Building a Web Based EIS for Data Analysis

Ed Confer, KGC Programming Solutions, Potomac Falls, VA

Abstract

Web based reporting has enhanced the ability of management to interface with data in a point and click architecture. In order to build an effective front-end, a structural framework design must be built on flexibility and efficiency. The data to be extracted should be stored as a collection table, to ensure that timely and accurate data can be extracted from a single source. The program should be written so that any number of parameters can be filtered in the extraction. The Web and SAS Software can deliver this reporting capability to management. Utilizing SAS Software as the data extraction tool, the user can select any number of data conditions based on reporting requirements. The data can be presented in multiple methods, such as a graph depicting data points over time, or a tabular report based on summary data. The reporting tool can be designed to provide to the user a variety of reporting capabilities and data filters. Although the focus of this paper is the SAS Software code used in developing this reporting tool, other components of the development of a Web based query tool such as Java Script and HTML will also be discussed.

Introduction

Today there are many ways to extract and analyze data contained in a data warehouse. The successful businesses realize the need to leverage the technical expertise of the IT department with the data needs of the business area to construct a robust decision making process.

This paper presents to the SAS programmer a methodology for constructing a reporting system that is both flexible and efficient. Through a series of examples I will illustrate how to: 1) extract data from data warehouse using PROC SQL; 2) design a filter frame with JavaScript and HTML; 3) create PROC SQL macros to create "where=" clauses from URL name resolutions; 4) output SAS/GRAPH to the Web.

Background

ABC Junkyard Cars Decision Management is involved with loss minimization in the leasing of its best wrecks. A study completed at the end of 2000 fiscal year showed that a certain brand of auto paid off much sooner relative to the population of all autos leased. Decision Management determined that the best way to test the validity of a predictive model of this adverse behavior was to monitor timely data. However, it also surmised that other forces would possibly contribute to the higher attrition rates, and therefore other trending data would be useful. In the past, the DM staff would have asked DSS (Decision Support) to produce a report on an ad-hoc basis, but now with Web reporting, reports can be viewed with a click of the mouse. The DM team made a choice; it opted for a Web-based EIS.

The first step taken by the team was to clarify the business needs and to communicate to the IT department the business rules. Once the programmers and the business unit reached an agreement on the specifications and actions needed, the work of developing a Collection Table based on the business rules could be initiated. To accomplish this, the IT department selected SAS PROC SQL.

Using SAS PROC SQL to Extract Data From the Data Warehouse

The Data Warehouse that contains ABC Junkyard data resides in a number of Oracle tables. To satisfy the business needs of the EIS, data needs to be extracted from multiple sources, and several tables will need to be joined together to create the data for the collection table. Additionally, based on the business rules, a number of attributes will require transformation to a new target variable. First, select the data based on those cases where a lease was terminated within a certain time period. This can be accomplished using SAS PROC SQL. Example 1 is an illustration of data extraction utilizing SQL The example below is an illustration of SAS code that reads in the data contained in temporary Data Set named TERMS created at the top of the example. The TERMS data contains ACCOUNT, the year the lease terminated, the value of the auto at termination and the balance of the lease agreement.

Example 1.

```
%let begin=2000;  /* Set Parameters */
%let end  =2003;

data terms;
infile cards;
input account $ termyear value bal flag $ rate;
cards;
123 2001 1300 4000 N 10
456 2001 2000 1090 N 5
567 2002 5000 1023 N 11
888 2003 2090 6700 N 12
199 2002 1300 1200 N 13
444 2002 3500 1700 N 6
177 2001 2700 1050 N 7
222 2003 5000 2310 N 9
333 2002 4002 1930 N 11
101 2001 5000 3000 N 12
202 2003 1340 1500 N 13
112 2001 2000 4000 N 6
116 2003 2333 3301 N 5
587 2002 1415 1110 N 4
988 2002 1011 1270 N 4
184 2002 1550 2200 N 6
```

```
445 2002 3510 1160 N 7
561 2003 1119 1100 N 7
232 2001 5505 2220 N 4
343 2002 9551 1910 N 4
161 2001 5021 5000 N 6
239 2003 4610 6021 N 7;
run;

proc sql;
create table terms as
 select * from (
 select distinct
    account       as account
   ,termyear      as termyear
   ,value          as value
   ,bal             as balance
   ,rate            as rate
 from
 terms
 where  termyear
 between &begin and &end
 and    flag    <> 'Y');
quit;
```

The code above in Example 1 will create a SAS Data Set in the Work Directory named TERMS. This represents to ABC the leasing agreements that were terminated between 2000 and 2003. This will be the BASE population. Please note that for simplification purposes, the PROC SQL code above illustrates data extraction from a SAS Data Set; it does not represent the Pass-thru method for connecting to Oracle.

The next step in the program is to extract the TRANSACTION record for all Terminations, which describes the type of termination of the lease agreement, [B Bankruptcy or P for Payoff]. For Payoffs, the TRANSTYPE attribute is broken out by CA, cash, MO, Money Order and CK for check. The Bankruptcy codes contain NA for Not Applicable. In Example 2 below the TRANS table is combined with the TERM table, and with the where clause, only those transactions with a type "P" will be extracted.

Example 2.

```
data trans;
infile cards;
input account $ type $ transtype $ flag $;
cards;
123 P CA N
456 P CA N
567 P MO N
888 P CK N
199 P CA N
444 P MO N
177 B NA N
222 B NA N
333 B NA N
101 B NA N
202 P CA N
112 P MO N
116 P CK N
587 P CA N
988 P CA N
184 P CK N
445 B NA N
561 B NA N
232 P CK N
343 P CK N
161 P MO N
239 B NA N
;
run;

proc sql;
create table newtrans as
 select * from (
 select distinct
    a.account          as account
   ,b.transtype        as transtype
   ,a.termyear         as termyear
   ,a.value            as value
   ,a.balance          as balance
   ,a.rate             as rate
 from
 terms a inner join trans b
 on
 a.account = b.account
 where
 b.type = 'P' and b.flag <> 'Y');
quit;
```

The desired data is extracted from two tables and is joined to form part of the collection table. The last bit of data needed to create useful data analytics is reference data, which is information regarding each account. The table in Example 3 below contains the year, make and color of auto. This data extract is then combined with the data produced in Example 2 above. At this point the collection table is ready for data analytics.

Example 3.

```
data ref;
infile cards;
input account $ autoyear make $ color $;
cards;
123 1960 Ford Red
456 2000 Buick Blue
567 1965 Toyota Black
888 1995 Chrysler Red
199 1965 Ford Black
444 2001 Buick White
177 1965 Ford Black
222 1995 Honda Brown
333 1980 Pontiac Orange
101 1986 Ford Purple
202 1954 Honda Black
112 1956 Ford Blue
116 1989 Ford White
587 1994 Toyota Silver
988 1996 BMW Red
184 1995 Ford Yellow
445 1996 Buick Maroon
561 1995 Toyota Grey
```

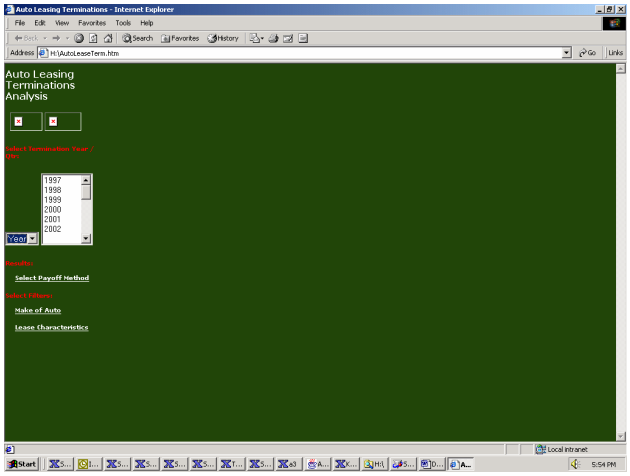
```
232 1999 Honda White
343 1998 Pontiac Blue
161 1994 Ford Black
239 2001 Honda Brown
;
run;

proc sql;
create table collection as
 select * from (
 select distinct
            a.account          as account
            ,a.transtype       as transtype
            ,a.termyear        as termyear
            ,a.value           as value
            ,a.balance         as balance
            ,a.rate            as rate
            ,b.autoyear        as autoyear
            ,b.make            as make
            ,b.color           as color
 from
 newtrans a inner join ref b
 on
 a.account = b.account);
```

Building the HTML Window and Filter Criteria

Now that the DM team has the data in place, it can decide on the means to filter out the data and to report on the analysis. The team has decided to combine the tools of HTML and JavaScript and to utilize the Web for its reporting.

Figure 1.



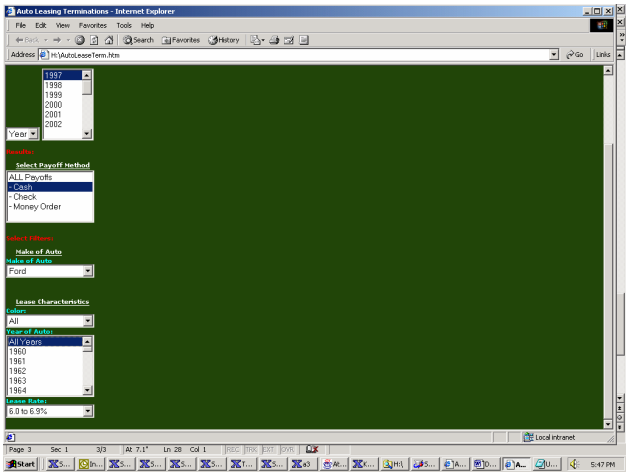
In Figure 1 above an Internet Browser displays the Filter page for the Auto Leasing Terminations Analysis. It is from this website that the DM team can select the criteria for its analysis. The frame is constructed by using HTML and JavaScript programming languages. The DM team can go to this website on the Internet Browser and by using the mouse can select the criteria to pull the data. This window above will be part of a larger frame that will encompass a main window that is much larger, where data output such as graphs and listings will go; and a header window that will provide titles and navigation buttons.

In this example the DM team would like to select records with the following criteria:

1. Autos Built in 1997,
2. Paid Off by CASH,
3. Make is Ford
4. Interest Rate between 6 and 6.9%.
5. Red and Black Colors

Figure 2 below shows the screen print of the way the Internet Browser would appear.

Figure 2.



By clicking on the selection criteria and thereby selecting the filtering criteria, the URL address will display and save the characteristics of the data selection. For instance, by selecting the value of 1997 for Lease Termination Date, the URL will display TERM=1997 in the Internet Browser window. Likewise, the URL will display "payment= CA" for Cash. This is the code behind the Internet Browser window:

Example 4.

```
name="payment" multiple >
            <Option value="0" selected>All Payoffs</Option>
            <Option value="CA">-  Cash            </Option>
            <Option value="CK">-  Check           </Option>
            <Option value="MO">- Money Order</Option>
</select></P>
```

By designating the option 'MULTIPLE' in Example 4 above JavaScript the URL will display multiple selections and the code that extracts the data from the collection table must be programmed to select each of the choices in the WHERE clause. This will be covered in the next section.

The user could also select RED for the color, and select 2000 as the Model Year. Additionally, if the Make of the auto was being analyzed, the user could select Ford, or Pontiac or any other listed in the scroll bar. The number of combinations of the data from the collection table is limited only by the business needs and the flexibility of the program that filters the data from the collection table.

Once the selection is finalized the user submits the code to the SAS Broker. SAS reads the variables on the URL and creates SAS Macro variables that are then used in the SAS PROC SQL extract from the collection table. For example, based on the criteria above, YEAR would resolve to "1997" (see Example 7), PAYMENT would resolve to "CA" (see Example 4), MAKE would resolve to "1" (see Example 5), RATE would resolve to "5" (see Example 8), and COLOR would resolve to "1" and "7" (see Example 6).

Example 5.

```
name="make" multiple>
          <Option Value=0 Selected>All</Option>
          <Option Value=1>Ford     </Option>
          <Option Value=2>Honda   </Option>
         .
         .
          <Option Value=9>BMW    </Option>
</select></P>
```

Example 6.

```
name="color" multiple>
          <Option value="0" Selected>All</Option>
          <Option value="1">RED       </Option>
          .
          .
          <Option value="7">BLACK   </Option>
          <Option value="8">GREY     </Option>
</select></P>
```

Example 7.

```
name="year" multiple>
          <Option value="0" Selected>All Years</Option>
          <Option value="1960">1960    </Option>
          <Option value="1961">1961    </Option>
         .
         .
          <Option value="2002">2002    </Option>
</select></P>
```

Example 8.

```
name="rate">
          <Option value="0" Selected>All</Option>
          <Option value="1">Under 3%      </Option>
          <Option value="2">3.0 to 3.9%    </Option>
          <Option value="3">4.0 to 4.9%    </Option>
          <Option value="4">5.0 to 5.9%    </Option>
          <Option value="5">6.0 to 6.9%     </Option>
         .
         .
          <Option value="13">Above 13.9%</Option>
</select></P>
```

In each of the examples above {examples 4 thru 8}, the default selection is set to all. This means that when the user opens the Web application each of these scroll bars will be highlighted with "All". Therefore in the SQL extract in the WHERE condition, all records will be selected. The next phase in the development of the EIS is to code the SAS program so that the MACRO values from the URL will be resolved correctly in the extract.

Designing an Automated SQL Where Clause Based on MACRO Values

One of the keys in developing a program to read the selection criteria from EIS into the WHERE clause in the PROC SQL data pull is to create an automated selection criteria from the macro variables generated on the URL. An obstacle to overcome in reading the URL is the occurrence of multiple selections within a filter category. SAS creates an additional value in the URL for those filter categories that contain MULTIPLE capabilities for selection. In order to recognize multiple selections and to designate the appropriate number of selections it is necessary to develop a SAS MACRO to accommodate all possible values.

Once the HTML submit button is selected SAS bundles all of the information from the JavaScript and sends the parameters as well as the SAS Program name to the Application Dispatcher. Based on the criteria 1-5 above, more than one COLOR variable value is selected. The Dispatcher then is going to send the following to the SAS application/program:

COLOR0	2
COLOR	1
COLOR1	1
COLOR2	7

The COLOR0 value designates the number of selections made in the COLOR scroll box. COLOR1 and COLOR2 values indicate the values for RED and BLACK respectively. The SAS program that reads this data will create SAS MACRO variables that resolve to these values. Example 9 below illustrates the code to accomplish this.

Example 9.

```
%macro color;

 %let comma  = %str();
 %local i;
 %global color color0;
 %if %superq(color) ne %str() %then %do;
  %if %superq(color0) eq %str() %then %let color0=1;
          %let color1 = %superq(color);
          %put Nbr Colors is &color0;
  %do i = 1 %to %superq(color0);
```

```
%let  xcolor=&xcolor&comma
%str(%'%superq(%qtrim(color&i))%');
      %let comma = %str(,);
            %end;
  %end;

%mend color;
```

The SAS MACRO variable XCOLOR will resolve to a character string:

"1","7"

Each of the parameters that allow for multiple selections will require the code indicated above in Example 9. Once each of the macros are called it will be possible to create the WHERE condition for the SAS PROC SQL data pull.

Example 10.

```
%if &color=0 %then %do;
    %let cond4   = %str();
%end;
%else %do;
   %let cond4   = %str(yr_orign in (%unquote(%str(&xcolor))));
%end;
```

In Example 10 above the MACRO creates two SAS MACRO variables, COND4 and TITLE4. The first "if" condition in the MACRO will assign a null value to MACRO variable COND4, which means that in the WHERE clause in the SQL query this condition will not appear. In other words, if all colors are selected in the EIS then there is no requirement to specify a color in the conditions statement. By default, the SQL query will select all records, regardless of the value contained in the attribute COLOR. Since the selection designated above is RED and BLACK the first "IF" condition is FALSE, and the next "IF" condition is executed. The MACRO variable COND4 will contain the values of "1" and "7", and will resolve to:

"1","7".

In order to complete the building of the conditional statement, each of the selection variables contained in the EIS will require code as in Example 10 above. Once all of the MACRO variables are created, it will be a matter of creating MACRO variables to insert into the WHERE statement; these are the "AND" 's that need to be placed between each condition.

First, set the number of EIS variables (less 1) for "AND" and "OUT" to null:

```
%let and1 = %str();
%let out1 = %str();
%let and2 = %str();
%let out2 = %str();
%let and3 = %str();
%let out3 = %str();
%let and4 = %str();
%let out4 = %str();
%let and5 = %str();
%let out5 = %str();
```

The reason to select number of variables less 1 is because this construction of "AND" will be placed between each condition, there will not be an "AND" necessary after the last possible condition.

Example 11 below is code to construct the "AND" statements and to complete the WHERE clause.

Example 11.

```
%do %while(%superq(out1) = %str());
  %if %superq(cond1) ne %str() %then %do;
    %if %superq(cond2) ne %str() %then %do;
        %let and1 = and;
                        %let out1 = 9;
            %end;
            %else
    %if %superq(cond3) ne %str() %then %do;
        %let and1 = and;
                                %let out1 = 9;
            %end;
            %else
    %if %superq(cond4) ne %str() %then %do;
        %let and1 = and;
                                %let out1 = 9;
            %end;
            %else
    %if %superq(cond5) ne %str() %then %do;
        %let and1 = and;
                                %let out1 = 9;
            %end;
            %else
    %if %superq(cond6) ne %str() %then %do;
        %let and1 = and;
                                %let out1 = 9;
            %end;
            %else
    %if %superq(cond7) ne %str() %then %do;
        %let and1 = and;
                                %let out1 = 9;
            %end;
            %else
    %if %superq(cond8) ne %str() %then %do;
        %let and1 = and;
                                %let out1 = 9;
            %end;
            %else
    %if %superq(cond9) ne %str() %then %do;
        %let and1 = and;
                                %let out1 = 9;
            %end;
            %else %do;
                %let and1 = %str();
                                %let out1 = 9;
    %end;
  %end;
  %else %do;
                %let and1 = %str();
                                %let out1 = 9;
  %end;
%end;
```

```
%do %while(%superq(out2) = %str());
 %if %superq(cond2) ne %str() %then %do;
   %if %superq(cond3) ne %str() %then %do;
      %let and2 = and;
                        %let out2 = 9;
           %end;
           %else
   %if %superq(cond4) ne %str() %then %do;
        %let and2 = and;
                                %let out2 = 9;
           %end;
           %else
   %if %superq(cond5) ne %str() %then %do;
        %let and2 = and;
                                %let out2 = 9;
           %end;
           %else
   %if %superq(cond6) ne %str() %then %do;
        %let and2 = and;
                                %let out2 = 9;
           %end;
           %else
   %if %superq(cond7) ne %str() %then %do;
        %let and2 = and;
                                %let out2 = 9;
           %end;
           %else
   %if %superq(cond8) ne %str() %then %do;
        %let and2 = and;
                                %let out2 = 9;
           %end;
           %else
   %if %superq(cond9) ne %str() %then %do;
        %let and2 = and;
                                %let out2 = 9;
           %end;
           %else %do;
                %let and2 = %str();
                                %let out2 = 9;
   %end;
 %end;
 %else %do;
                %let and2 = %str();
                                %let out2 = 9;
 %end;
%end;
```

This code includes the construction up to condition 2, but for the entire program it will be necessary to complete for each of the six conditions. The value of "9" for the MACRO variable OUT could be any value other than NULL, in order to break out of the DO LOOP. Once the program loops through each of the DO LOOPS, the SQL extract can be processed. In example 12 below, the data is queried from the collection table.

Example 12.

```
create table extract as
 select *
 from connection to DW
 select * from
 some ORACLE table
 where &cond1 &and1 &cond2 &and2 &cond3 &and3
      &cond4 &and4 &cond5 &and5 &cond6);
quit;
```

The SAS Data Set EXTRACT is created based on the parameters designated in the EIS. Now that the data is available, there are a number of presentation possibilities to consider. SAS offers a wide variety of output including TABULATE, SAS/GRAPH, PDF and HTML to name a few. Using ODS the user can direct the output to the WEB, and for the purposes of the EIS here, that would be the next step.

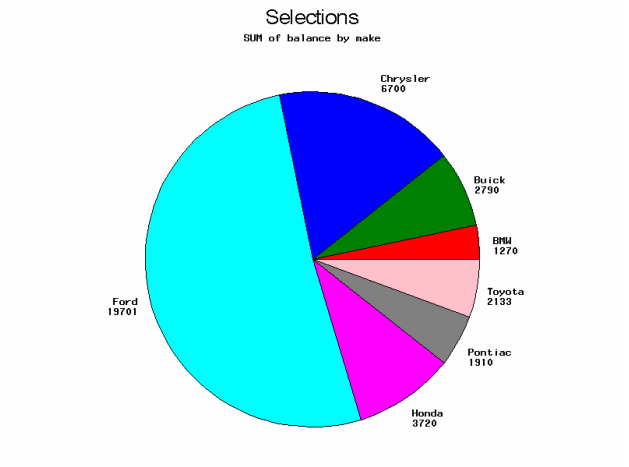
Sending Graphs and Table Output to the WEB

For a simple PIE Chart see Example 12 below.

Example 12.

```
proc gchart data=extract;
  pie make / sumvar= balance;
run;
title 'Selections';
quit;
```

Figure 13.



Each slice value in Figure 13 above is equal to the sum of the balance remaining on the lease agreement at the termination of the lease date. Because FORD has a very large balance relative to the other Makes or auto, the slice width is larger.

To send this output to the FRAME within the EIS , Example 13 is used for illustration.

Example 13.

```
ods listing close;
      ATTRIBUTES  =("codebase"="graphdirectory")
      PARAMETERS=("drilldownmode"="local"
"colorscheme"="financial");

insert SAS/GRAPH code here …..

goptions reset=all
        device   =java
        gsfname=gout
        gsfmode=replace;

ods html close;
ods listing;
```

Conclusion

An EIS application on the WEB provides an opportunity for data analytics and decision making that is far superior to running ad-hoc reports on an as-needed basis. The flexibility that SAS provides for generating output enables a number of different presentations to be displayed in the EIS output frame. It is important however to maximize the efficiency of the system by constructing WHERE clauses that extract only the appropriate data based on the user interface. With a well thought out plan, and a solid sense of the business a successful EIS can be built very easily.

References

Miron, Thomas. (1995), The How-To Book for SAS/GRAPH Software, Cary, NC; SAS Institute Inc.

Contact Information

Ed Confer
KGC Programming Solutions
P.O. Box 650283
Potomac Falls, VA 20165

Email: EdConfer1031@Yahoo.com