

# My Top 10 ways to use SAS® Stored Processes

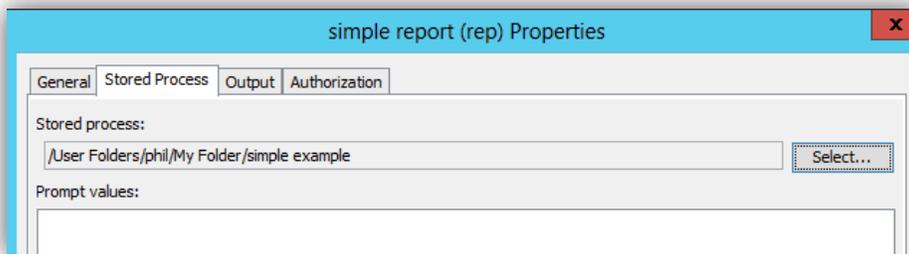
Phil Mason, Wood Street Consultants

## ABSTRACT

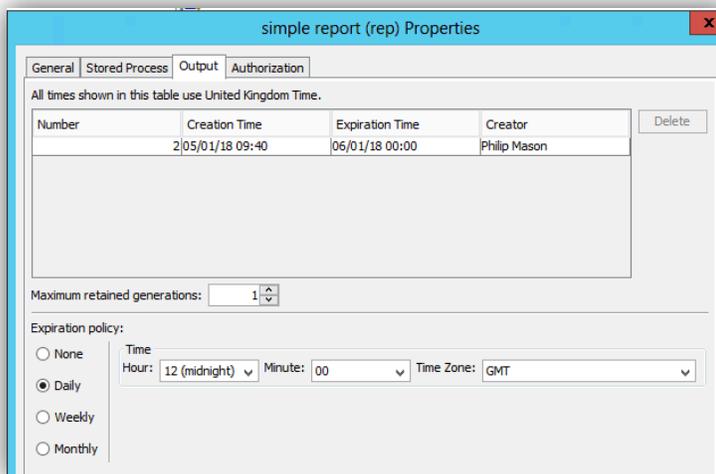
SAS® Stored Processes are a powerful facility within SAS. Having recently written a book about SAS® Stored Processes I have picked up the 10 best ways to use them so that I can illustrate the different abilities and ways to use them. I will explain how to run almost any code from your web browser, how to use them to export data from SAS® to systems like Tableau, how to optimise their use for thousands of users, how to build mobile applications, build visualisations like seen in SAS® Visual Analytics, how to make web services to integrate with other clients, and much more. All of these techniques are not generally well known, although they are not complex. If you don't already understand these techniques, then adding these to your skill set will enable you to achieve much more.

## 1. SAS® STORED PROCESS REPORTS

A SAS® stored process report is a cached version of a stored process. You define a stored process report by pointing it at a stored process. When you run the stored process report it will return the results for this stored process. If the stored process has been run within a certain timeframe then those results will come back. If the stored process was run too long ago then it will be run again to update the results, and then those results will come back.



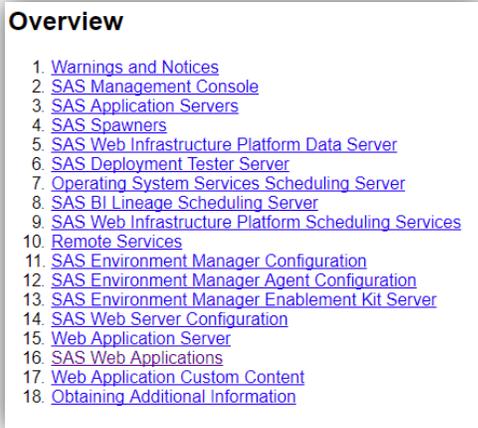
You also define an expiration policy for the stored process which defines how long the stored process results are kept. If the results have expired, then the next time the report is accessed it will be run again to freshen up the results.



You can also define how many generations of results to keep, so that you can go back and look at previous results.

## 2. RUN CODE FROM WEB BROWSER

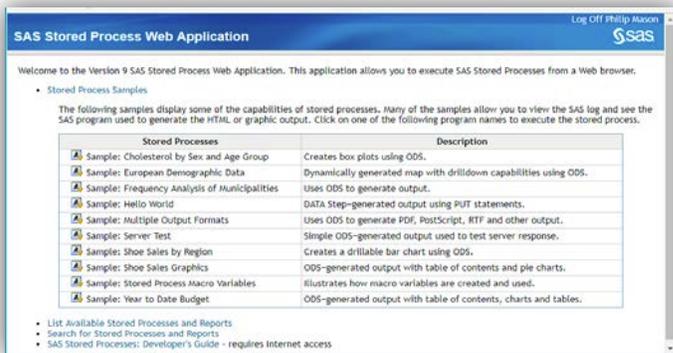
When installing SAS® an instructions.html file is created which has all the steps for installing SAS. If you have installed the SAS® Stored Process Web Application, then you will find out about it in the list of steps under the SAS® Web Applications step. For example, here is what is displayed in my file.



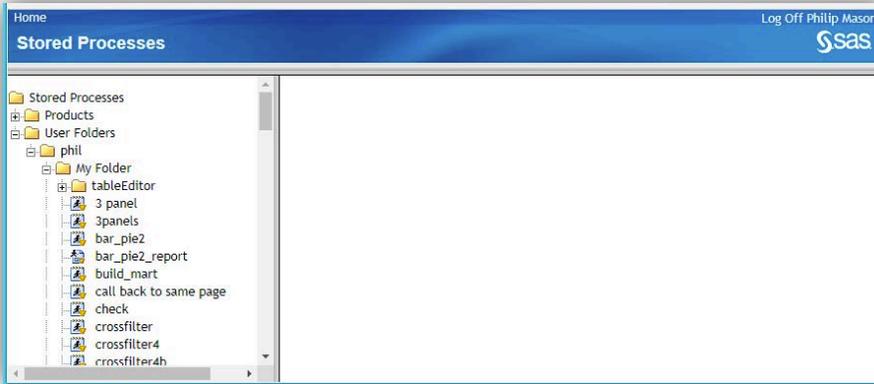
I can then go and find the link for the SAS® Stored Process Web Application.

SAS Stored Process Web Application	
<b>URL</b>	<a href="http://d351tq92-80/SASStoredProcess/do">http://d351tq92-80/SASStoredProcess/do</a>
<b>Log file</b>	<a href="C:\SAS\Config\Lev1\Web\Logs\SASServer1_1\SASStoredProcess9_4.log">C:\SAS\Config\Lev1\Web\Logs\SASServer1_1\SASStoredProcess9_4.log</a>
<b>Validation steps</b>	<ol style="list-style-type: none"> <li>1. Access the URL provided above.</li> <li>2. Log in using valid credentials defined in the SAS Metadata Server.</li> <li>3. On the Welcome page, click the "Sample: Server Test" link to access the custom input form for this sample stored process.</li> <li>4. Select an output format and ODS style or accept the default values, and click the "Display SAS Output" button.</li> <li>5. The output from the stored process is a simple message that the server processed the request.</li> </ol>

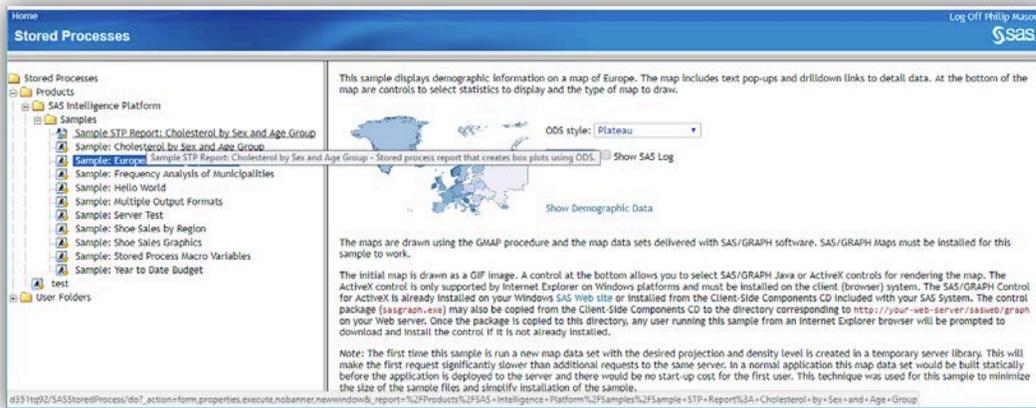
Going to that link I will see the Welcome page for the SAS® Stored Process Web Application.



Clicking on “List Available Stored Processes and Reports” displays the following screen in which you can navigate through metadata folders to find any Stored Processes and Reports you have access to.

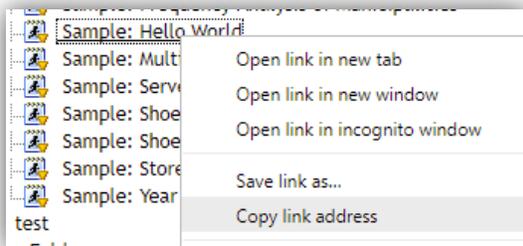


Clicking on a stored process in the left panel will run it. If there are parameters to prompt for, then they will be shown in the right panel.



The results of the stored process might be shown in the right panel or a new tab, depending on the Stored Process.

You can get the URL to run a Stored Process using the web application by right clicking on one of the stored processes in the left panel and copying the link.

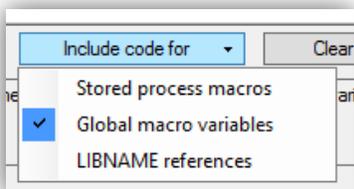


Pasting this in to a web browser will let you run the stored process without going through the general menu.

### 3. WRITING OUTPUT DIRECTLY TO BROWSER

If you create a stored process using the defaults in the wizard from Enterprise Guide, which is the most common way to do so, then it will add in the stpbegin and stpend macro calls. These setup ODS for whatever context you are using the stored process in. However, you can turn these off and take full control of what is returned when the stored process is called.

You turn them off by unchecking the “Stored process macros” item in the SAS® Code part of the wizard.



You can return content to the caller of the stored process by writing to the fileref **\_webout**. When calling the stored process using the stored process web application, writing to **\_webout** will write content directly into the web browser. This is how we can write custom HTML from a stored process.

The following code in a stored process will write some HTML to the web browser which creates a menu, allowing the user to make a selection and then call another stored process.

```
SAS Code
proc sql ;
  select distinct '<option value="'||strip(product_line)||'">'||strip(product_line)||'</option>'
  into :options separated by ' '
  from sashelp.orsales ;
quit ;
data _null_ ;
  file _webout ;
  input ;
  line=resolve(_infile_) ;
  put line ;
  cards4 ;
  <html>
  <body>
  <h1>Pick a report to run</h1>
  <form method="get" action="http://d351tq92/SASStoredProcess/do?">
  <input type="hidden" name="_program" value="/User Folders/phil/My Folder/test">
  <select name="product_line">
  &options
  </select>
  <input type="submit" value="Run">
  </form>
  </body>
  </html>
  ;;;
run ;
```

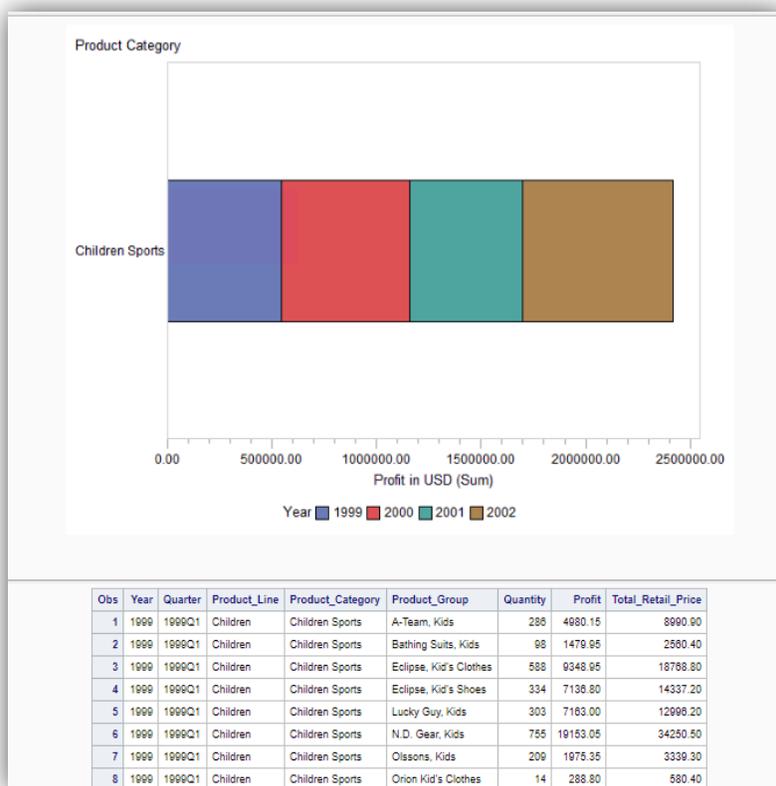
The proc sql creates some HTML tags that will have the values for product\_line so that they can be selected from a list. That data step writes to \_webout, which will write the lines to the browser. The HTML is a simple FORM that lets the user select a value for product\_line and then runs a stored process passing that value to it. Here is the menu displayed.

### Pick a report to run

Children ▼ Run

- Children
- Clothes & Shoes
- Outdoors
- Sports

Here are the results after it has run.



#### 4. RETURNING OUTPUT TO THE SAME PAGE

Usually when using stored processes in the web browser you will run them and be prompted for choices (if there are any) and then the output will be displayed in a new page. To run the stored process again you can go back to the prompts and make another selection, then producing new output on a new page.

It is useful to combine the prompts and the output onto a single page. This is sometimes called a single page application, although these can be far more complex.

There is a simple way to do this with a stored process. Picking up an example from earlier, we can modify this to prompt us for parameters and then display the output on the same page. We simply add a parameter **target='content'** to the FORM tag. This means that when the form is submitted the output will be sent to the target which is named content. Then we define an iFrame tag which can accept the content of an HTML page and show it on another page. We

specify **name='content'** in the iFrame and also give it a height and width.

```
SAS Code
data _null_ ;
  file _webout ;
  input ;
  put _infile_ ;
  cards4 ;
  <html>
  <body>
  <h1>Pick a report to run</h1>
  <form method="get" action="http://d351tq92/SASStoredProcess/do?" target='content'>
  <input type="hidden" name="_program" value="/User Folders/phil/My Folder/test">
  <select name="product_line">
  <option value="Children">Children</option>
  <option value="Clothes & Shoes">Clothes & Shoes</option>
  <option value="Outdoors">Outdoors</option>
  <option value="Sports">Sports</option>
  </select>
  <input type="submit" value="Run">
  </form>
  <iframe name="content" height="100%" width="100%">
  </iframe>
  </body>
  </html>
  ;;;
run ;
```

When a selection

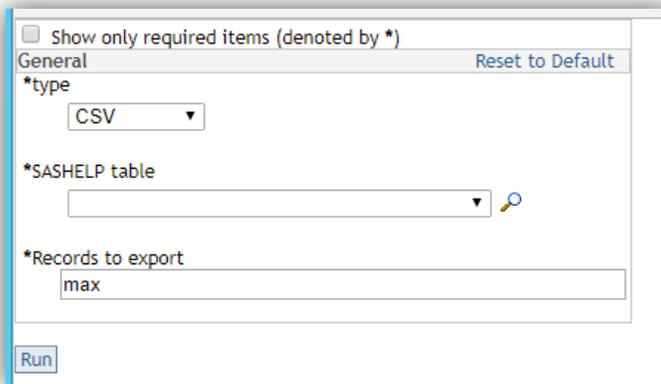
is made, and this is run, we see the following.



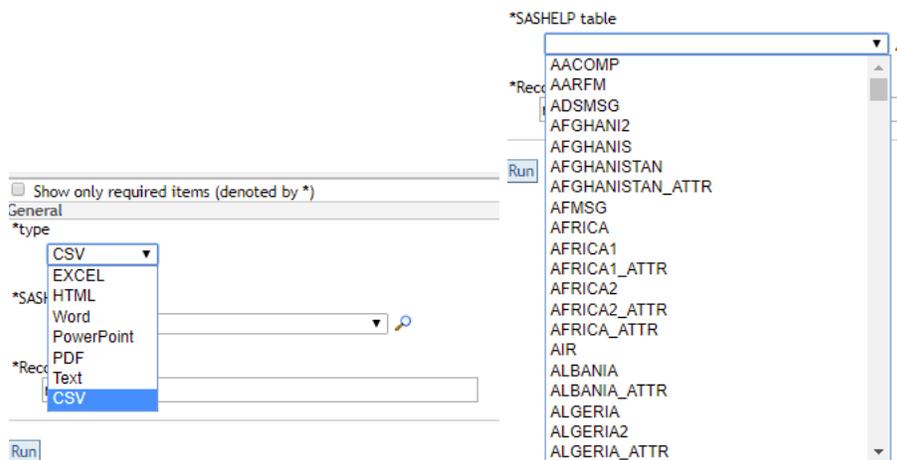
## 5. EXPORT SAS® DATA TO OTHER CLIENTS

A great use of stored processes is to provide a smooth way to get data from SAS® into some other application or format such as Microsoft EXCEL, Tableau, PDF, etc. It is easy for anyone to run and requires no SAS® knowledge or anything to be setup on their machine.

To show how to make a flexible stored process to do this I have made one whose code is included in the appendix. This is the screen that prompts you first.



You can select the type of file you want to export and the table to be exported. You can also enter the number of records to export or max if you want them all.



The stored process makes use of several techniques for producing the data in the required format. Some types use ODS and others use PROC EXPORT. It is also very important to set the HTTP headers using the http\_header function. This ensures that the application receiving the data from the stored process understands what type of data it is. For example if the header is set to PowerPoint then the data will be sent to the Microsoft PowerPoint application for opening.

See the source code in the appendix.

## 6. FEEDING SAS® DATA TO JAVASCRIPT OBJECTS

You can make use of many JavaScript objects to make visualizations, such as Crossfilter and D3.

For example, I have made a stored process called crossfilter, which displays the following prompts when run.

Show only required items (denoted by \*)

**General**

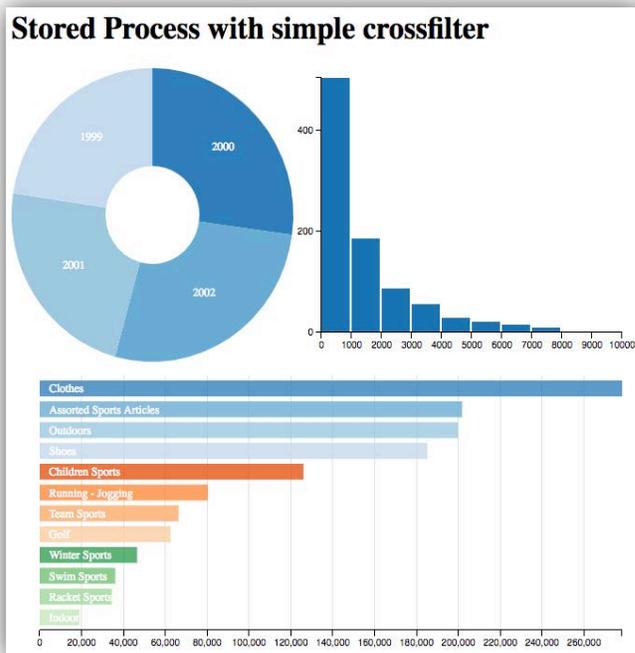
\*measure1  
Quantity

\*category\_var1  
Year

\*category\_var2  
Product\_Category

Run

This lets you choose some variables from sashelp.orsales to be used to create a visualisation. Selecting these variables makes the following interactive visualisation.



[View Page Source](#)  
Inspect

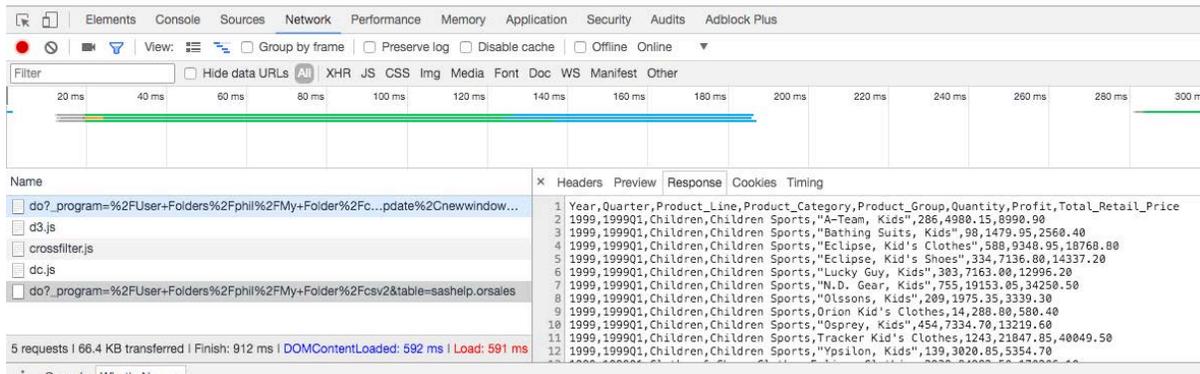
```
24 d3.csv("http://d351tq92/SASStoredProcess/do?_program=%2FUser+Folders%2Fphil%2FMy+Folder%2Fcsv2&table=sashelp.orsales", function(spendData) {
```

The code used in the stored process csv2, which feeds data to the D3 JavaScript objects has very simple code as follows:

```
proc export data=&table outfile=_webout dbms=csv replace ;
run ;
```

This code just uses PROC EXPORT to send the table passed in as a parameter to the file `_webout` in CSV format. `_webout` is used to send output to the browser when used with the SAS® Stored Process web application, so using it in this way just returns the CSV data to the JavaScript function that calls it. The `replace` parameter is needed since `_webout` already exists and otherwise it won't work with `proc export`.

You can use the Developer Tools in your web browser to look at what is going on behind the scenes. Here I am using Chrome, and I can use the Network tab to see what calls are made when running the stored process. I have chosen the stored process that provides the data to the JavaScript function, and I can look at the response to see the CSV data produced by PROC EXPORT.



Some other JavaScript objects require different data, such as the grids in `jqgrid` which need JSON data. Here is some JSON data I use in one of the grids produced by PROC JSON.

```
{
  "rows": [
    {
      "Year": 1999,
      "Quarter": "1999Q1",
      "Product_Line": "Children",
      "Product_Category": "Children Sports",
      "Product_Group": "A-Team, Kids",
      "Quantity": 286,
      "Profit": 4988.15,
      "Total_Retail_Price": 8990.9
    },
    {
      "Year": 1999,
      "Quarter": "1999Q1",
      "Product_Line": "Children",
      "Product_Category": "Children Sports",
      "Product_Group": "Bathing Suits, Kids",
      "Quantity": 98,
      "Profit": 1479.95,
      "Total_Retail_Price": 2560.4
    }
  ]
}
```

## 7. MAKE TABLES IN HTML WORK PROPERLY

When viewing a table in HTML produced by a stored process you will typically run into some problems when the table is large. For example just doing a `proc print` of `sashelp.orsales` from a stored process would produce this.

Obs	Year	Quarter	Product_Line	Product_Category	Product_Group	Quantity
1	1999	1999Q1	Children	Children Sports	A-Team, Kids	298
2	1999	1999Q1	Children	Children Sports	Bathing Suits, Kids	98
3	1999	1999Q1	Children	Children Sports	Eclipse, Kid's Clothes	588
4	1999	1999Q1	Children	Children Sports	Eclipse, Kid's Shoes	334
5	1999	1999Q1	Children	Children Sports	Lucky Guy, Kids	303
6	1999	1999Q1	Children	Children Sports	N.D. Gear, Kids	755
7	1999	1999Q1	Children	Children Sports	Olsons, Kids	209
8	1999	1999Q1	Children	Children Sports	Orion Kid's Clothes	14
9	1999	1999Q1	Children	Children Sports	Osprey, Kids	454
10	1999	1999Q1	Children	Children Sports	Tracker Kid's Clothes	1243
11	1999	1999Q1	Children	Children Sports	Ypsilon, Kids	139
12	1999	1999Q1	Clothes & Shoes	Clothes	Eclipse Clothing	2038
13	1999	1999Q1	Clothes & Shoes	Clothes	Green Tomato	171
14	1999	1999Q1	Clothes & Shoes	Clothes	Knitwear	1654
15	1999	1999Q1	Clothes & Shoes	Clothes	LSF	335
16	1999	1999Q1	Clothes & Shoes	Clothes	Leisure	312
17	1999	1999Q1	Clothes & Shoes	Clothes	Massif	25

If I scroll down and to the right I am left with something that looks like this. However as you can see we no longer have the row numbers displayed as they scrolled off to the left, and the column titles have scrolled off at the top.

othes & ioes	Clothes	Leisure	312	14394.30	28047.20
othes & ioes	Clothes	Massif	25	3787.90	7192.10
othes & ioes	Clothes	Orion	793	38086.35	72236.50
othes & ioes	Clothes	Orion Clothing	460	11850.25	22294.30
othes & ioes	Clothes	Osprey	266	9218.75	15712.80
othes & ioes	Clothes	Shorts	150	2278.65	4063.00
othes & ioes	Clothes	Stockings & Socks	237	2099.55	3564.90
othes & ioes	Clothes	Street Wear	408	15325.30	26391.80
othes & ioes	Clothes	T-Shirts	362	5829.10	10497.10
othes & ioes	Clothes	Tracker Clothes	2624	89039.20	127046.80
othes & ioes	Clothes	Twain	71	9522.57	18878.33
othes & ioes	Shoes	Eclipse Shoes	4121	205449.95	413796.40
othes & ioes	Shoes	Shoes	1905	98815.55	183279.20
othes & ioes	Shoes	Tracker Shoes	2179	82492.55	170200.40
othes & ioes	Outdoors	Anoraks & Parkas	3139	236288.94	420621.54

I have heard many a manager over my years complain about SAS® tables displayed like this, and insist on them being in EXCEL so that the titles can always be seen, data can be sorted, and so on.

By using a JavaScript object such as DataTables (from <https://datatables.net/examples/index>) we can display a table in a web page and then feed that object with data from another stored process. That enables us to produce something like this.

Show  entries Search:

Name	Position	Office	Extn.	Start date	Salary
Airi Satou	Accountant	Tokyo	5407	2008/11/28	\$162,700
Angelica Ramos	Chief Executive Officer (CEO)	London	5797	2009/10/09	\$1,200,000
Ashton Cox	Junior Technical Author	San Francisco	1562	2009/01/12	\$86,000
Bradley Greer	Software Engineer	London	2558	2012/10/13	\$132,000
Brenden Wagner	Software Engineer	San Francisco	1314	2011/06/07	\$206,850
Brielle Williamson	Integration Specialist	New York	4804	2012/12/02	\$372,000
Bruno Nash	Software Engineer	London	6222	2011/05/03	\$163,500
Caesar Vance	Pre-Sales Support	New York	8330	2011/12/12	\$106,450
Cara Stevens	Sales Assistant	New York	3990	2011/12/06	\$145,600
Cedric Kelly	Senior Javascript Developer	Edinburgh	6224	2012/03/29	\$433,060

Showing 1 to 10 of 57 entries

Previous  2 3 4 5 6 Next

The key JavaScript part that defines the table and where its data comes from is like this.

```
<script>
$(document).ready(function() {
  $('#table1').DataTable({
    "ajax": 'http://d351tq92/SASStoredProcess/do?_program=%2FUser+Folders%2Fphil%2FMy+Folder%2Fdatatables_data'
  });
});
</script>
```

This object just shows what fits on the screen and lets us move through other pages of data. It is produced by a stored process called DataTables (code in appendix). We can also sort by the values in columns. There are many options available with this object so we can actually achieve much of what can be done in EXCEL as far as viewing tables.

The data is provided by a second stored process called DataTables\_data (code in appendix). This data comes from the stored process in JSON format, and is consumed by the JavaScript table. Not only does it provide a better looking table, but it gets past any objections people have about functionality.

By adding a few statements to the JavaScript we can get a scroll bar, rather than using the default paging.

```
<script>
$(document).ready(function() {
  $('#table1').DataTable({
    "ajax": 'http://d351tq92/SASStoredProcess/do?_program=%2FUser+Folders%2Fphil%2FMy+Folder%2Fdatatables_data',
    "scrollY": "800px",
    "scrollCollapse": true,
    "paging": false
  });
});
</script>
```

This produces this table, which many people prefer.

Search:

Name ▲	Position ◆	Office ◆	Extn. ◆	Start date ◆	Salary ◆
Airi Satou	Accountant	Tokyo	5407	2008/11/28	\$162,700
Angelica Ramos	Chief Executive Officer (CEO)	London	5797	2009/10/09	\$1,200,000
Ashton Cox	Junior Technical Author	San Francisco	1562	2009/01/12	\$86,000
Bradley Greer	Software Engineer	London	2558	2012/10/13	\$132,000
Brenden Wagner	Software Engineer	San Francisco	1314	2011/06/07	\$206,850
Brielle Williamson	Integration Specialist	New York	4804	2012/12/02	\$372,000
Bruno Nash	Software Engineer	London	6222	2011/05/03	\$163,500
Caesar Vance	Pre-Sales Support	New York	8330	2011/12/12	\$106,450
Cara Stevens	Sales Assistant	New York	3990	2011/12/06	\$145,600
Cedric Kelly	Senior Javascript Developer	Edinburgh	6224	2012/03/29	\$433,060
Charde Marshall	Regional Director	San Francisco	6741	2008/10/16	\$470,600

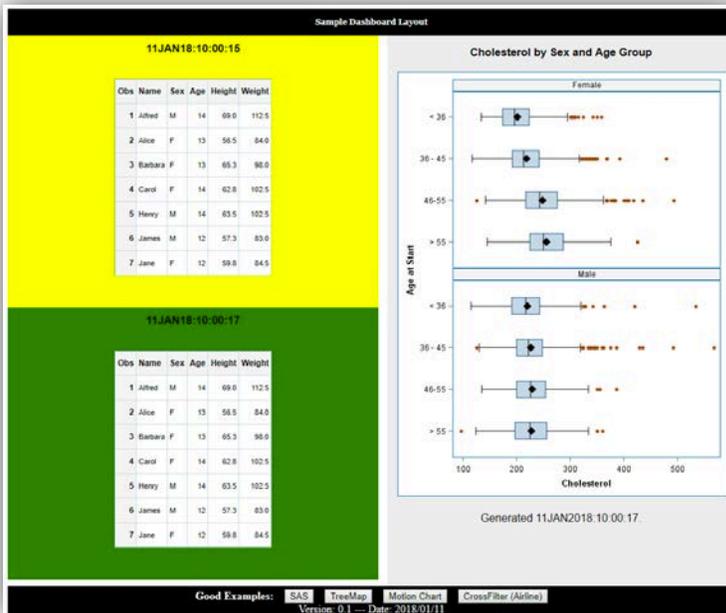
Showing 1 to 57 of 57 entries

## 8. PUTTING TOGETHER A DASHBOARD

There are a number of ways to make a simple dashboard. The key thing about dashboards is that the screen is divided into parts and each part is filled with some information. That information could be a graph, table, indicator, symbol, number, color, etc. You can use your imagination really. To divide the screen up like this you could create a single image using SAS/Graph annotate facility which has all the information in the image. But a better way is to use a web page and populate sections of the page with different content.

We can create parts of a page with stored processes. Each of these parts can be placed on a page by creating a layout with CSS. Each part can be loaded using AJAX, which will load them as required. This gives us lots of flexibility in that each part could contain almost anything, and include interactive functionality or not as required.

The following screen shot shows a simple dashboard layout which is populated by running 3 stored processes.



Here is the code used to create it. The key things to note in the code are highlighted in **bold**.

- We point to the jQuery library, which allows us to use the .load functions to easily load sections of the HTML with the output of our stored processes.
- In the CSS code by using height, float and width we are able to define where our HTML sections are laid out on the screen.
- By using the jQuery .click functions we can define what happens when the pushbuttons are pressed. The .open functions will open a new window with the link.

```

<!DOCTYPE html>
<meta charset="utf-8">
<head>
  <script type="text/ecmascript" src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-
  2.1.1.min.js"></script>
<style>
#header {
  background-color:black;
  font: bold 12px/30px Georgia, serif;
  color:white;
  text-align:center;
  padding:5px;
}
#aside {
  width: 45%;
  height: 600px;
  background-color:#eee;
  float:right;
  padding:15px;
}
#section1 {
  line-height:30px;
  background-color:#fff;
  height:400px;
  width:50%;
  float:left;
  padding:5px;
}
#section2 {
  font: bold 12px/30px Georgia, serif;
  line-height:30px;
  background-color:#fff;
  height:200px;
  
```

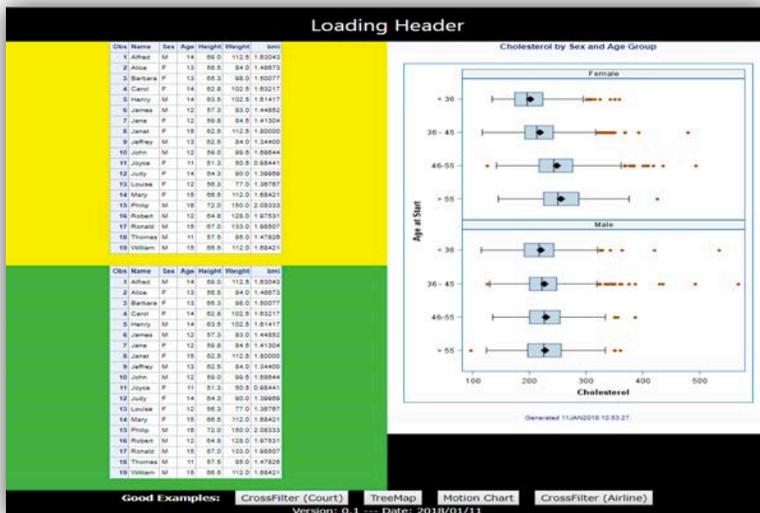


```

<button id="link4">CrossFilter (Airline)</button>
<div id="now"></div>
</footer>
</body>
</html>

```

We could improve this code a bit by using a CSS framework to lay it out. This does a lot of the hard work for you, in the same way that jQuery does the hard JavaScript work for you. So we can achieve the following output using a CSS framework.



Here is the code, with some important parts bolded. We have been able to simplify the HTML somewhat and replace the CSS directives with some W3.CSS directives contained in class= parameters on HTML tags. We can use the W3.CSS directives to choose background color, alignment of content, how much space sections take up across the page, and much more.

Check <https://www.w3schools.com/w3css/default.asp> for info.

```

<!DOCTYPE html>
<meta charset="utf-8">
<head>
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  <script type="text/ecmascript" src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-2.1.1.min.js"></script>
</script>
$(document).ready(function() {

$("#aside").load("http://my_server/SASStoredProcess/do?_program=%2FProducts%2FSAS+Intelligence+Platform%2FSamples%2FSample%3A+Cholesterol+by+Sex+and+Age+Group") ;

$("#section1").load("http://my_server/SASStoredProcess/do?_program=%2FUser+Folders%2Fphil%2FMy+Folder%2Ftest9") ;

$("#section2").load("http://my_server/SASStoredProcess/do?_program=%2FUser+Folders%2Fphil%2FMy+Folder%2Ftest9") ;
  $('#link1').click(function() {
    window.open('http://support.sas.com/', '_blank') ;
  })
  $('#link2').click(function() {

window.open('http://mbostock.github.io/d3/talk/20111018/treemap.html', '_blank') ;
  })
  $('#link3').click(function() {

```



If you put **?wsdl** on the end of the URL for the web service then it will return the Web Service Description Language for that web service (stored process) so that it can be used by a client. The URL would look like this.

[http://my\\_server/SASBIWS/services/User+Folders/phil/My+Folder/webservice1?wsdl](http://my_server/SASBIWS/services/User+Folders/phil/My+Folder/webservice1?wsdl)

To use a web service, you can use some code like this, demonstrated on the “Who’s in space, news feed” web service.

```
filename resp temp;

/* Neat service from Open Notify project */
proc http
  url="http://api.open-notify.org/astros.json"
  method= "GET"
  out=resp;
run;

/* Assign a JSON library to the HTTP response */
libname space JSON fileref=resp;
data _null_ ;
  infile resp ;
  input ;
  put _infile_ ;
run ;

/* Print result, dropping automatic ordinal metadata */
title "Who is in space right now? (as of &sysdate)";
proc print data=space.people (drop=ordinal:);
run;
```

To read in JSON data from a web service that you might make can be done easily with the JSON libname engine. Here is some sample JSON data.

```
{
  "name": "Phil Mason",
  "home": {
    "town": "Wallingford",
    "phone": 1491824891,
    "current home": true,
    "ages": [
      51,
      46,
      18,
      16,
      13
    ]
  }
}
```

Here is some code to read it in. Note that a number of tables are mapped to display the JSON data.

```
* generate a map to see what is in the JSON, and what datasets are automatically made
;
filename ex 'C:\Users\phil\Documents\My SAS® Files\9.4\exsmple JSON.sas' ;
filename jmap temp ;
libname in JSON fileref=ex map=jmap automap=create ;

* look at the map that was generated ;
```

```

data _null_ ;
  infile jmap ;
  input ;
  put _infile_ ;
run ;

* look at each of the tables that were mapped ;
title 'in.root' ;
proc print data=in.root ;
run ;
title 'in.home' ;
proc print data=in.home ;
run ;
title 'in.home_ages' ;
proc print data=in.home_ages ;
run ;

```

Produces this output

in.root		
Obs	ordinal_root	name
1	1	Phil Mason

in.home					
Obs	ordinal_root	ordinal_home	town	phone	current_home
1	1	1	Wallingford	1491824891	1

in.home_ages							
Obs	ordinal_home	ordinal_ages	ages1	ages2	ages3	ages4	ages5
1	1	1	51	46	18	16	13

## 10. USING INPUT STREAMS, OUTPUT STREAMS AND OUTPUT PARAMETERS

If you have the Microsoft Office SAS® add-in then you can use stored processes to read data in from EXCEL, process it and write data back out.

Here is the SAS® source code I have used to demonstrate this technique.

- My libname statement points to the fileref instr using the XML engine, and it will read the specified data in from EXCEL.
- &\_webin\_sasname is an automatically populated macro variable holding the table name for the EXCEL data I am reading in.
- I set a macro variable call out\_value which I will be able to write into an EXCEL cell, along with 2 other automatic macro variables.
- My Proc Print will produce some ODS output to be written back to EXCEL too.

```

libname instr xml ;
data input_from_excel(drop=i) ;
  set instr.&_webin_sasname ;
  array numbers(*) _numeric_ ;
  do i=1 to dim(numbers) ;
    calculated_total+numbers(i) ;
  end ;
run ;

```

```

* ODS output ;
title "Data read in from EXCEL, numbers totalled and written back to EXCEL" ;
title2 "Ran at %sysfunc(datetime()),datetime." ;
proc print ;
run;

** Get the number of observations in the table, and assign to output parameter ;
%let dsid=%sysfunc(open(input_from_excel)) ;
%let out_value=%sysfunc(attrn(&dsid,nobs)) ;
%let dsid=%sysfunc(close(&dsid)) ;

```

When defining your stored process you can define some output parameters, which will be able to be written to EXCEL in specific cells.

Output Parameters:		
Name	Type	Displayed Text
out_value	String	Number of observations
_CLIENTUSERID	String	Client Userid
_CLIENTUSERNAME	String	Client User Name

You can define a data source, which you will be able to select a range from an EXCEL sheet for this and then the data in that range will be read in to SAS® for use there.

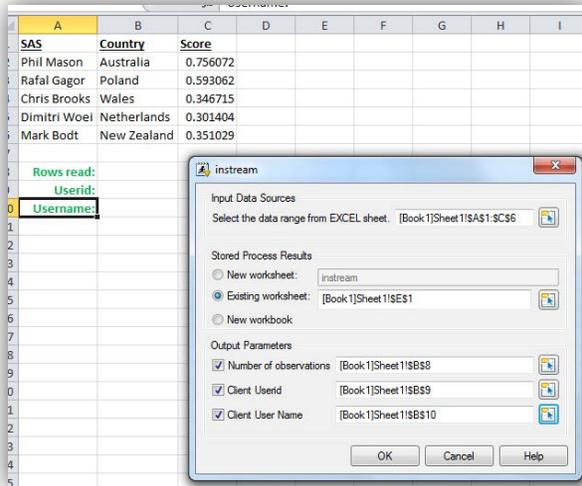
Data Sources and Targets			
Data Sources (input streams to a stored process):			
Fileref / Table Parameter	Content	Label	Description
instr	text/xml	Select the data rang...	

When defining a data source to use from EXCEL you should choose XML based data, give it a fileref (which you will then refer to from the SAS® code in your stored process) and tick the box for allow rewinding stream.

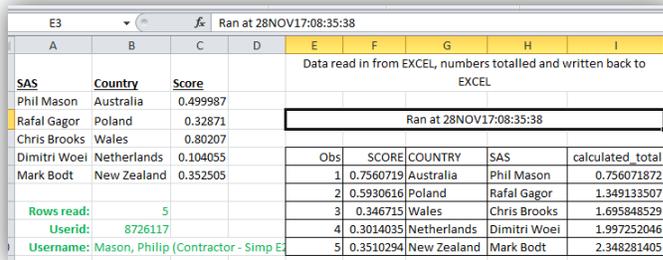
From the SAS® add-in toolbar you can click on reports and choose the stored process to run.



The stored process will prompt you to select the input data source. In this example I have made up a table of random values. You also select the destination for the ODS results that the stored process will produce. Finally you can select cells for any output parameters you might like to use.



After running the stored process you can see the results produced here.



## CONCLUSION

I hope you enjoyed this paper. Look out for my book about Stored Processes which is being published by SAS Publishing in 2018 Q2..

## CONTACT INFORMATION

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

**Name:** Philip Mason  
**Email:** [phil@woodstreet.org.uk](mailto:phil@woodstreet.org.uk)

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.