Paper SAS1832-2015

# What's New in SAS® Studio?

Mike Porter, Amy Peters, and Michael Monaco, SAS Institute Inc., Cary, NC

## ABSTRACT

If you have not had a chance to explore SAS® Studio yet, or if you're anxious to see what's new, this paper gives you an introduction to this new browser-based interface for SAS® programmers and a peek at what's coming. With SAS Studio, you can access your data files, libraries, and existing programs, and you can write new programs while using SAS software behind the scenes. SAS Studio connects to a SAS server in order to process SAS programs. The SAS server can be a hosted server in a cloud environment, a server in your local environment, or a copy of SAS on your local machine. It's a comfortable environment for those used to the traditional SAS windowing environment (SAS® Display Manager) but new features like a query window, process flow diagrams, and tasks have been added to appeal to traditional SAS® Enterprise Guide® users.

## INTRODUCTION

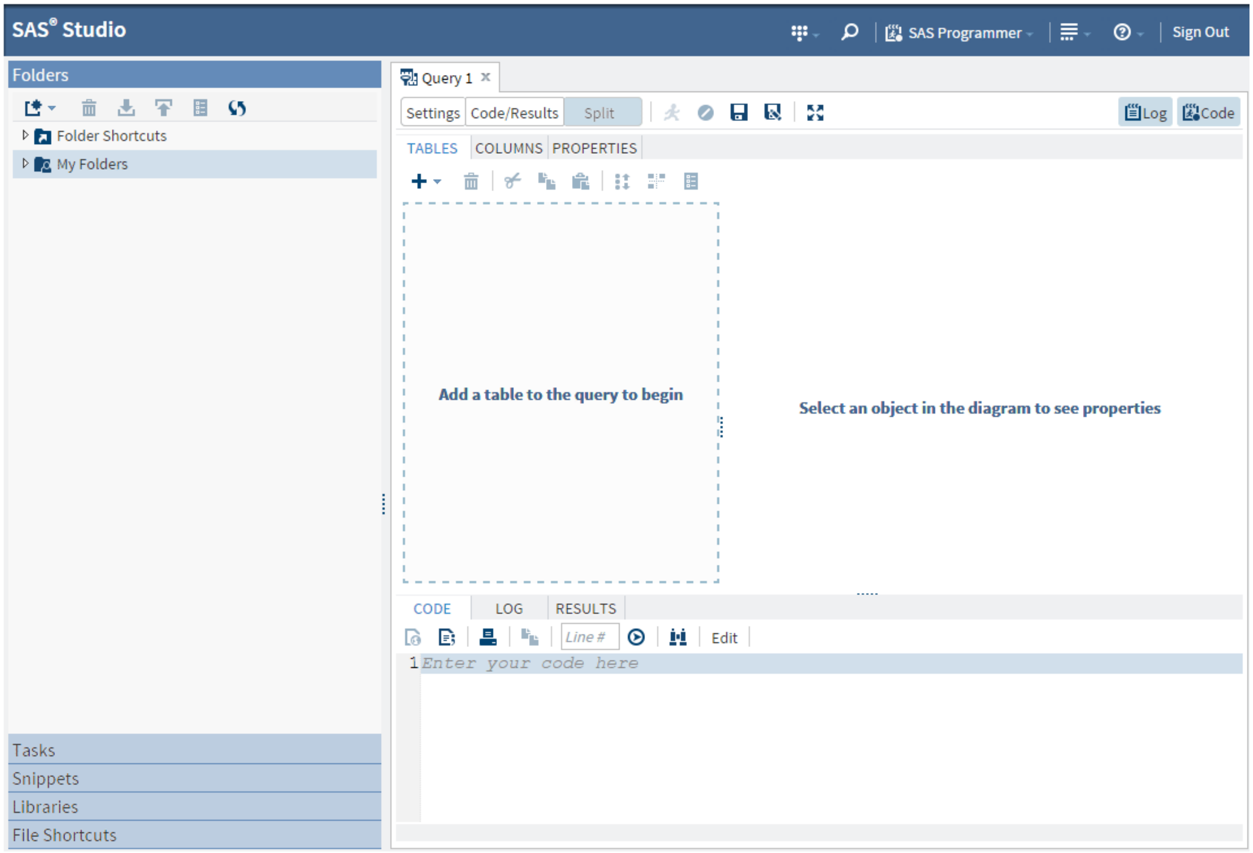
SAS® Studio is a SAS developer environment that runs in a web browser, enabling developers to program and interact with SAS wherever and whenever they want. SAS Studio delivers a consistent experience from any computing device, with always-available productivity features integrated into the framework. Writing and running SAS code has never been faster or easier. SAS Studio is not a stand-alone product, but an interface that comes with every SAS product and solution (whenever Base SAS® is included). SAS Studio is:

- Consistent. Learn one interface that you can use throughout your career—as a student, an individual SAS user or consultant, a departmental user, and an enterprise user.
- Available. You can use the same interface wherever you need it—my Mac in my dorm, my Windows desktop at work, and my laptop at home.
- Assistive. For programmers, the code is front and center, but you can use functions that are similar to SAS® Enterprise Guide® functions, such as code-generating tasks or process flows to help if you need them.

SAS Studio began shipping in March 2014 and over the course of several releases is now available on all platforms SAS supports. SAS Studio can be used to access a local copy of SAS on a desktop machine on Windows, to access SAS running on a Windows or UNIX server, or as part of a full SAS Intelligence Platform enterprise deployment. SAS Studio 3.3 was released in February 2015 and this paper focuses on the new features available in this release.

## USING THE QUERY BUILDER TO GENERATE SQL

You can use Structured Query Language (SQL) in your SAS programs via PROC SQL, and you can get help generating PROC SQL in SAS Studio via the new Query Builder. Let's start with a basic query. On the main SAS Studio toolbar, click ▦ and select **New Query**. The Query Builder opens on a new tab in the work area.



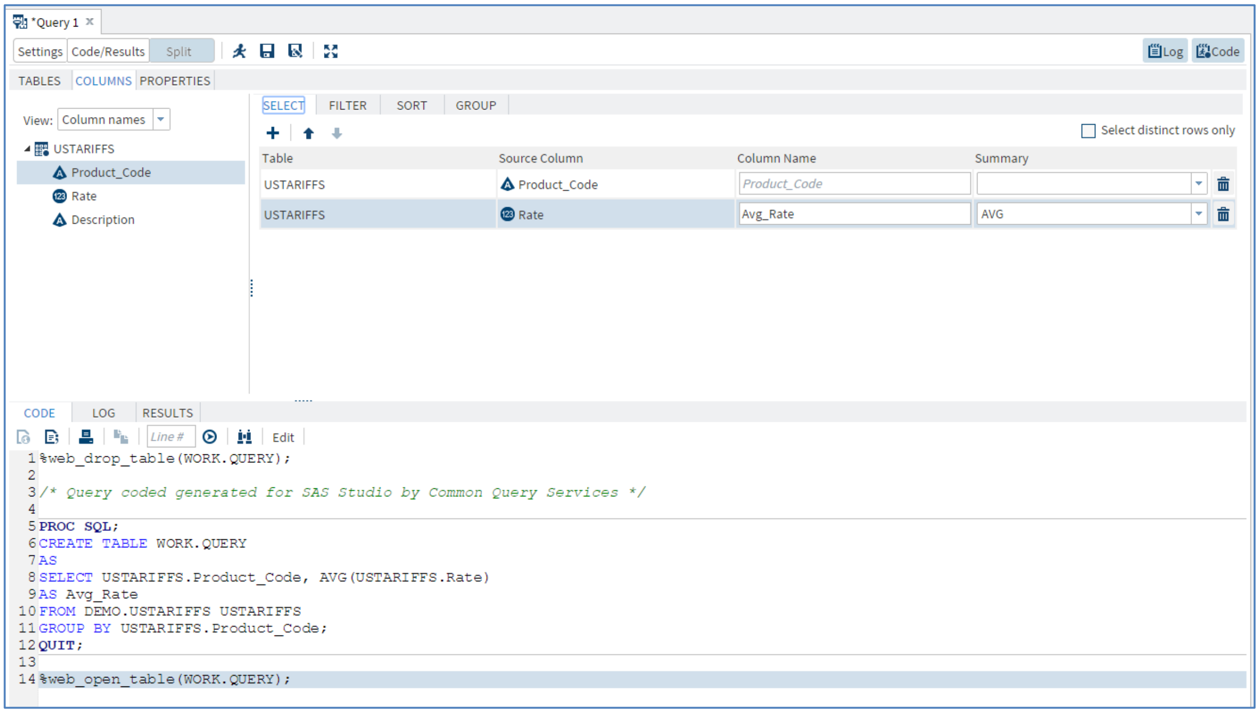
Add a table to the query to begin

Select an object in the diagram to see properties

```
1 Enter your code here
```

**Figure 1: Starting the Query Builder**

You can drag a SAS table from the Libraries section of the navigation pane (not from the Folders section) or open one from the Tables tab in the Query Builder by clicking + and selecting **Table**. For this example, we use the DEMO.USTARIFFS table. Switch to the Columns tab to determine what columns to include in the result. In this example, we include PRODUCT_CODES and RATES. Change RATES to display the average rather than individual values, and then rename it AVG_RATES. To get average rates for each PRODUCT_CODE, use the Group tab. You can watch the SQL code being built. Here is what it looks like once you are done:



| Table | Source Column | Column Name | Summary |
|---|---|---|---|
| USTARIFFS | Product_Code | Product_Code | |
| USTARIFFS | Rate | Avg_Rate | AVG |

```
%web_drop_table(WORK.QUERY);

/* Query coded generated for SAS Studio by Common Query Services */

PROC SQL;
CREATE TABLE WORK.QUERY
AS
SELECT USTARIFFS.Product_Code, AVG(USTARIFFS.Rate)
AS Avg_Rate
FROM DEMO.USTARIFFS USTARIFFS
GROUP BY USTARIFFS.Product_Code;
QUIT;

%web_open_table(WORK.QUERY);
```

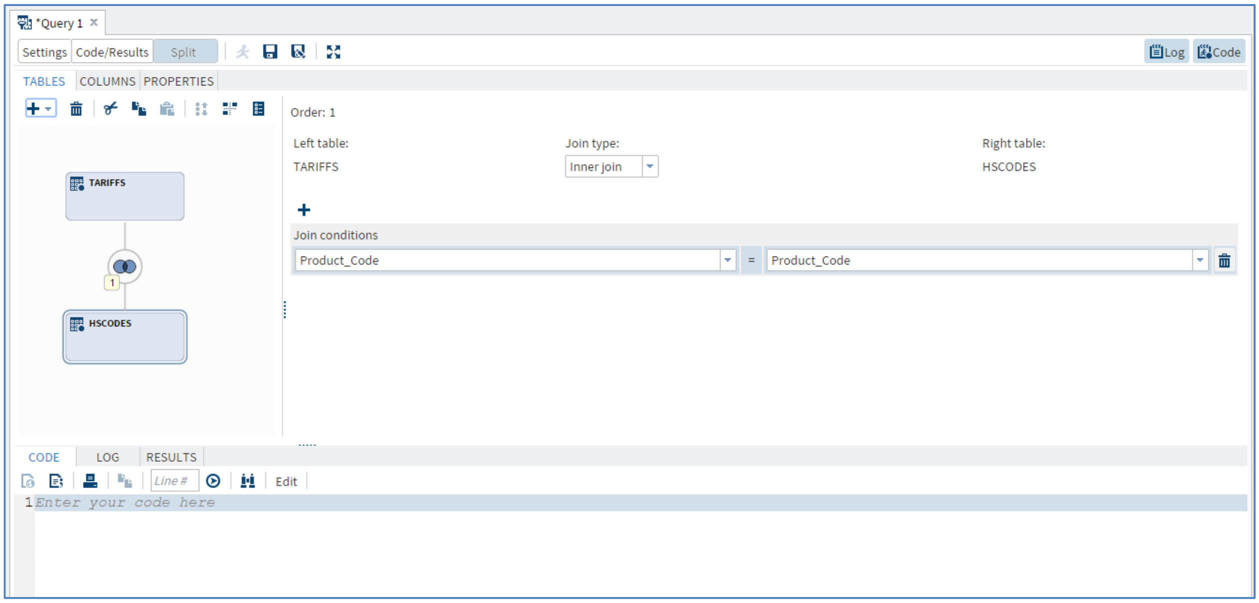
**Figure 2: Generating a Basic Query with the Query Builder**

By default, a table is created in your Work library so that when you run the query, the table opens in the table viewer. You can use the Properties tab to change your results to be a view or a report instead.

## USING THE QUERY BUILDER TO JOIN TABLES

One of the most powerful features of SQL is its ability to combine tables in many ways. The Query Builder can help you start to understand what is possible or give you some help if you have forgotten the syntax.

From the Libraries section of the navigation pane, drag the DEMO.TARIFFS table to the Query Builder, and then drag DEMO.HSCODES on top of it. The Query Builder recognizes you want to perform a join and finds the common column PRODUCT_CODE to join on.



Order: 1

Left table: TARIFFS

Join type: Inner join

Right table: HSCODES

Join conditions

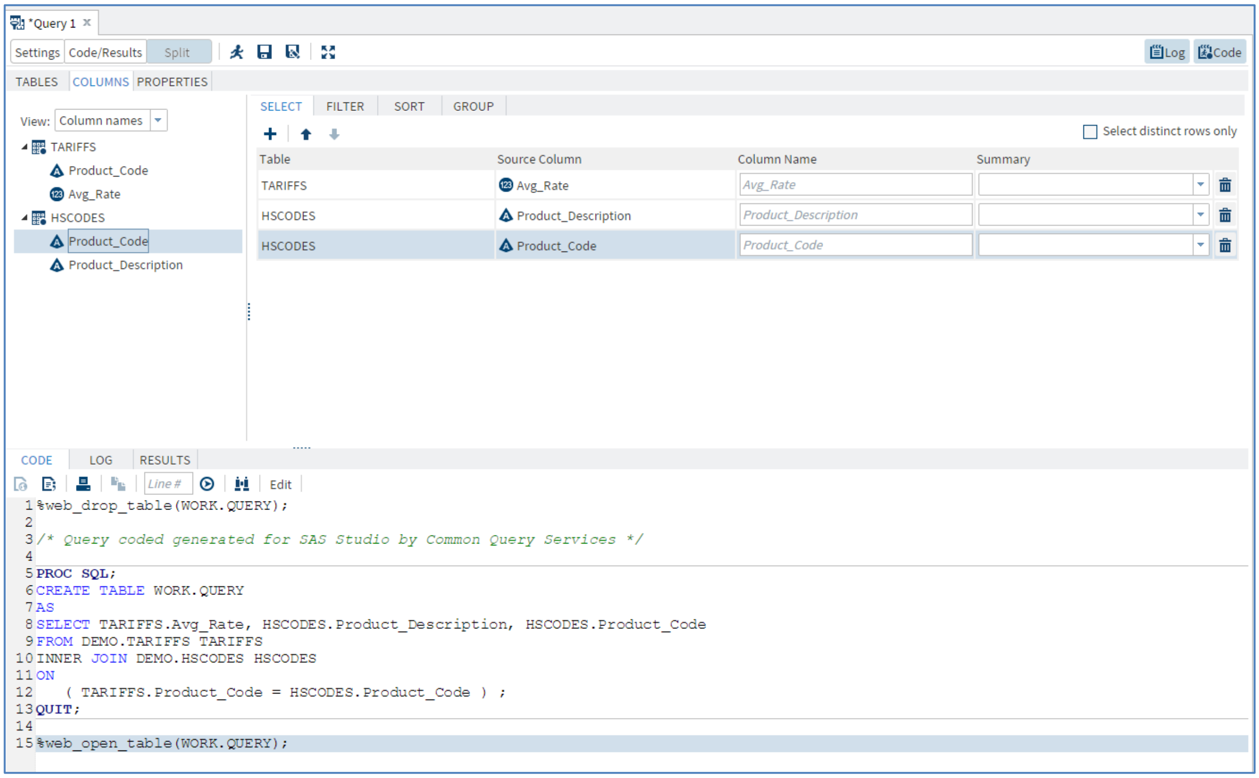
Product_Code = Product_Code

```
1 Enter your code here
```

**Figure 3: Joining Tables with the Query Tool**

You can also initiate a join by clicking ✚ on the toolbar and selecting **Join**, which opens the Join window.

Notice no code has been generated yet. We didn’t mention this in the earlier example, but until you select what columns you want in the result, there is no code generated nor anything to run. As before, use the Columns tab to select what you want. This time you notice all of the columns from both tables are available to choose from. The order in which you choose the columns determines the order they appear in the result.



| Table | Source Column | Column Name | Summary |
|---|---|---|---|
| TARIFFS | Avg_Rate | Avg_Rate | |
| HSCODES | Product_Description | Product_Description | |
| HSCODES | Product_Code | Product_Code | |

```
%web_drop_table(WORK.QUERY);

/* Query coded generated for SAS Studio by Common Query Services */

PROC SQL;
CREATE TABLE WORK.QUERY
AS
SELECT TARIFFS.Avg_Rate, HSCODES.Product_Description, HSCODES.Product_Code
FROM DEMO.TARIFFS TARIFFS
INNER JOIN DEMO.HSCODES HSCODES
ON
    ( TARIFFS.Product_Code = HSCODES.Product_Code ) ;
QUIT;

%web_open_table(WORK.QUERY);
```

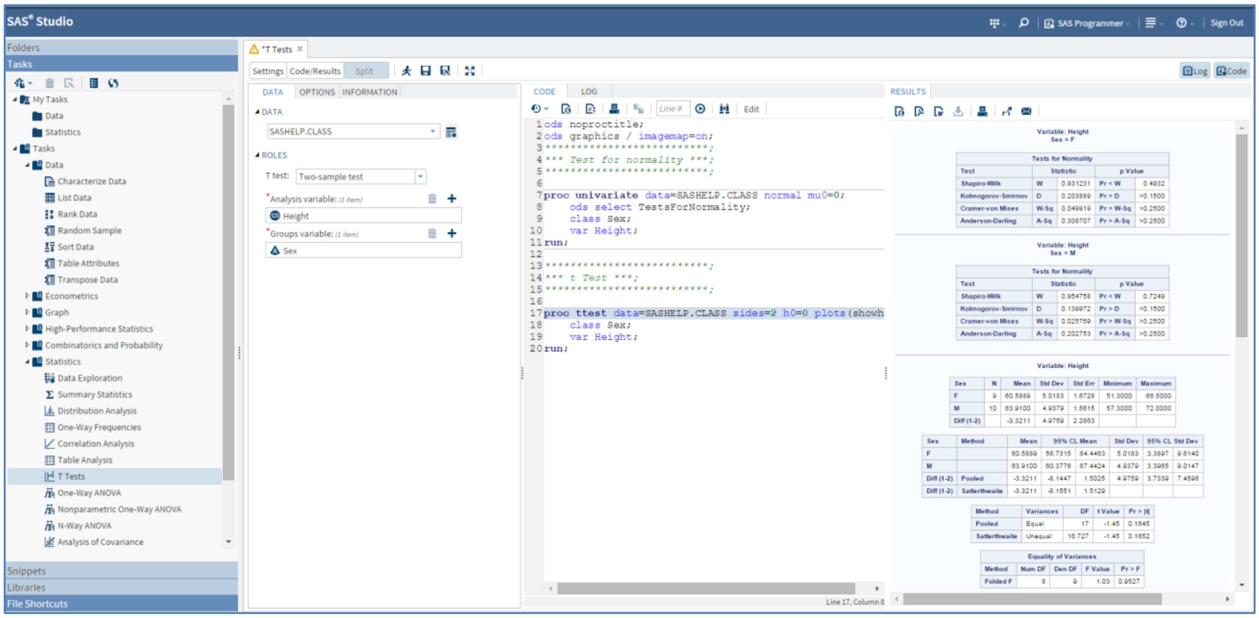
**Figure 4: Choosing Columns for the Result in the Query Builder**

If you save a query, a CQY file is created that you can bring into another SAS Studio session.

## MORE AND BETTER TASKS

Tasks in SAS Studio are basic front ends to SAS code. Because tasks are open to anyone to create, the different R&D groups around SAS have been busy adding tasks in their areas of expertise. You can use tasks as point-and-click tools to get something done, or you can use tasks as syntax help to generate SAS code that you then take and edit. Because you can watch the SAS code being built, tasks are a great way to learn about a topic. Let's take a quick tour of some of the new tasks that have been added in the statistics area.

You can use the new t Test task to perform a one-sample *t* test, paired sample *t* test, or two-sample *t* test. We will use the two-sample *t* test to analyze the height values for males and females in the SASHELP.CLASS data set.



DATA: SASHELP.CLASS

ROLES

T test: Two-sample test

*Analysis variable: (1 item) Height

*Groups variable: (1 item) Sex

```
ods noproctitle;
ods graphics / imagemap=on;
**************************;
*** Test for normality ***;
**************************;

proc univariate data=SASHELP.CLASS normal mu0=0;
    ods select TestsForNormality;
    class Sex;
    var Height;
run;

**************************;
*** t Test ***;
**************************;

proc ttest data=SASHELP.CLASS sides=2 h0=0 plots(showh
    class Sex;
    var Height;
run;
```

Variable: Height
Sex = F

| Tests for Normality | | | | |
|---|---|---|---|---|
| Test | Statistic | | p Value | |
| Shapiro-Wilk | W | 0.931231 | Pr < W | 0.4932 |
| Kolmogorov-Smirnov | D | 0.203889 | Pr > D | >0.1500 |
| Cramer-von Mises | W-Sq | 0.049919 | Pr > W-Sq | >0.2500 |
| Anderson-Darling | A-Sq | 0.308707 | Pr > A-Sq | >0.2500 |

Variable: Height
Sex = M

| Tests for Normality | | | | |
|---|---|---|---|---|
| Test | Statistic | | p Value | |
| Shapiro-Wilk | W | 0.954758 | Pr < W | 0.7249 |
| Kolmogorov-Smirnov | D | 0.139972 | Pr > D | >0.1500 |
| Cramer-von Mises | W-Sq | 0.025769 | Pr > W-Sq | >0.2500 |
| Anderson-Darling | A-Sq | 0.202753 | Pr > A-Sq | >0.2500 |

Variable: Height

| Sex | N | Mean | Std Dev | Std Err | Minimum | Maximum |
|---|---|---|---|---|---|---|
| F | 9 | 60.5889 | 5.0183 | 1.6728 | 51.3000 | 66.5000 |
| M | 10 | 63.9100 | 4.9379 | 1.5615 | 57.3000 | 72.0000 |
| Diff (1-2) | | -3.3211 | 4.9759 | 2.2863 | | |

| Sex | Method | Mean | 95% CL Mean | | Std Dev | 95% CL Std Dev | |
|---|---|---|---|---|---|---|---|
| F | | 60.5889 | 56.7315 | 64.4463 | 5.0183 | 3.3897 | 9.6140 |
| M | | 63.9100 | 60.3776 | 67.4424 | 4.9379 | 3.3965 | 9.0147 |
| Diff (1-2) | Pooled | -3.3211 | -8.1447 | 1.5025 | 4.9759 | 3.7339 | 7.4596 |
| Diff (1-2) | Satterthwaite | -3.3211 | -8.1551 | 1.5129 | | | |

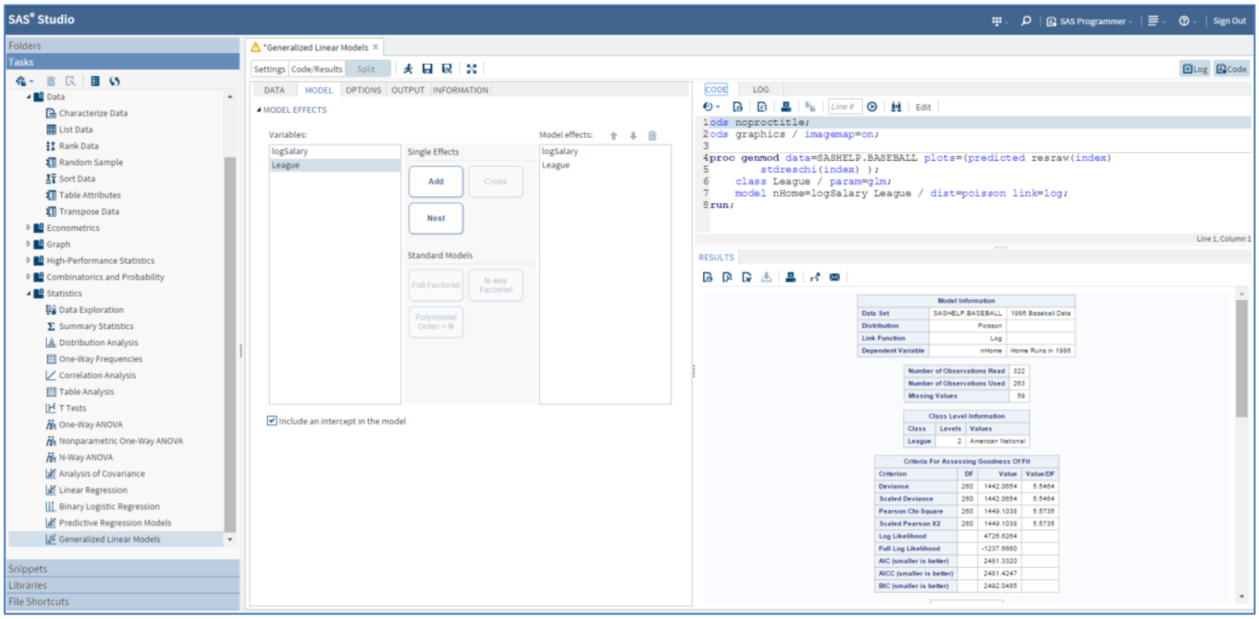
| Method | Variances | DF | t Value | Pr > \|t\| |
|---|---|---|---|---|
| Pooled | Equal | 17 | -1.45 | 0.1645 |
| Satterthwaite | Unequal | 16.727 | -1.45 | 0.1652 |

| Equality of Variances | | | | |
|---|---|---|---|---|
| Method | Num DF | Den DF | F Value | Pr > F |
| Folded F | 8 | 9 | 1.03 | 0.9527 |

Line 17, Column 8

**Figure 5: T Test Task**

The new Generalized Linear Models task enables you to create classic linear models with normal errors, logistic and probit models for binary data, and log-linear models for multinomial data. In this example, we use the SASHELP.BASEBALL data set highlighting the model builder on the Model tab.



Variables: logSalary, League

Model effects: logSalary, League

☑ Include an intercept in the model

```
ods noproctitle;
ods graphics / imagemap=on;

proc genmod data=SASHELP.BASEBALL plots=(predicted resraw(index)
        stdreschi(index) );
    class League / param=glm;
    model nHome=logSalary League / dist=poisson link=log;
run;
```

Line 1, Column 1

| Model Information | | |
|---|---|---|
| Data Set | SASHELP.BASEBALL | 1986 Baseball Data |
| Distribution | Poisson | |
| Link Function | Log | |
| Dependent Variable | nHome | Home Runs in 1986 |

| | |
|---|---|
| Number of Observations Read | 322 |
| Number of Observations Used | 263 |
| Missing Values | 59 |

| Class Level Information | | |
|---|---|---|
| Class | Levels | Values |
| League | 2 | American National |

| Criteria For Assessing Goodness Of Fit | | | |
|---|---|---|---|
| Criterion | DF | Value | Value/DF |
| Deviance | 260 | 1442.0654 | 5.5464 |
| Scaled Deviance | 260 | 1442.0654 | 5.5464 |
| Pearson Chi-Square | 260 | 1449.1038 | 5.5735 |
| Scaled Pearson X2 | 260 | 1449.1038 | 5.5735 |
| Log Likelihood | | 4726.6284 | |
| Full Log Likelihood | | -1237.6660 | |
| AIC (smaller is better) | | 2481.3320 | |
| AICC (smaller is better) | | 2481.4247 | |
| BIC (smaller is better) | | 2492.0485 | |

**Figure 6: Generalized Linear Models Task**

Other new and updated tasks include the following:

- Analysis of Covariance task enables you to augment the analysis of variance model with a quantitative variable in order to reduce the variance of the error terms in the model.
- Bubble Plot task explores the relationship between three or more variables.
- Combinatorics and Probability tasks include the Same Birthday Probability task and the Permutations task.

- N-way ANOVA task performs an N-way analysis of variance.
- High-Performance Generalized Linear Model task has been updated so that you can now specify an offset variable to use as an offset to the linear predictor. The new Dispersion parameter option enables you to specify a fixed dispersion parameter for distributions that have a dispersion parameter.
- One-Way ANOVA task now includes these comparison methods: Dunnett two-tail, Dunnett lower one-tail, and Dunnett upper one-tail.

The Common Task Model continues to be updated with more features, which allows you more flexibility in creating your own tasks. For more information, see *SAS Studio: Developer's Guide*.

## VISUAL PROGRAMMER PERSPECTIVE

Because SAS Studio can be used by a variety of people and groups within an organization, you can choose to view a specific subset of features, or perspectives that best meets your needs. Perspectives are sets of functionality that are customized to meet the needs of different types of users. SAS Studio includes two perspectives: the SAS Programmer perspective and the Visual Programmer perspective. More perspectives are planned.

The SAS Programmer perspective is for users who intend to use SAS Studio mainly for writing and editing SAS programs. By default, the SAS Programmer perspective opens with a new program window and includes all five sections of the navigation pane so that you can easily manage all of your program files and code snippets. The new Visual Programmer perspective is designed for users who want to work with process flows. You can use process flows in the Visual Programmer perspective to combine individual processes into one repeatable process flow that you can save, reuse, and share with other users. By default, the Visual Programmer perspective opens with a new process flow window and includes four sections in the navigation pane: Folders, Tasks, Snippets, and Libraries.

## WHY USE A PROCESS FLOW?

Suppose you have a multi-step program where you have to do a bit of data manipulation before you can create the report you want. Without a lot of good comments (and who has time for those), it’s not easy to tell what is going on.

```
/*Subset WTO commodity codes to high-level data*/
data demo.hscodes;
   length Product_Code $ 2;
   set demo.codes(where=(length(productcode)=2));
   Product_Code=ProductCode;
   KEEP Product_Code Product_Description;
run;
/*Set up US tariff rates to match WTO and Commerce Dept data*/
data demo.ustariffs;
   length Product_Code $ 2;
   Rate=0.0;
   set demo.us(where=(Col__1_Rate <> ''));
   Product_Code=substr(HTS_No_, 1, 2);
   if (index(Col__1_Rate, '%')) then
      do;
         Rate=input(Col__1_Rate, ?? PERCENT20.);
         if (Rate=.) then delete;
      end;
   else if (LEFT(UPCASE(Col__1_Rate))=:'FREE') then rate=0;
        else delete;
   do while(1);
      i=notprint(Description);
      if (i=0) then leave;
      substr(Description, i, 1)='*';
   end;
   keep Product_Code Rate Description;
run;
/*Aggregate US rates*/
proc sql;
   create table demo.tariffs as
      select Product_Code, avg(Rate) as Avg_Rate
      from demo.ustariffs
      group by Product_Code;
quit;
/*Join US rates back to harmonized code*/
proc sql;
   create table demo.summary as
      select tariffs.Avg_Rate, hscodes.Product_Description,
hscodes.Product_Code
      from demo.tariffs tariffs inner join demo.hscodes hscodes
         on (tariffs.Product_Code = hscodes.Product_Code);
quit;
/*Get tariff categories over 6%*/
proc sql;
   create table demo.final as
      select Avg_Rate, Product_Description
         from demo.summary
         where (Avg_Rate > 0.06);
quit;
/*Visualize tariff categories over 6%*/
ods graphics / reset imagemap;
proc sgplot data=demo.final;
   title "Top US Tariff Rates By Commodity Code";
   footnote2 j=l "Source - US Commerce Department";
   hbar Product_Description / response=Avg_Rate stat=Sum;
   xaxis grid;
run;
ods graphics / reset;
title;
footnote;
```

You can use a process flow to break this program up and create a separate node for each piece so that it's easier to see what is going on. To create a process flow, you must use the Visual Programmer perspective. Click the perspectives menu on the toolbar and switch from SAS Programmer to Visual Programmer. In the new process flow that's now open, click + on the toolbar and select **SAS Program**.

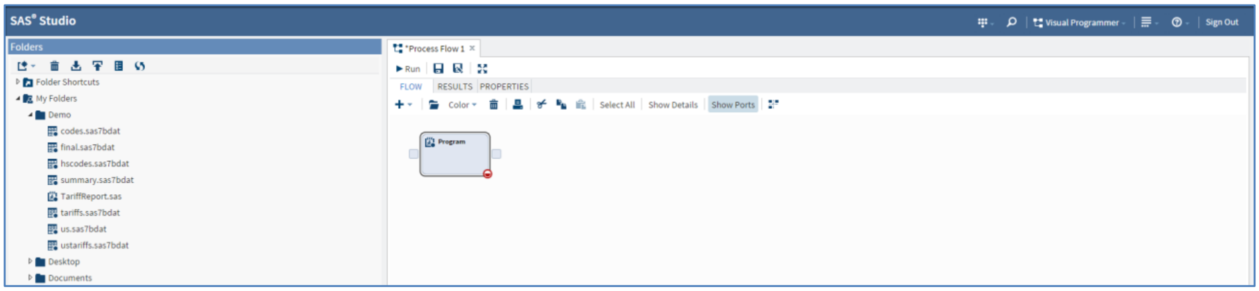
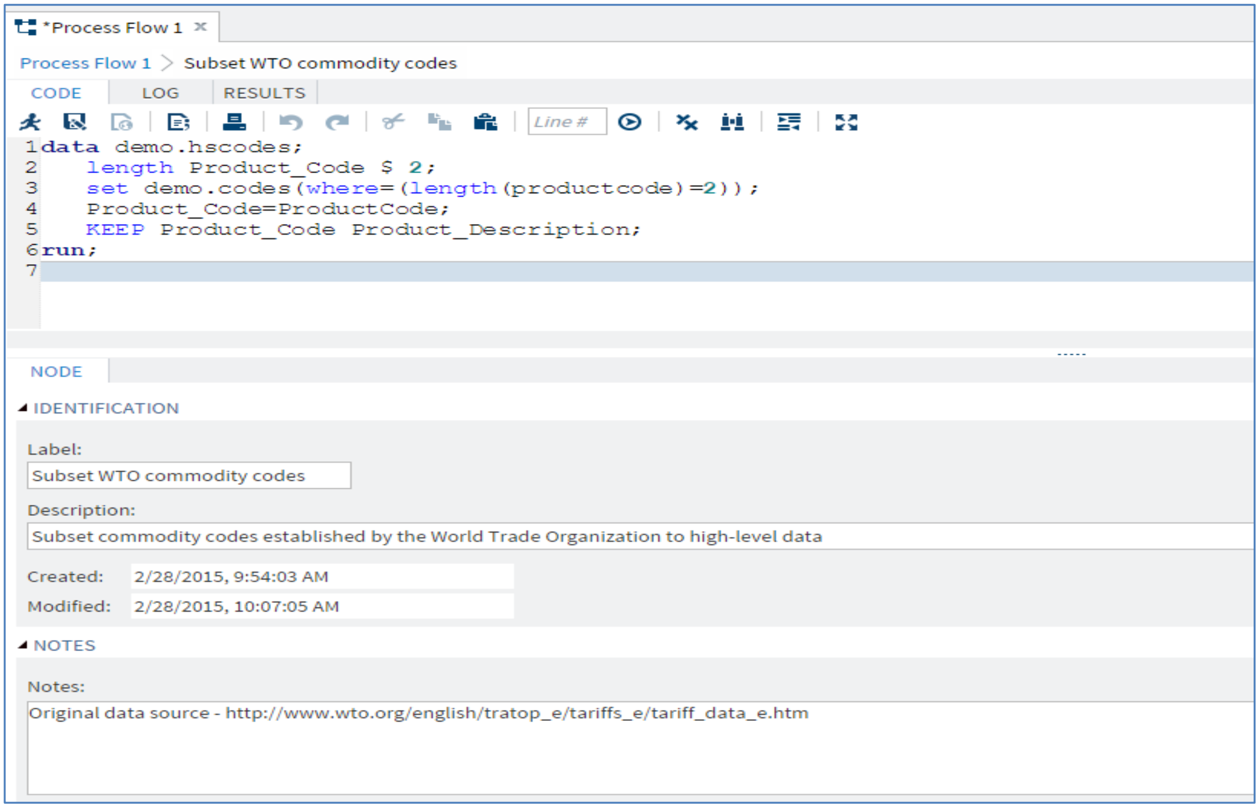


**Figure 7: New Process Flow Containing a SAS Program Node**

Select the node and click 📂 (or double-click the node). The familiar code editor appears and you can paste the first step of your program on the Code tab. There is an extra tab named Node where you can put more information about the program. In this screenshot, the Node tab has been dragged to the bottom of the screen so that you can see both the code and the node information.



```
data demo.hscodes;
   length Product_Code $ 2;
   set demo.codes(where=(length(productcode)=2));
   Product_Code=ProductCode;
   KEEP Product_Code Product_Description;
run;
```

Label: Subset WTO commodity codes

Description: Subset commodity codes established by the World Trade Organization to high-level data

Created: 2/28/2015, 9:54:03 AM

Modified: 2/28/2015, 10:07:05 AM

Notes: Original data source - http://www.wto.org/english/tratop_e/tariffs_e/tariff_data_e.htm

**Figure 8: Creating a SAS Program Node**

Repeat the same process to create a Program node for each piece of your code. Drag the nodes around to show the order they need to run in and link them by drawing an arrow from the control port of one node to the control port of another node. Generally, each node has an input port and an output port. To view the ports for the nodes in your process flow, click **Show Ports**. Once you have all of the steps of the example program set up as separate program nodes, your process flow would look like the following:

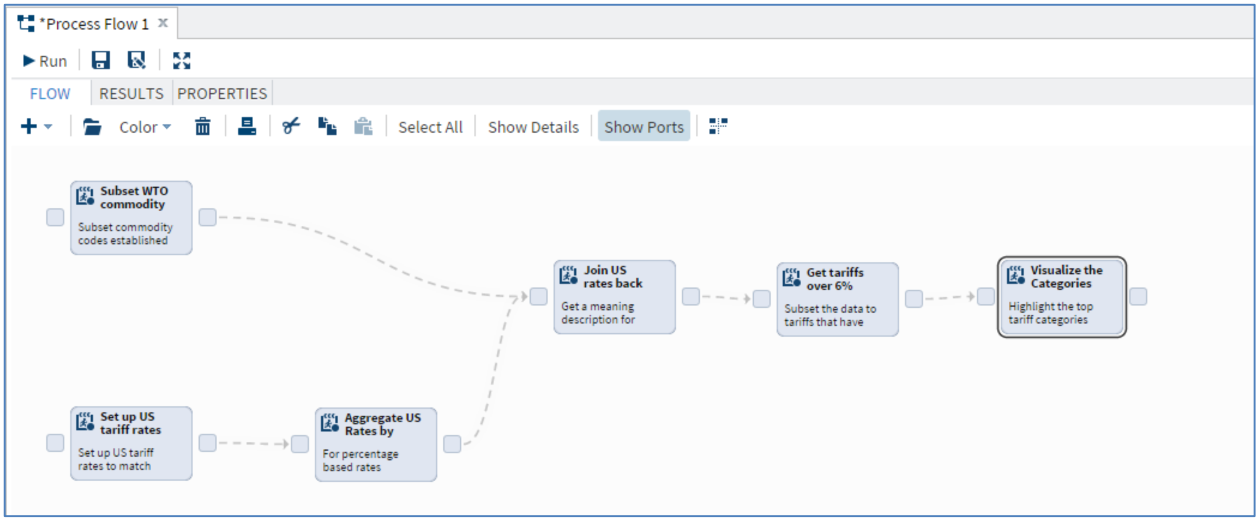


**Figure 9: Final Process Flow Using Program Nodes**

The process flow runs the nodes in the order in which the nodes are added to the process flow. If node 1 is dependent on node 2, node 2 must run completely before node 1 will run. You can also choose to run the nodes in parallel, which means that multiple workspace servers are used to run the nodes; this is done via the Properties tab of the flow. You can run the entire process flow, draw a box around a group of nodes to run only those nodes, or you can run nodes individually. A status indicator on the node lets you know whether a node ran, ran with warnings, or failed to run. You can also use the Results tab to see the status of each node, how long it took to run the node, and whether any output data was created.

You can create a subflow so that you can easily run the nodes within the subflow without running the entire process flow. If you save a process flow, a CPF file is created that you can bring into another SAS Studio session.

## BUILDING A PROCESS FLOW WITH SOME HELP

What if you are not starting from an existing program where you can just paste the code in? You are already familiar with the tools in SAS Studio (for example, tasks and the query tool) that can help you write code. You can use those tools in the Visual Programmer perspective, too, to help you create process flows. Let's take the same example, but get a bit more help.

Create a new process flow by clicking ▦ and select **New Process Flow**. The first two nodes are DATA steps, so let's just leave those as SAS program nodes and copy them from the existing process flow and paste them into the new one.

The Aggregate US Rates by Commodity node is a PROC SQL step, so you can use the Query Builder to set this up. Click **Add**, and then select **Query**. A node for the query is added to the process flow. Open the node and the Query Builder appears. You can complete the same steps as in the earlier basic query example by using the Query Builder to create the SQL code needed for this node. As with the Program node, you can use the Node tab to add more descriptive information.

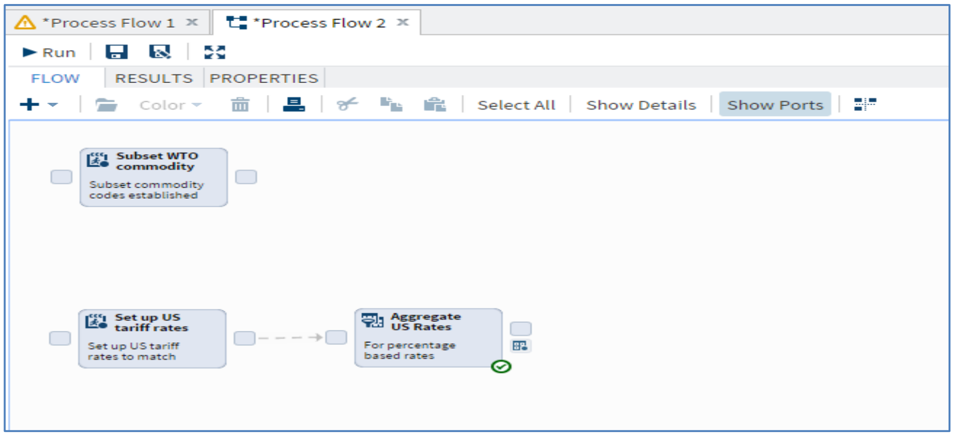


**Figure 10: Building a Flow Using the Query Builder**

Once the query node is added and has been run, you notice that there is an extra port on the flow for this node. Earlier we used the control ports to connect nodes together in a flow. Those types of ports appear as □. This new port is a data port and appears as ▦. You can use data ports to document inputs and outputs of each node if you want. In the case of a query node, a statement such as %WEB_OPEN_TABLE(*table_name*); is added by the Query Builder to the code and the process flow uses that to populate the data port.

The next two nodes are also queries. One is a join similar to the one shown earlier, and the other is another subset so that you can add those nodes in a similar fashion by adding query nodes. The final node generates a bar chart and you can get help doing that from the Bar Chart graph task. In the navigation pane, click the **Tasks** section, navigate to the task you want, and then drag it into the process flow. Then you can use the task interface to generate the code you need; in this case, to generate the PROC SGPLOT to create the final output.

When you are done, the process flow pretty much looks like the one you created earlier. But when you open a query or task node, the interface helps you with your node rather than the editor alone. As before, when you run the flow, you see the status indicators and can drill into the final node to see the resulting graph.

## MORE NEW THINGS YOU WANT TO KNOW ABOUT

The new SAS variable name policy option enables you to specify the set of rules that you want to use for SAS variable names. The Default text encoding option specifies the character-set encoding that is used when text files are read or written. You can find these two options in the Preferences window.

There are two experimental features available with SAS Studio 3.3, but to access them requires an administrator to turn them on via the configuration properties file. Once configured, both of these experimental features can be turned on in the Preferences window. One of the features is the ability to run a SAS stored process as a SAS Studio task. This feature is only available in the Enterprise edition of SAS Studio and also includes a new SAS Folders navigation pane to access the stored processes in your metadata. The other experimental feature is the ability to share a SAS Studio task via a URL. You save your task as a CTM or CTK file in an accessible location, and then click the Share Task icon, which generates a URL you can send to other authorized users to run your task.

## CONCLUSION

SAS Studio continues to focus on the needs of SAS programmers. It's a comfortable environment for those used to the traditional SAS windowing environment (SAS® Display Manager), but new features such as the query tool, process flow diagrams, and tasks have been added to appeal to traditional SAS® Enterprise Guide® users and others who would prefer a bit of help when coding. SAS Studio will continue to be enhanced with both programming and point-and-click features so that users can pick and choose to create a SAS development environment perfect for their needs.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Mike Porter
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Mike.Porter@sas.com
http://www.sas.com

Amy Peters
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Amy.Peters@sas.com
http://www.sas.com

Michael Monaco
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Michael.Monaco@sas.com
http://www.sas.com