit;
<script type="text/javascript" src="&jspath/waiting_dialog.js"></script>
<script type="text/javascript" src="&jspath/info_dialog.js"></script>
<script type="text/javascript" src="&jspath/forgot_password.js"></script>

   <script type="text/javascript">
```

```
      function init1()
      {
            ClassWaitingDialog.initWaitingDialog();
            ClassInfoDialog.initInfoDialog();
            ClassLoginForgotDialog.initDialog();
      }

      dojo.addOnLoad(init1);
   </script>
...
   endsubmit;
```

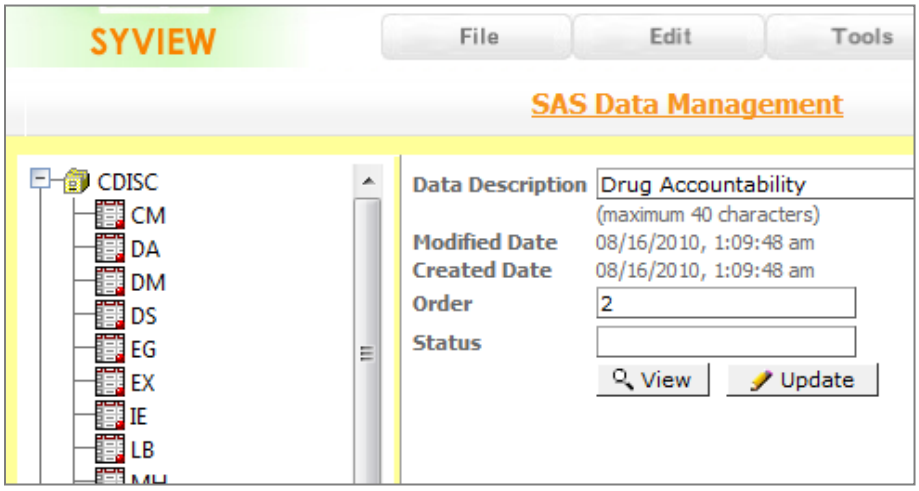
This method efficiently generates code with variables in the program that is dynamically inserted into the HTML and JavaScript with an ampersand. This is similar to how a SAS macro variables are used but with greater legibility. In this example, the path is dynamic and is set to the variable JSPATH storing the path value.

```
<script type="text/javascript" src="&jspath/waiting_dialog.js"></script>
```

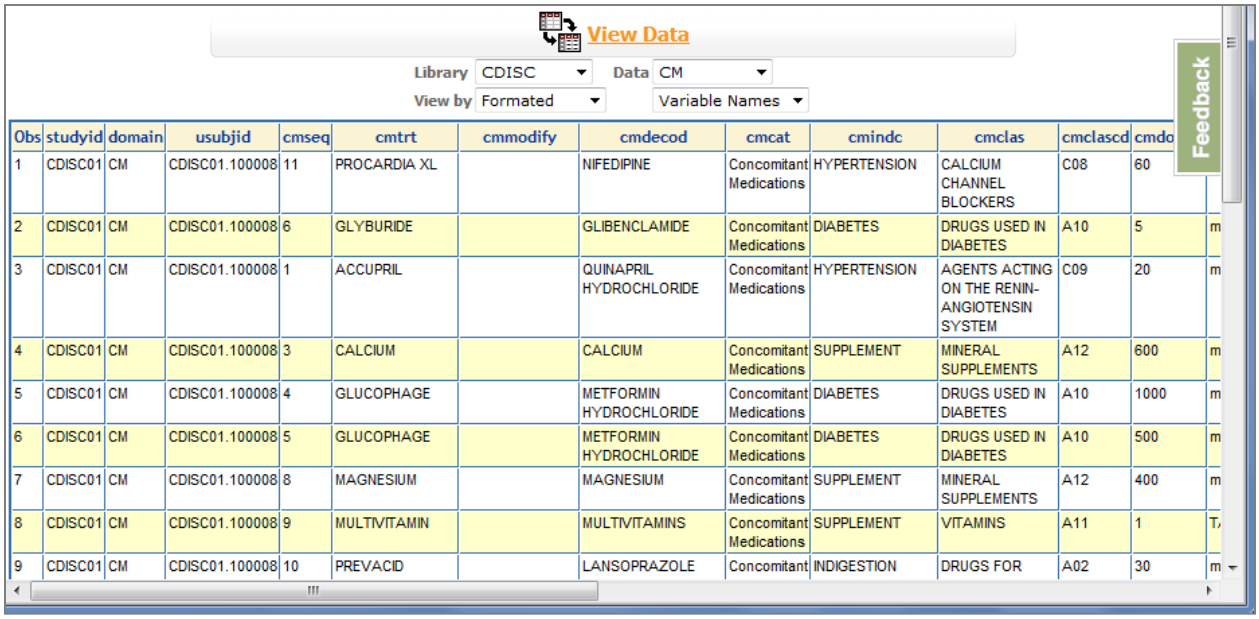
The definition is applied during installation and is dynamically resolved during the call from the SAS program. The use of DATA _NULL_ and SAS Macro will accomplish the same task. However, this alternative SCL approach is easier to read for ease of development and software maintenance.

STEP 2 – Viewing SAS Datasets

After the user has authenticated with a valid user name and password, they can navigate to the dataset if the user has permissions. This is controlled on the server through the use of an ACL or access control list. This implementation is stored in a SAS dataset and can easily be managed by a SAS program. The navigation to the dataset is displayed in a folder system similar to Windows Explorer.



Each folder displayed corresponds to a SAS library. Upon selection, the user would click on the “View” button on the detail panel to view the dataset.



View Data

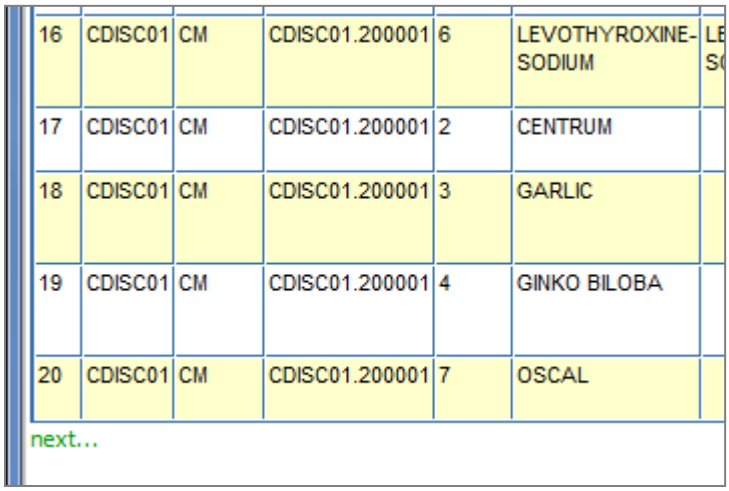
Library CDISC Data CM

View by Formated Variable Names

Feedback

Obs	studyid	domain	usubjid	cmseq	cmtrt	cmmodify	cmdecod	cmcat	cmindc	cmclas	cmclascd	cmdo	
1	CDISC01	CM	CDISC01.100008	11	PROCARDIA XL		NIFEDIPINE	Concomitant Medications	HYPERTENSION	CALCIUM CHANNEL BLOCKERS	C08	60	
2	CDISC01	CM	CDISC01.100008	6	GLYBURIDE		GLIBENCLAMIDE	Concomitant Medications	DIABETES	DRUGS USED IN DIABETES	A10	5	m
3	CDISC01	CM	CDISC01.100008	1	ACCUPRIL		QUINAPRIL HYDROCHLORIDE	Concomitant Medications	HYPERTENSION	AGENTS ACTING ON THE RENIN-ANGIOTENSIN SYSTEM	C09	20	m
4	CDISC01	CM	CDISC01.100008	3	CALCIUM		CALCIUM	Concomitant Medications	SUPPLEMENT	MINERAL SUPPLEMENTS	A12	600	m
5	CDISC01	CM	CDISC01.100008	4	GLUCOPHAGE		METFORMIN HYDROCHLORIDE	Concomitant Medications	DIABETES	DRUGS USED IN DIABETES	A10	1000	m
6	CDISC01	CM	CDISC01.100008	5	GLUCOPHAGE		METFORMIN HYDROCHLORIDE	Concomitant Medications	DIABETES	DRUGS USED IN DIABETES	A10	500	m
7	CDISC01	CM	CDISC01.100008	8	MAGNESIUM		MAGNESIUM	Concomitant Medications	SUPPLEMENT	MINERAL SUPPLEMENTS	A12	400	m
8	CDISC01	CM	CDISC01.100008	9	MULTIVITAMIN		MULTIVITAMINS	Concomitant Medications	SUPPLEMENT	VITAMINS	A11	1	T
9	CDISC01	CM	CDISC01.100008	10	PREVACID		LANSOPRAZOLE	Concomitant	INDIGESTION	DRUGS FOR	A02	30	m

The initial view of the data is generated using an ODS PROC REPORT that is similar to a standard SAS program. The difference is when this is viewed on a browser; the data is broken down to smaller chunks rather than being displaying in its entirety. Depending on the network connection and bandwidth, displaying the entire dataset at once can impede on the systems performance and responsiveness. By breaking down the data into smaller chunks, this can be quickly displayed for user review. In this example, 20 rows are shown at once.



16	CDISC01	CM	CDISC01.200001	6	LEVOTHYROXINE-SODIUM	LE SO
17	CDISC01	CM	CDISC01.200001	2	CENTRUM	
18	CDISC01	CM	CDISC01.200001	3	GARLIC	
19	CDISC01	CM	CDISC01.200001	4	GINKO BILOBA	
20	CDISC01	CM	CDISC01.200001	7	OSCAL	

next...

The user can click on the “next…” hyperlink at the bottom of each screen to navigate to the next block of data. This is dynamically generated for each dataset with the following HTML code:

```
</table>
</center></div>
<!-- DATAVIEW: skip1 //-->
<div id='nextbar'>
<table>
  <tr>
    <td colspan="&span">
    <a href="javascript:datchange('',&cnt);">
    next...</a></td>
  </tr>
</table>
```

The HTML table includes a call to a JavaScript function DATCHANGE, which in turn calls the following SAS program logic on the server.

```
*** Capture the data with the correct block starting with CURCOUNT ***;
cnt = &CURCOUNT;

do while(fetchobs(dsid,cnt)=0);
   curdat = getvarc(dsid,varnum(dsid,'datname'));
   rc = insertc(datlst,curdat,-1);
   if lowcase(datname) = lowcase(curdat) then do;
      cursrc = getvarc(dsid,varnum(dsid,'source'));
      sourcepath = getvarc(dsid,varnum(dsid,'path1'));
      if sourcepath = '' then sourcepath = getvarc(dsid,varnum(dsid,'path2'));
   end;
   cnt = cnt + 1;
end;
```

Each time the SAS code is processed, it captures the current data block with the CURCOUNT variable which tracks the user's current record count being viewed. When the user clicks on the "next…" link, the current count variable is then incremented to the next block of data.

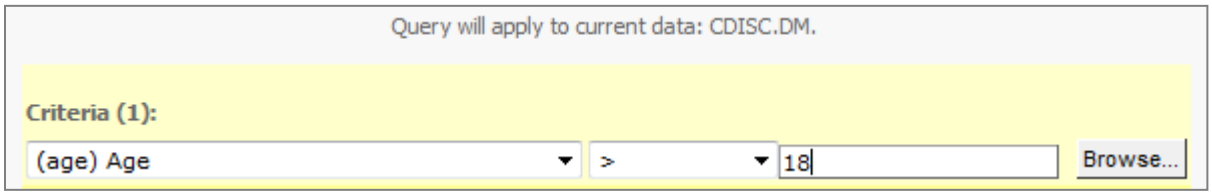
In this example, three different distinct languages were used starting with HTML displaying data on the browser directly to the user. The JavaScript within the HTML then manages what the current block of data the user is currently viewing. This code then calls the server SAS code which captures the correct block of data to then generate the report using ODS. The output of the SAS program resulting in HTML looping back to the browser. This loop occurs under one second so the user does not see it as an arduous process, but rather a transparent and seamless view of the SAS dataset.

STEP 3 – Creating a Query

A query starts out with the user wanting to ask a question about the values with the dataset of interest. The question is expressed in a form of a data constraint to filter the entire dataset down into a smaller set containing the data values that the user would like to view. An example data constraint is used to show all subjects in a demographic dataset who are older than 18 years of age. If this were to be written in a traditional SAS statement, it would look like:

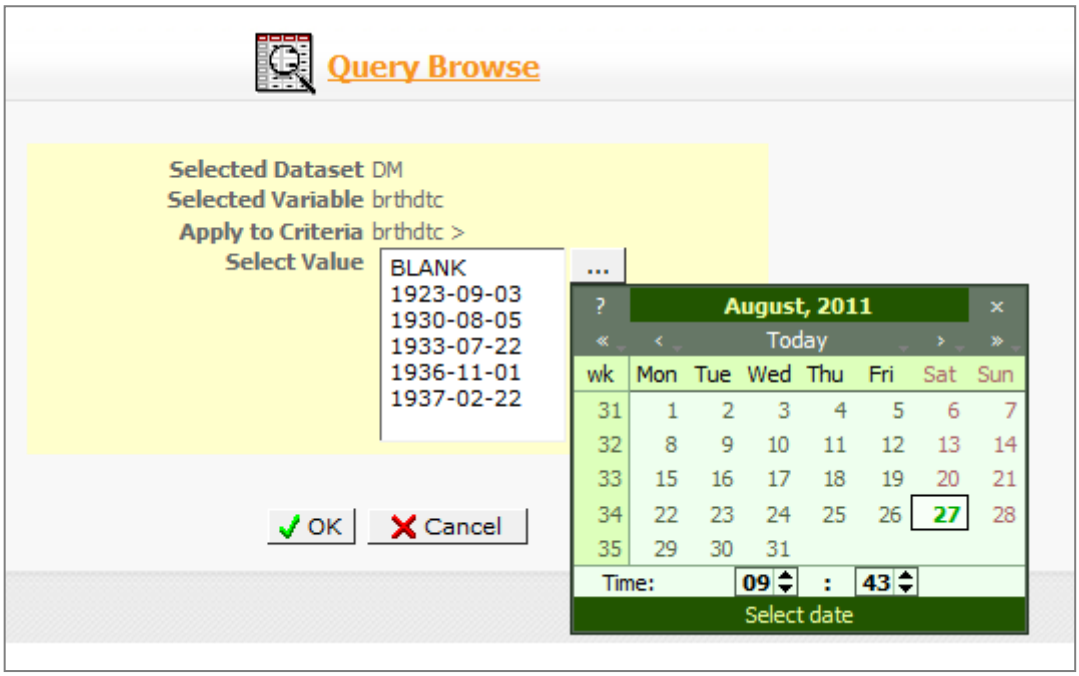
```
where age > 18;
```

This same expression can be selected using the interface as shown here:



The advantage of building the query expression with an interface is that all the variables are presented in the pull down menu and the type of operators such as ">" greater than sign is also prompted. In this example, you would type the value of "18" for the expected value. However, there are many cases where you need to select a real value which exist inside the dataset. The "Browse…" button can be used to display all the distinct values for you to select. This is normally not readily available within a scripting environment, unless you write additional code with PROC PRINT.

The value of 18 above is an example for a numeric value. You can also specify other types of variables such as a character or a date value. The numeric and character variable format are open and do not have strict formats applied. If the value is a date variable, there are many SAS date formats that can be applied. Here is an example of specifying a birth date.



In this example, the user can select an existing date in the format of yyyy-mm-dd. In addition, the "…" button is used to display a popup calendar allowing the selection of a specific date. This is much easier to specify since you do not have to be concern about the format such as typing the year before the months with separating dashes.

There are several ways of creating this "Query Browse" selection screen. In this example, the program reads the dataset DM and stores the distinct values into an SCL list. We find that this is even faster than using a SAS dataset since the list is stored in memory which can be manipulated and processed with lightening speed.

```
*** Capture the dataset to populate the selection list ***;
do while(fetchobs(dsid,cnt) = 0);
   *** Handle characters ***;
   if vartype = "C" then do;
      curvalc = getvarc(dsid,varnum(dsid,variable));
      if curvalc ne '' then rc = insertc(vallst,curvalc,-1);
   end;

   *** Handle Numeric ***;
   if vartype = "N" then do;
      fmt = '';
      fmt = varfmt(dsid,varnum(dsid,variable));

      curvaln = getvarn(dsid,varnum(dsid,variable));
     if curvaln ne '' then do;
         if  index(fmt,'DATE') > 0 or index(fmt,'TIME')>0 then do;
             datefmt  = fmt;
             inputflaq = "1";
             curvaln2 = putn(curvaln,fmt);
             rc = insertc(valdatlst,curvaln2,-1);

         end;
         rc = insertc(vallst,compress(curvaln),-1);
      end;
   end;
   cnt = cnt + 1;
end;
```

```
rc = sortlist(vallst,'NODUP');
```

The last step sorts the list with the option of NODUP. This is similar to the PROC SORT option or the “select distinct” option within PROC SQL. For a list of values within a menu, we find that the use of this SAS SCL list is more efficient than a dataset using either PROC SORT or PROC SQL.

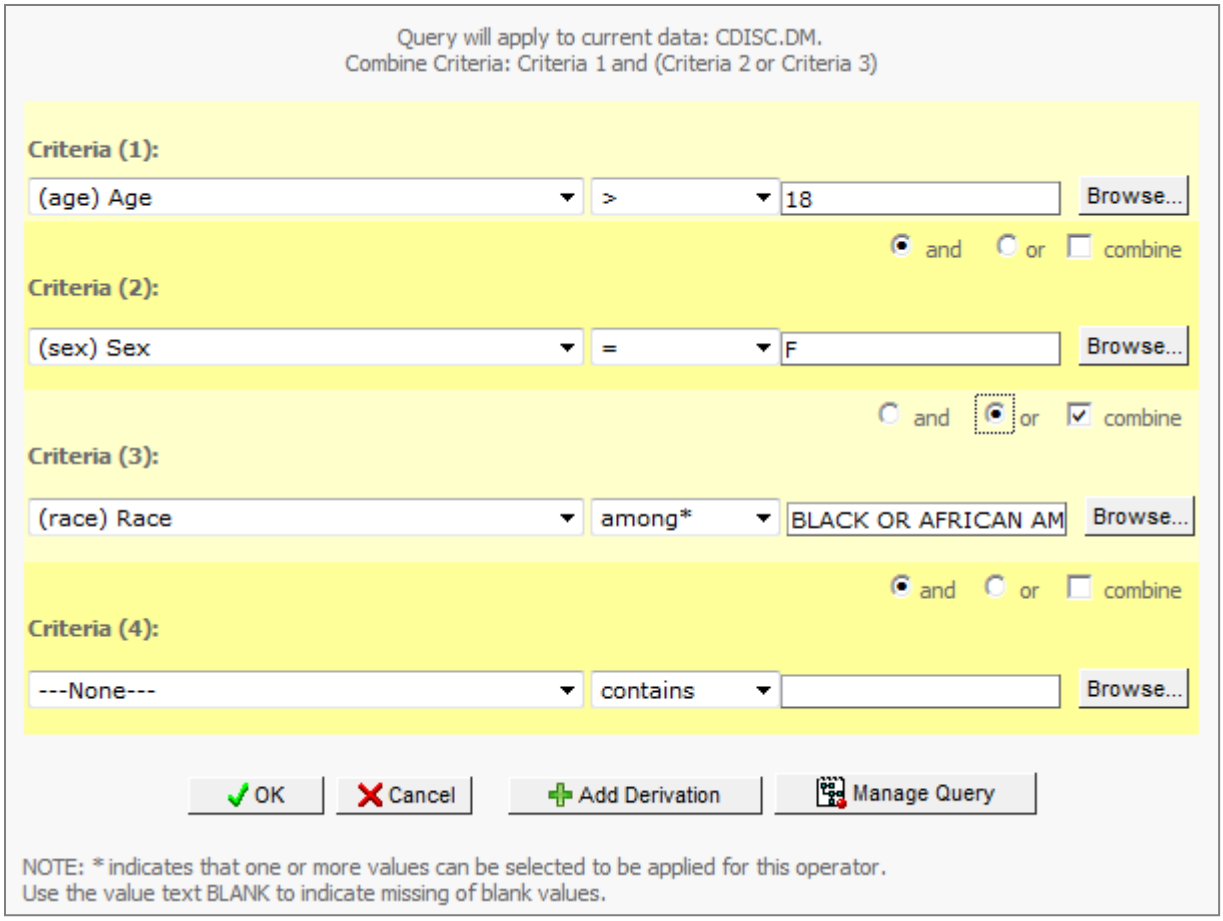
The use of a popup calendar is accomplished using JavaScript and AJAX. There are many libraries that exist, so rather than re-inventing the wheel, we included these into the current HMTL.

```
<script type="text/javascript" src="../syview/cal/js/zapatec.js"></script>
<!-- Custom includes -->
<!-- import the calendar script -->
<script type="text/javascript" src="../syview/cal/js/calendar.js"></script>
<!-- import the language module -->
<script type="text/javascript" src="../syview/cal/js/calendar-en.js"></script>
```

Different scripts function slightly differently on different web browser. An example calendar designed by Zapatec functions one way on Internet Explorer but may be different on Firefox. Due to these variations, we have selected the best calendar routine for each browser depending on what the user is currently using. Some of the software has to be licensed but many of these JavaScript and AJAX code is open source, so it can be used more freely. We find it worth the effort researching and finding a good routine that exist rather than creating one for general purpose tools such as a calendar selection in this example.

STEP 4 – Storing and Running Queries

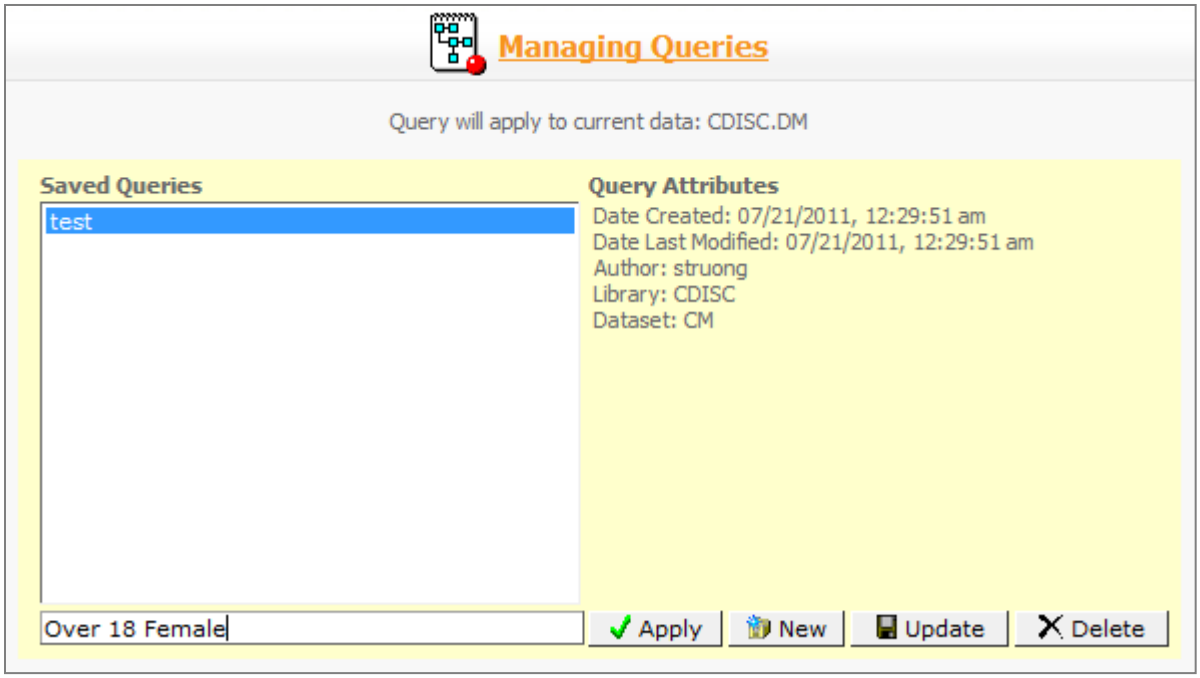
The above example is a simple single expression query. In reality, queries are more complex with compound expressions applied with AND/OR logic. This example shows a continuation of the above query with more conditions.



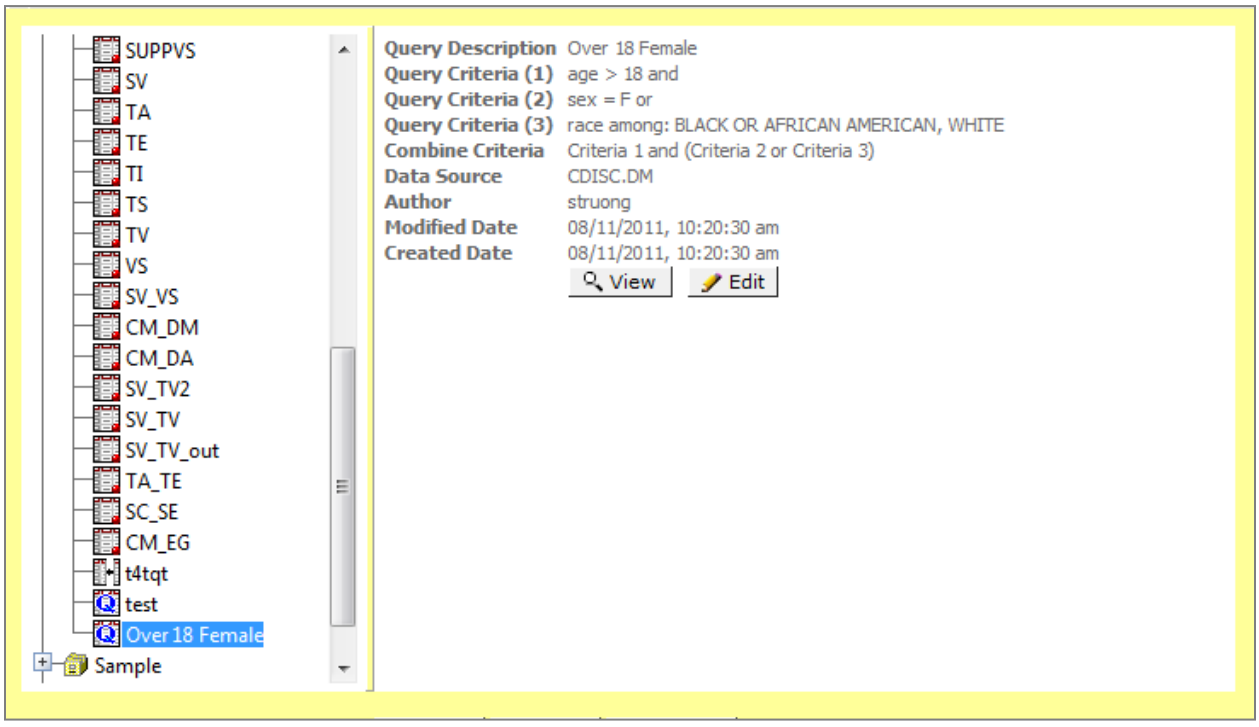
By selecting the AND/OR radio button along with the “combine” check boxes, the query is more complex as shown in the compound expression:

Combine Criteria: Criteria 1 and (Criteria 2 or Criteria 3)

It is suggested that complex queries be saved since it is useful to apply the same query again once the data has been refreshed. This is accomplished through the “Manage Query” button which brings the user to the “Managing Query” screen. This allows an English name to be associated with the query such as “Over 18 Female”.



This name can be recalled and reference more easily compared to a series of logical data constraint expressions. Once saved, it will be available through the explorer view along with all the other datasets in this library.



The metadata for this query is managed within a SAS dataset. This is an efficient way of storing metadata since updates can be made easily while PROC PRINT and PROC REPORT make it easy for reporting. The dataset that stores this looks like:

queryid	name	pos	dataset	querylib	desc	author	moddat	crtdat	usrname
8	test	8	CM	CDISC	test	struong	21JUL11:00:29	21JUL11:00:29	struong
9	Over 18 Female	9	DM	CDISC	Over 18 Female	struong	11AUG11:10:20	11AUG11:10:20	struong

This is a similar structure that is used to managed datasets and other objects within the system. The SAS dataset attributes for this is shown here:

Variable	Type	Len	Format	Label
queryid	Num	8	12	Query ID
name	Char	40	$40	Query Name
pos	Num	8	12	Position
dataset	Char	200	$200	Dataset
querylib	Char	40	$40	Library
desc	Char	200	$200	Query Description
author	Char	200	$200	Author
moddat	Num	8	DATETIME13.	Date Time Modified
crtdat	Num	8	DATETIME13.	Date Time Created
usrname	Char	20	$20	User Name

The specific variable names used to store this structure is not significant. What is important is that the metadata is captured in a consistent way so that similar programming logic can be re-used to mange queries along with other objects in the system, such as dataset permissions or the joined datasets.

Various technologies are used to manage the queries including the HTML, JavaScript and AJAX on the front end, while SAS remain the core technology used on the server side. Each component requires its own syntax and different approaches. While the front end elements contributes to what users see when they interact with the system through the web browser; the SAS server component has the most impact on the speed and efficiency of data access.

CONCLUSION

Querying data can be a complicated task, especially when different data sources must be combined. Syview is a product which can simplify this task by providing data managers with a user-friendly interface. Thus Syview can meet the needs of data managers for data query, yet maintain regulatory standards for accessing clinical data. The success of a system such as Syview is similar to other systems where users access information through a web interface such as Google. There are many different types of sophisticated technologies that are applied both on the front end web browser in conjunction with components on the server. In this implementation, SAS is the main technology on the server, but it is interwoven with dynamic HTML5 related technologies on the web browser side. What makes this effective and successful from a user's perspective is that all the complexity is invisible to the user since all that is needed to perform a query on a SAS dataset is just to type the query option on a web browser. The minimal approach of Google is what we also strive towards in the design of Syview.

CONTACT INFORMATION

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

Sy Truong
MetaXceed, Inc. (MXI)
42978 Osgood Rd
Fremont, CA 94539-5627
tel: 510.979.9333
fax: 510.440.8301
E-mail: sy.truong@meta-x.com
Web: www.meta-x.com

Carey Smoak
Roche Molecular Systems, Inc.
4300 Hacienda Drive
Pleasanton, CA 94588
Tel: (925) 730-8033
Fax: (925) 225-0207
E-mail: carey.smoak@roche.com