SAS® Studio: User's Guide

2020.1.5*

* This document might apply to additional versions of the software. Open this document in SAS Help Center and click on the version in the banner to see all available versions.

SAS® Documentation
April 19, 2021

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2021. *SAS® Studio: User's Guide*. Cary, NC: SAS Institute Inc.

SAS® Studio: User's Guide

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

April 2021

v_005-P1:webeditorug

Contents

1

Introduction to SAS Studio

About SAS Studio

SAS Studio is a development application for SAS that you access through your web browser. With SAS Studio, you can access your data files, libraries, and existing programs, and you can write new programs. You can also organize your work in flows and use the predefined tasks in SAS Studio to generate SAS code. When you run a program or task, SAS Studio connects to a SAS server to process the SAS code. The SAS server can be a hosted server in a cloud environment or a remote server in your local environment. After the code is processed, the results are returned to SAS Studio in your browser.

SAS Studio supports multiple web browsers, such as Apple Safari, Google Chrome, Microsoft Edge, Microsoft Edge on Chromium, and Mozilla Firefox. For more information, see "Web Browsers" in *SAS Viya for Containers: Deployment Guide*.

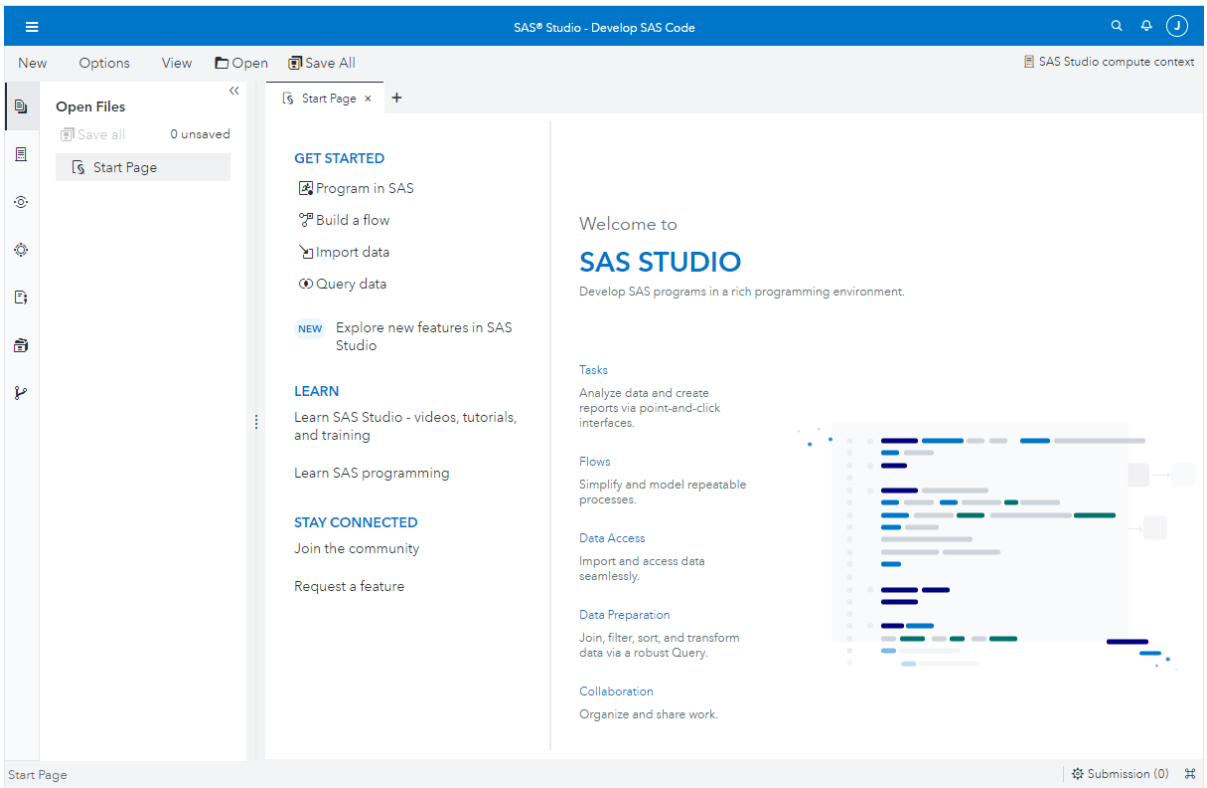
In addition to writing and running your own SAS programs, you can use the predefined tasks that are included with SAS Studio to analyze your data. The tasks are based on SAS procedures and provide access to some of the most commonly used graph and analytical procedures. You can also use the default task template to write your own tasks.

SAS Studio includes two different perspectives: the Standard perspective and the Interactive perspective. A *perspective* is a predetermined set of features that is customized to meet the needs of a specific user type. By selecting a specific perspective, you can narrow the choices that are available in the interface and focus on the features that you need to use regularly. By default, when you open SAS Studio, the Standard perspective is selected. After you open SAS Studio, you can change the perspective by using the **Options** menu on the toolbar. For more information, see "Understanding Perspectives" on page 25.

Understanding the SAS Studio Interface

When you sign in to SAS Studio, the Start Page is available, so you can quickly get started writing a new SAS program, building a flow, importing data, or creating a query. Use the icons to access the different sections of the navigation pane.

Note: To sign out of SAS Studio, click the button with the first initial of your user name on the far right of the application toolbar and select **Sign out**. Do not use the Back button on your web browser.



At the top, the application bar enables you to access other SAS applications. Click the search icon to search for items across SAS applications.

The main window of SAS Studio consists of a navigation pane on the left and a work area on the right. The navigation pane provides easy access to your open files, your folder shortcuts, file system and SAS content, steps, your tasks and snippets, the libraries that you have access to, your Git repositories, and your file references. The **Open Files** section is displayed by default. The **File References** section is not displayed by default, but can be opened by selecting **View** ⇨ **Navigation panes** ⇨ **File References**. For more information, see “Using File References” on page 11.

The work area is used to display your data, code, tasks, logs, results, and flows. As you open these items, they are added to the work area as windows in a tabbed interface. When you first open SAS Studio, the **Start Page** tab is displayed in the work area by default. For more information, see “Using the Work Area” on page 13.

Using the Navigation Pane

About Using the Navigation Pane

You can expand the sections of the navigation pane by clicking the section that you want to view. To close and reopen the navigation pane, click a section button twice.

Accessing Your Open Files

The **Open Files** section of the navigation pane enables you to quickly view and access all of the files that you have opened in your current SAS Studio session. To access a file from the list of open files, click the file that you want to view. You can save changes to all files in the list by clicking **Save all**.

You can also use the **Open Files** section to add an open program to the active flow by right-clicking the program file and selecting **Add to flow**. This option is available only for programs and only if there is an active flow in the work area.

Using the Explorer

The **Explorer** section in the navigation pane enables you to access files and folders from your folder shortcuts, your server file system, and your SAS Content Server locations. For users of SAS Studio 2020.1 and later, the default file storage location is SAS Content. SAS Content locations are integrated with other SAS Viya applications, such as SAS Drive, so that you can seamlessly share files among the applications and with other SAS users. For more information, see *SAS Drive: Getting Started*.

If you also need access to a file system to perform other tasks, such as working with a Git repository, your Kubernetes and SAS Studio administrators can create access to the file system during deployment. In SAS Studio 5.x, by default, you had access to both SAS Content and file system storage locations. For more information, see "Creating Persistent File Storage" in *SAS Studio: Administrator's Guide*.

Note: The ability to access files on a SAS Content Server is available only in SAS Studio 5.1 or later. For more information, contact your site administrator.

The **Explorer** section includes a My Favorites folder that you can use to quickly access the files that you add to it. You can add many types of files to your My Favorites folder, including SAS data sets, other data files, programs, flows, snippets, tasks, and queries.

- To add a file to your My Favorites folder, right-click the file in the appropriate section of the navigation pane and select **Add to My Favorites**.
- To remove a file from your My Favorites folder, right-click the file in the appropriate section of the navigation pane and select **Remove from My Favorites**. You can also select the file in the My Favorites folder and click .

You can use the **Explorer** section to create folders and folder shortcuts as well as download and upload files. From the folders tree, you can expand and collapse folders, copy and move items, and open items in folders by double-clicking them or dragging them to the work area. In addition, you can open files from your folders and folder shortcuts by clicking **Open** on the SAS Studio toolbar.

Note: If you want to drag an item to another location in the **Explorer** section when there is a scroll bar, drag the item to the upper right or lower right corner of the pane to scroll upward or downward.

You can change the files that are displayed in the **Explorer** section in the following ways:

- To display only SAS program files, click and select **Show .sas files**.
- To display only SAS log files, click and select **Show .log files**.

You can view a file as text by right-clicking the file and selecting **View file as text**.

To open a file as a SAS program, right-click the file and select **Open as program**.

To create a folder shortcut:

1. Click in the navigation pane to open the **Explorer** section. Next, click and select **Folder shortcut**.
2. In the **Name** box, enter the name of the folder shortcut.
3. Click to select a folder. To create a folder, click .
4. Click **OK** to create the folder shortcut. The new shortcut is added to the list of folder shortcuts.

Note: You can rename a folder shortcut by selecting the folder shortcut in the **Explorer** section and clicking . In the Shortcut Properties window, enter a new name in the **Name** box.

To create a folder, select the folder in the **Explorer** section in which you want to create the folder. Click and select **Folder**. Enter the name of the new folder. The new folder is added to the list of folders.

To download one or more files, right-click the files that you want to download and select **Download file**.

Note: You can download a flow only from a SAS Compute Server folder. You cannot download a flow from SAS Content.

To upload one or more files from your local computer, select the folder to which you want to upload the files and click . Click + to browse for the files that you want to upload. The maximum size of each uploaded file cannot exceed 100 MB.

Note: You can sort the files that are listed in the **Explorer** section by right-clicking the folder that you want to sort and selecting **Sort by**. You can sort the files by name, modified date, or size or choose no sort order. By default, the files are sorted in ascending order by file name.

Working with Steps

The **Steps** section of the navigation pane enables you to add different types of nodes as placeholders in a flow. For example, if you know that your flow is going to contain a query that generates an output table that you want to use as input in a program, you could start by adding a Query node, a Table node, and a SAS Program node to your flow. After you have defined the steps in the flow, you can use the node properties for each node to specify the attributes and content of the node.

For more information, see “What Is a Flow?” on page 102 .

Note: You can also add existing files, such as SAS programs, SAS data sets, snippets, and external files that are valid as input to an Import node, to a flow by dragging them from different sections of the navigation pane.

Working with Tasks

The **Tasks** section of the navigation pane enables you to access tasks in SAS Studio. Tasks are based on SAS procedures and generate SAS code and formatted results for you. SAS Studio is shipped with many predefined tasks that you can run. You can also edit a copy of most of these predefined tasks, and you can create your own new tasks.

To create a task, click  and select **Task**. SAS Studio creates a template in the work area that you can use to create custom tasks for your site. Custom tasks can be accessed from the My Tasks folder or from the **Explorer** section of the navigation pane. For more information, see “Create a Custom Task” on page 255.

To edit a task that you have created, right-click the task in the My Tasks folder and select **Edit task template**. The XML code that is used to create the task is opened in the work area. If you want to edit a predefined task, you must first right-click the task and select **Copy as Task Template** and then select either **My Tasks** or **Explorer**. For more information, see “Edit a Predefined Task” on page 255.

Working with Snippets

The **Snippets** section in the navigation pane enables you to access your saved snippets. Snippets are lines of commonly used code or text that you can save and reuse. Snippets can contain SAS code, XML code, and text. You can insert snippets in locations including the Program Editor, XML editor, text editor, job definitions, job

forms, and job prompts. To insert a snippet, right-click the location where you want to insert the snippet and select **Insert snippet**.

SAS Studio is shipped with predefined code snippets that you can use. You can also edit a copy of these snippets and create your own custom snippets. Your custom snippets can be accessed from the My Snippets folder. For more information, see Chapter 2, “Working with Programs,” on page 31.

To edit a snippet that you have created, right-click the snippet and select **Edit**. If you want to edit a predefined snippet, you must first right-click the snippet and select **Copy snippet to**, and then select either **My Snippets** or **Explorer**.

Note: You can edit only the snippets that are in the `My Snippets` folder or a folder that you can access from the **Explorer** section.

Working with Libraries

The **Libraries** section of the navigation pane enables you to access your SAS libraries. SAS tables are stored in SAS libraries.

Note: Table names that include { } (braces) appear as blank. Do not use { } in table names.

From the **Libraries** section, you can open SAS tables and add them to your programs. You can use the **Libraries** section to expand a table and view the columns in that table. The icon in front of the column name indicates the type.

Note: Caslib names cannot be longer than 256 characters. SAS librefs are limited to eight characters.

Here are examples of common icons for the column types.

Icon	Type of Column
	Character
	Numeric
	Date
	Datetime

You can drag tables and columns from the **Libraries** section to a program, and SAS Studio adds the names of the dragged items to your program. The cursor changes as you drag an item to indicate when you can drop the item to insert it in the program. If the item name is a reserved word in a database, SAS Studio creates a name literal by automatically adding quotation marks and a lowercase letter n to the

item name.For more information, see “Opening and Creating Programs” on page 32.

You can also drag tables from the **Libraries** and **Explorer** sections to other locations in the work area to open them.

Note: The Sasuser library is read-only, as in any SAS server environment. You cannot save content to this library.

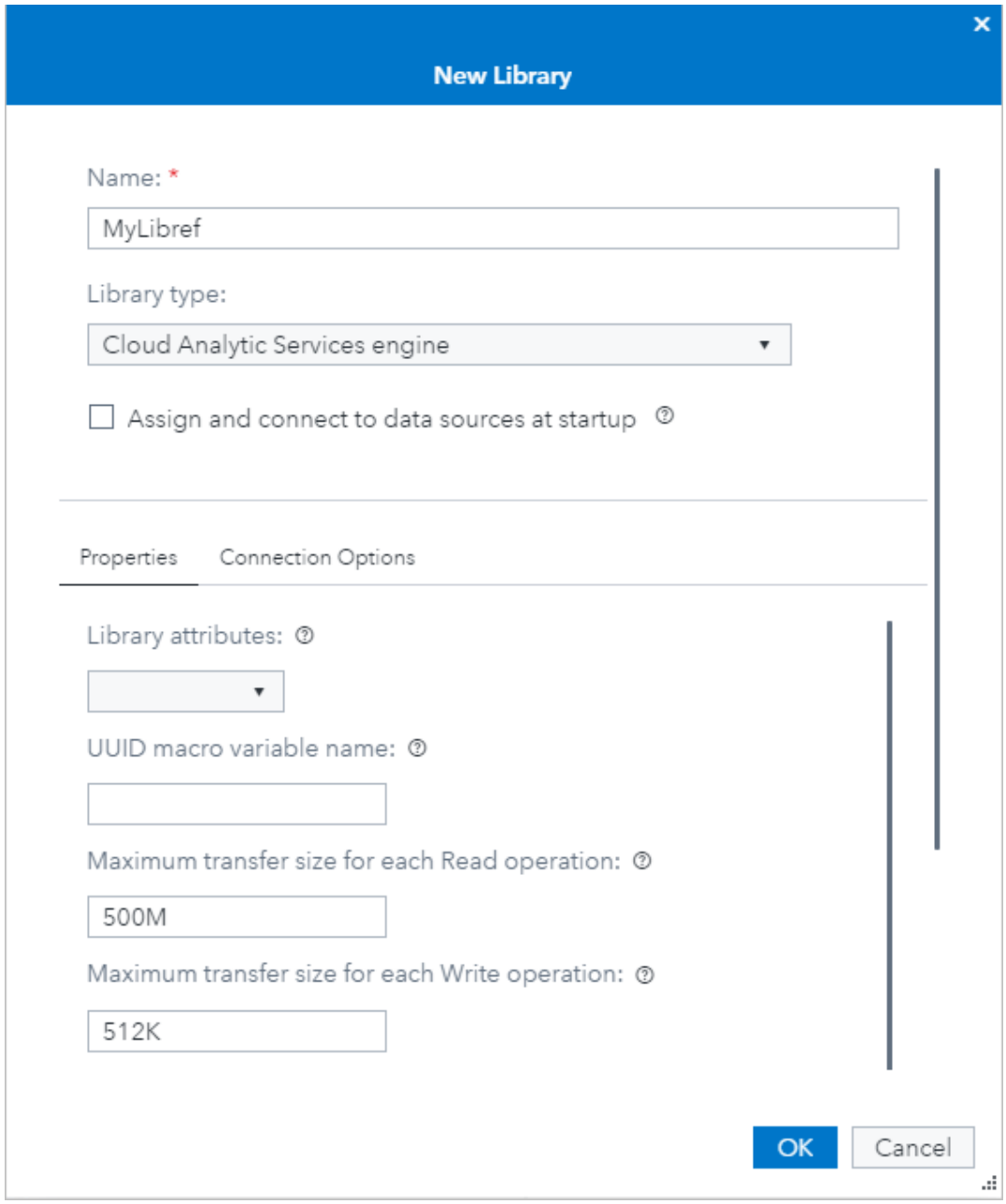
You can refresh all of the libraries and tables in the Libraries pane by selecting > **Refresh**. To refresh an individual table or library, right-click the table or library and select **Refresh**.

To view the properties of a library, select the library in the **Libraries** section, and click .

You can also create new libraries and assign existing libraries.

To create a library:

1 Click in the navigation pane, and then click .



2 In the **Name** box, enter the libref for the library. The libref must be eight characters or fewer.

3 In the **Library Type** box, select the SAS library engine that you want to use. The library engines that are available depend on your deployment.

Note: To create a library using the specified library type, you must have both the library engine and the appropriate client software installed.

For information about using the SAS Base Engine library type, see SAS V9 LIBNAME Engine: Reference.

4 If you want to access this library each time you use SAS Studio, select **Assign and connect to data sources at startup** to add the LIBNAME statement to your autoexec.sas file.

5 If you have administrator privileges and want to make this library available to all users of the current compute context, select **Make data sources available to all users**.

Note: This option is available only if you are authorized to update the current compute context.

6 Use the **Properties** and **Connection Options** tabs to specify the configuration and connection options that you need. For help with the specific options, see the documentation for your library engine.

7 Click **OK** to create the library. The new library is added to the list of libraries in the navigation pane.

TIP You can edit a library that you have created by right-clicking the library in the **Libraries** section of the navigation pane and selecting **Edit library**.

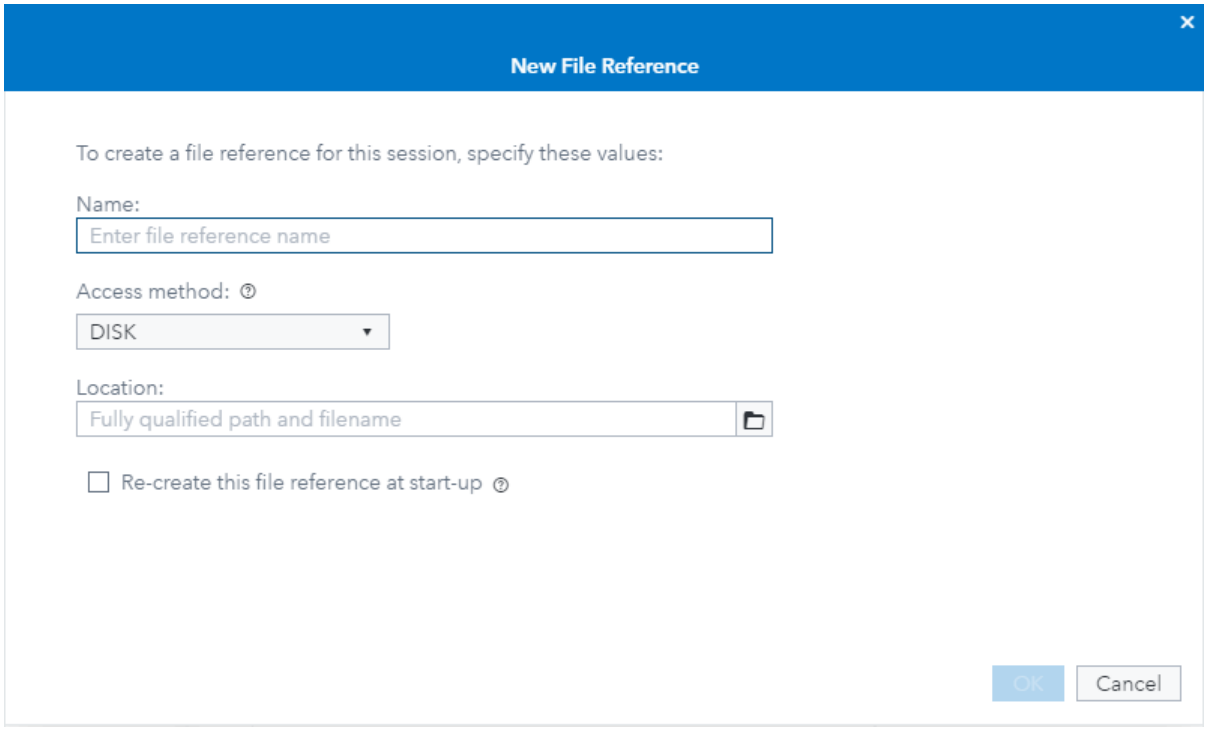
Using File References

File references enable you to quickly access files that you specify. You can create a file reference to a file on your SAS file system, a file from your SAS Content folder, or a file that you access via a URL.

Note: You can also use the FILENAME statement to create a file reference to a file in a remote location that is accessible by your SAS server.

To create a file reference to a file on your SAS file system, a file from your SAS Content folder, or a file that you access via a URL:

1 Click in the navigation pane, and then click .



Note: If the File References button is not available in the navigation pane, select **View** ⇨ **Navigation panes** ⇨ **File References**.

2 In the **Name** box, enter the file reference name.

3 Use the **Access method** box to specify how to access the file:

- **DISK** - accesses the file from your SAS file system. In the **Location** box, browse to find the file that you want to use.
- **FILESRVC** - accesses the file from your SAS Content folder. In the **Location** box, browse to find the file that you want to use.
- **URL** - accesses the file by using the specified URL. In the **URL** box, enter the URL for the file.

4 If you want this reference to be available the next time you use SAS Studio, select **Re-create this file reference at start-up**.

5 Click **OK** to create the file reference. The new file reference is added to the list of file references in the navigation pane.

You can also create a file reference to a file on your SAS file system (DISK), a file from your SAS Content folder (FILESRV), or a file that you access via a URL (URL) programmatically. The file reference is displayed in the File References pane. You can open a file or URL from a file reference by double-clicking it or dragging it to the work area.

Here is some sample code:

```
FILENAME MyProg FILESRVC folderpath='/Users/userid/'
filename="MyProgram.sas';
FILENAME MyURL URL 'http://www.sas.com';
FILENAME SASPROG '/sastest/General_Test/MyProgram.sas';
```

Working with Git Repositories

The **Git Repositories** section in the navigation pane enables you to access Git features from within SAS Studio. These features include cloning repositories, committing and stashing file changes, pulling and pushing files, viewing your repository history, creating and merging branches, rebasing a branch, and performing a basic differentiation between files in your local repository. For more information, see "About Git Integration in SAS Studio" on page 220.

Customizing the Navigation Pane

To customize which sections of the navigation pane are displayed, click **View** and select **Navigation panes**. Select or clear any sections that you want to add or remove. The navigation pane is updated immediately.

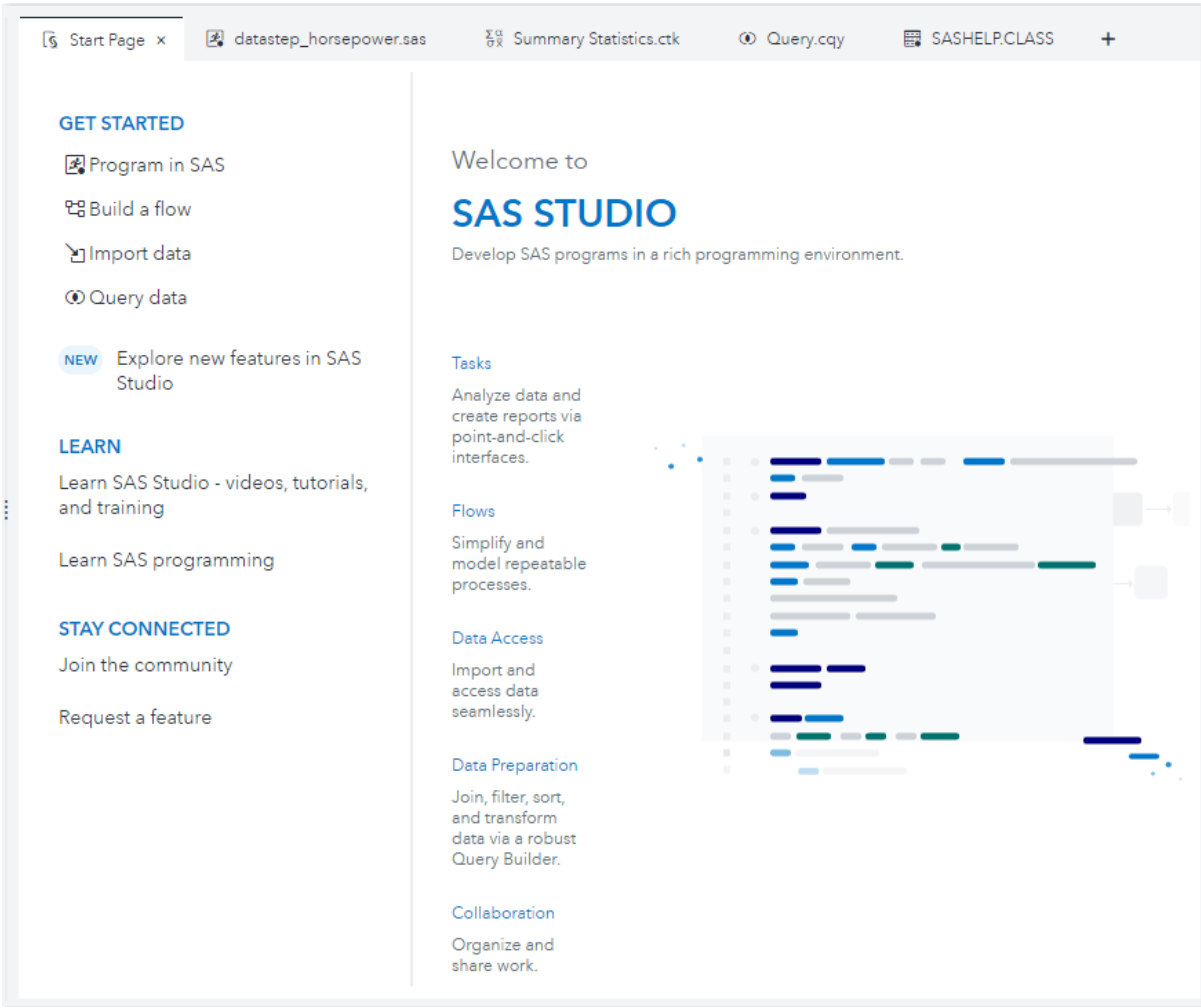
Note: The **File References** section is not displayed by default.

Using the Work Area

About Using the Work Area

The work area is the main portion of the SAS Studio application for accessing programs, tasks, and flows and for viewing data. The work area is always displayed and cannot be closed. When you open a program, task, table, or flow, the windows open as new tabs in the work area. The code, log, and results that are associated with programs and tasks are grouped together under the main tab for the program or task.

When you first open SAS Studio, the **Start Page** tab is displayed in the work area by default. The Start Page provides quick access to a few basic tasks (for example, creating a program or query) and links to learning resources. If you close the **Start Page** tab, you can re-open it by selecting **View** ⇨ **Start Page**. You can use the **Show Start Page** option to control whether to display the **Start Page** tab each time you start SAS Studio. For more information, see "Setting the Start-Up Preferences" on page 260.



Customizing the Work Area

By default, the work area is displayed beside the navigation pane, but you can maximize the work area and hide the navigation pane. You can also close all of the tabs in the work area at once.

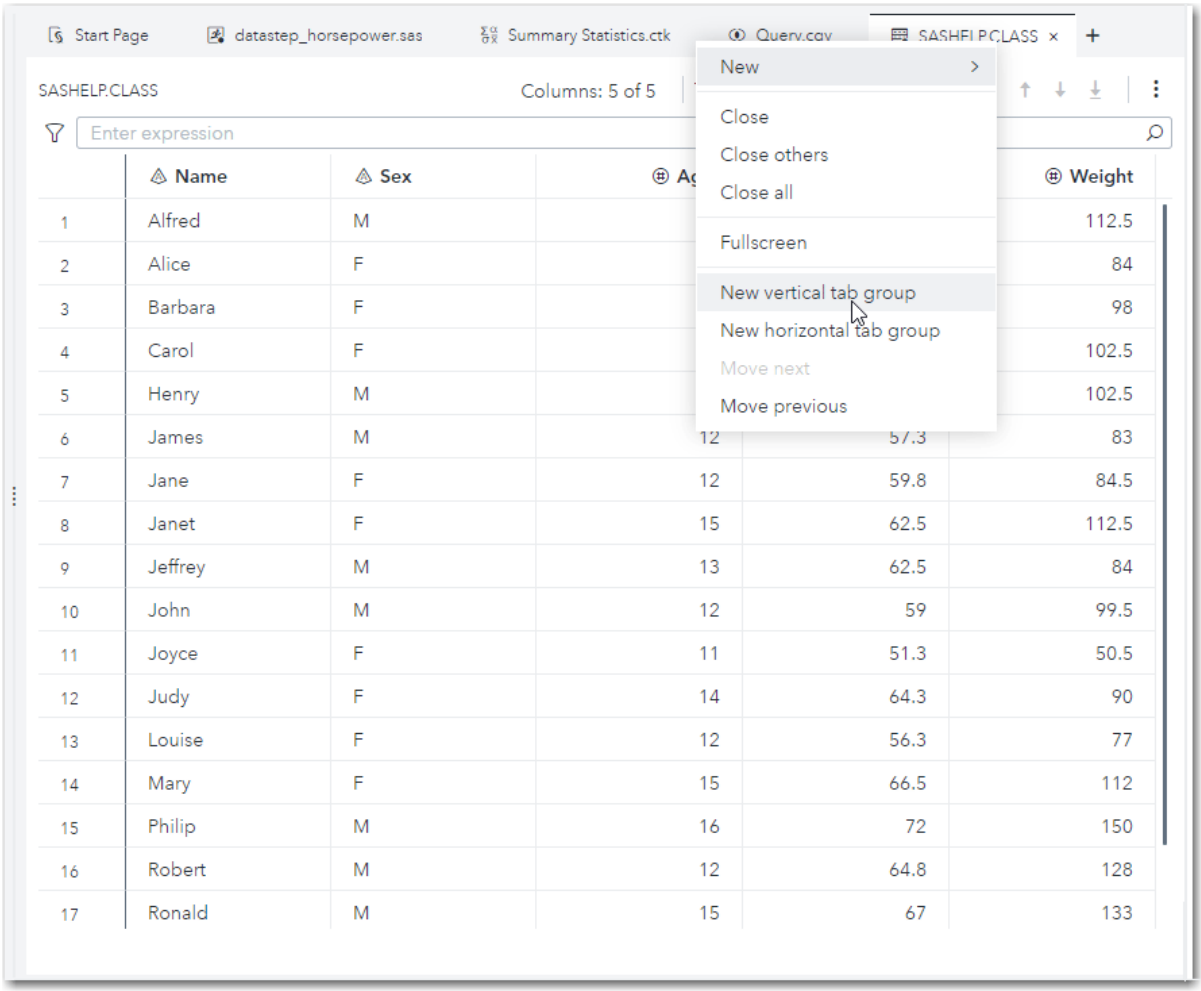
To maximize the work area, select **View** ⇨ **Fullscreen**.

Note: To reopen the navigation pane, click **Restore** to the right of the tabs. You can also double-click any tab in the work area or right-click a tab and select **Restore**.

You can also close some or all of the open tabs in the work area. To close all tabs that are open in the work area, right-click any tab and select **Close all**. To close all tabs except the active tab, right-click the active tab and select **Close others**. You are prompted to save any unsaved programs or tasks.

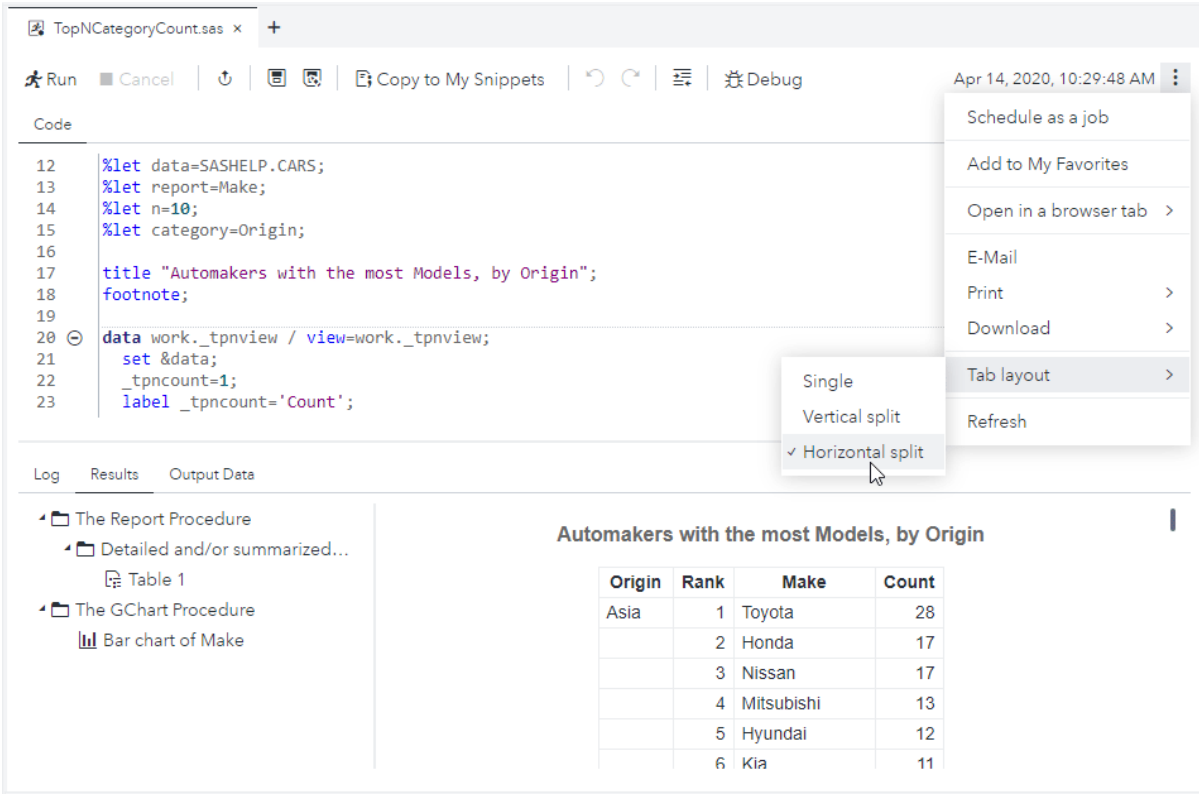
Rearranging the Tabs in the Work Area

In the work area, you can rearrange the tabs by dragging them to the left or right. You can also create separate vertical and horizontal groups and move tabs between the groups. To rearrange a tab, right-click the tab that you want to move and select the appropriate option.



In the Standard perspective, you can change the layout of a program tab by displaying the **Log**, **Results**, and **Output Data** subtabs in a single group with the **Code** subtab split vertically or horizontally. To change the layout, click ⋮ and **Tab layout**. Then, select the appropriate option.

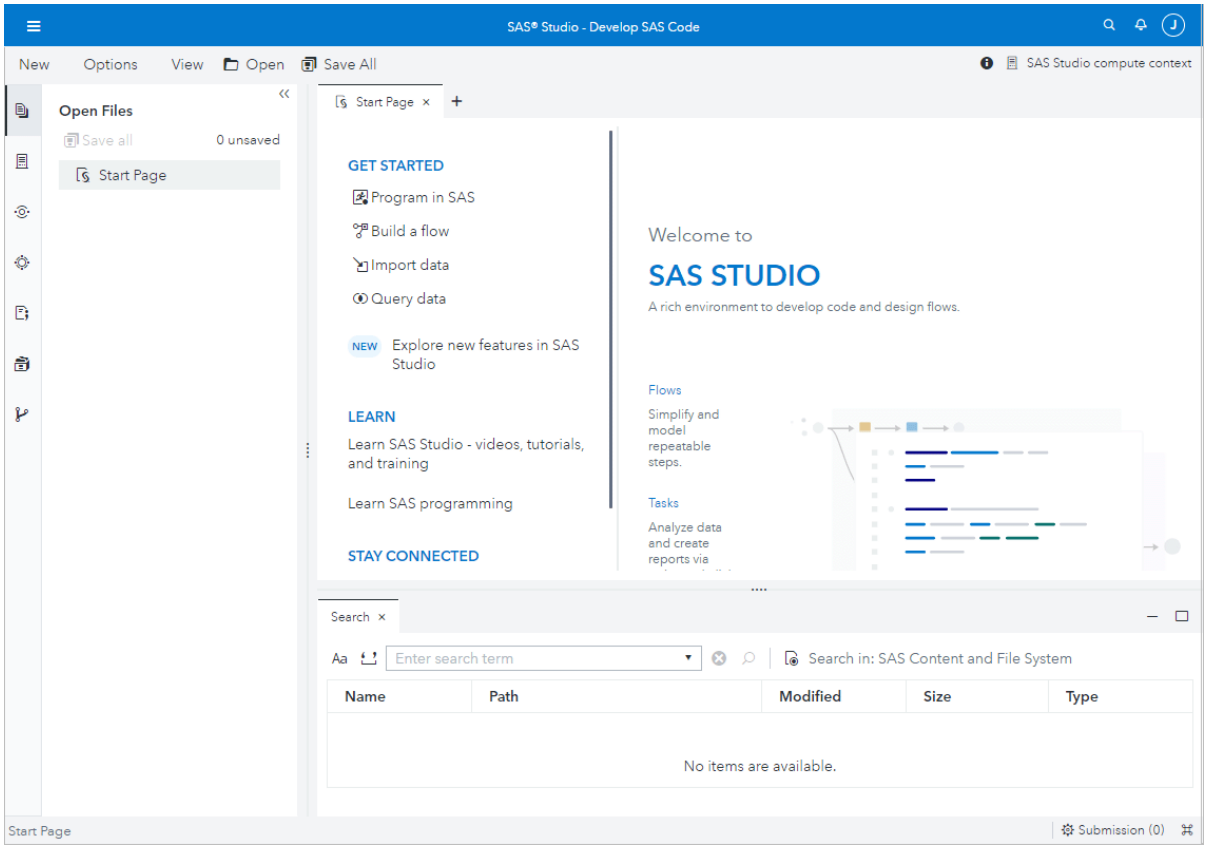
In the Interactive perspective, you can display the **Log**, **Results**, and **Output Data** subtabs as a separate vertical or horizontal tab group. To change the layout, click ⋮ and select the appropriate option.



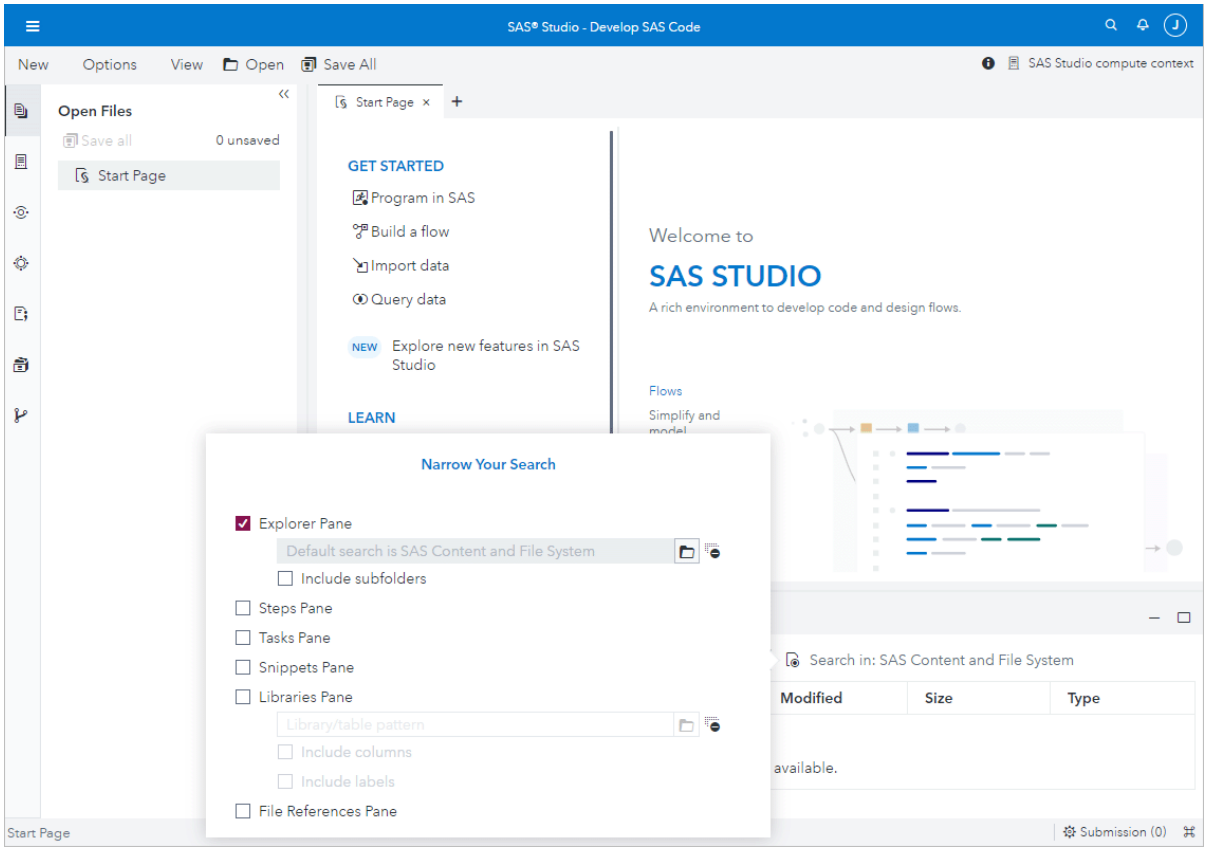
Searching in SAS Studio

You can use the Search feature to search all of the sections of the navigation pane. You can specify which sections and which types of items you want to include in your search. You can also specify whether you want the search to be case-sensitive and match whole words only.

To access the Search feature, select **View** ⇨ **Search**.

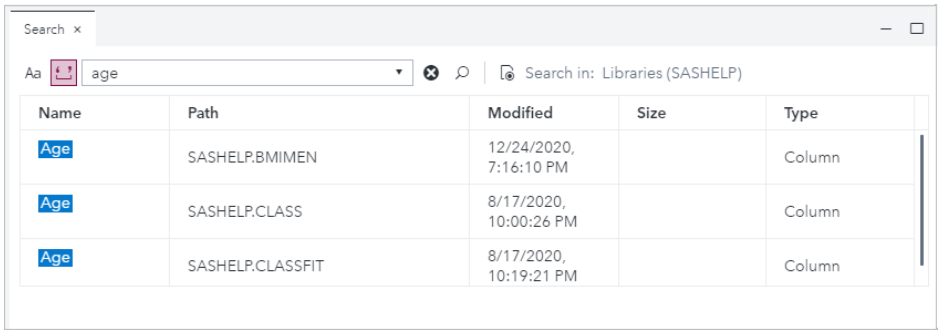


To search for an item, enter the text that you want to search for in the search term box. To specify the areas of the navigation pane to search, click . By default, SAS Studio searches your SAS Content and file system folders in the **Explorer** section of the navigation pane.



To make your search case-sensitive, click Aa. To match the whole word in your search, click .

When you have specified your search criteria, click . You can open items in the search results by double-clicking them.

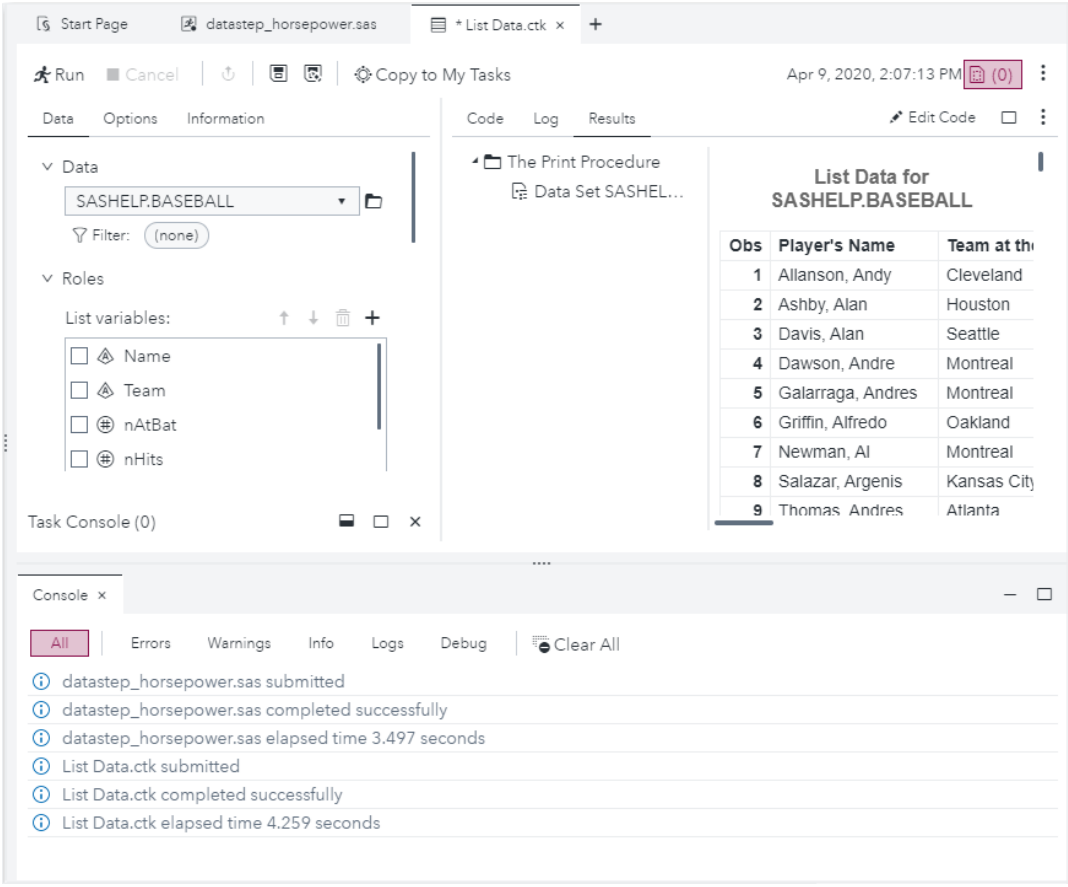


Using the Console

The Console window displays a list of all of the messages, such as error messages and warnings, that are generated as you run programs and tasks. To open the console, select **View** ⇨ **Console**. If you minimize the console tab, you can click in

the lower right corner of the work area to restore the console or ☐ to maximize the console.

You can use the buttons at the top of the window to filter the types of messages that are displayed. By default, all types of messages are displayed. The messages are cleared by default when you sign out of SAS Studio. You can manually clear the messages from the console by clicking **Clear All**.



Using the Background Submit Feature

About the Background Submit Feature

You can run a saved SAS program, query, task, or flow as a background submission, which means that the file can run while you continue to use SAS Studio. You can view the status of files that have been submitted in the background, and you can cancel files that are currently running in the background.

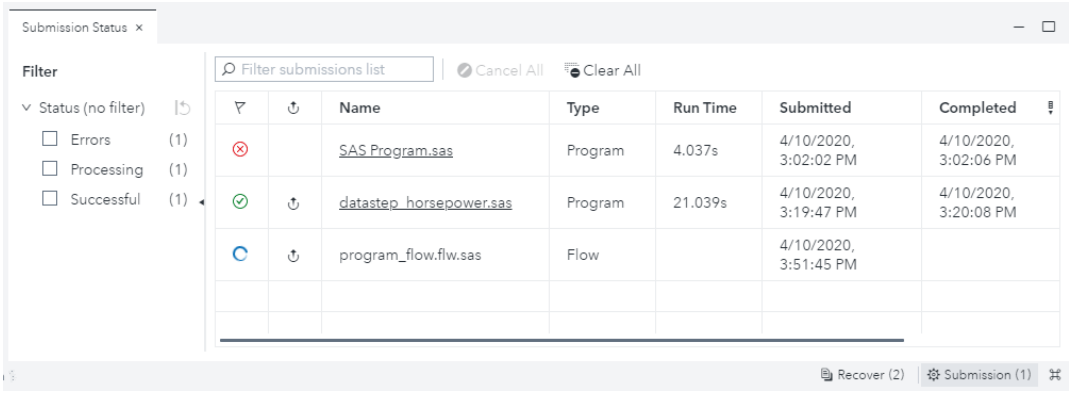
Running a File as a Background Submission

To run a file as a background submission, right-click the file in the navigation pane and select **Background submit** or click in the toolbar. Before the file is run, the background process changes the current working directory to the directory in which the program is located.

By default, a notification message is displayed when the file is submitted and again when the file has finished running. If you log off from SAS Studio while the file is running, the file continues to run, but the notification message that indicates when the file is finished is not displayed.

Note: Because a background submission uses a separate compute server, any libraries or tables that are created by the submission do not appear in the **Libraries** section of the navigation pane in SAS Studio.

To view the status and properties of your background submissions, click **View** and select **Submission Status**. Background submissions are indicated by . For more information, see "Using the Submission Status Window" on page 21.



Customizing Your Background Submissions

The Preferences window enables you to customize how to handle background submissions.

To change whether existing log and output files are deleted or overwritten when you rerun a background submission, select **Options** ⇨ **Preferences**. Click **Background Submit**.

For more information about each option, see "Setting Background Submit Preferences" on page 271.

Using the Submission Status Window

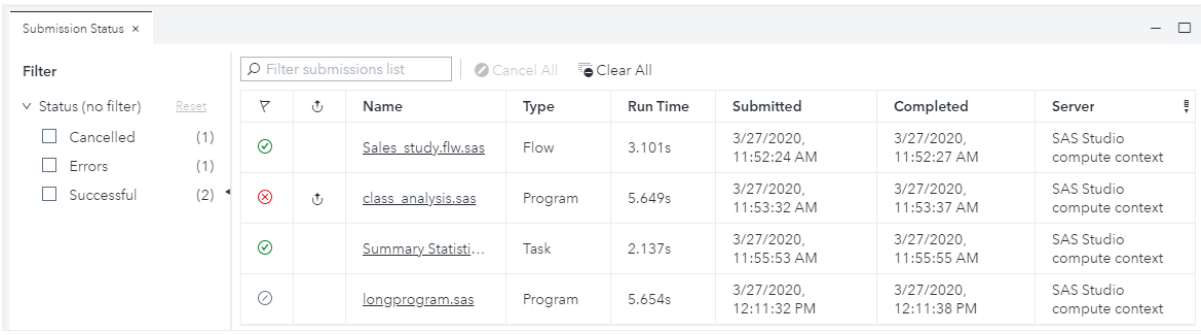
The Submission Status window displays information about the programs, tasks, queries, and flows that you run in SAS Studio. To open the Submission Status window, select **View** ⇨ **Submission Status** or click **Submission** in the lower right corner of your SAS Studio browser window.

To cancel a submission that is currently running, right-click the item and select **Cancel**. To cancel all programs, tasks, queries, and flows that are currently running, click **Cancel All** on the Submission Status window toolbar.

Note: You cannot cancel background submissions.

The Submission Status window includes an entry for each time you have run a program, task, query, or flow. You can use the Submission Status window to return to an earlier version of a program, task, query, or flow. When you click an item in the Submission Status window, you open the version of the item that was run at that time and the associated log. The program or automatically generated task, query, or flow code opens in a new tab from which you can make and save changes.

Note: The code that you open from the Submission Status window is a copy of the code that SAS Studio ran and is no longer associated with the original program, task, query, or flow. Editing this code does not affect the original program, task, query, or flow.



You can customize the items that are displayed in the Submission Status window in several ways:

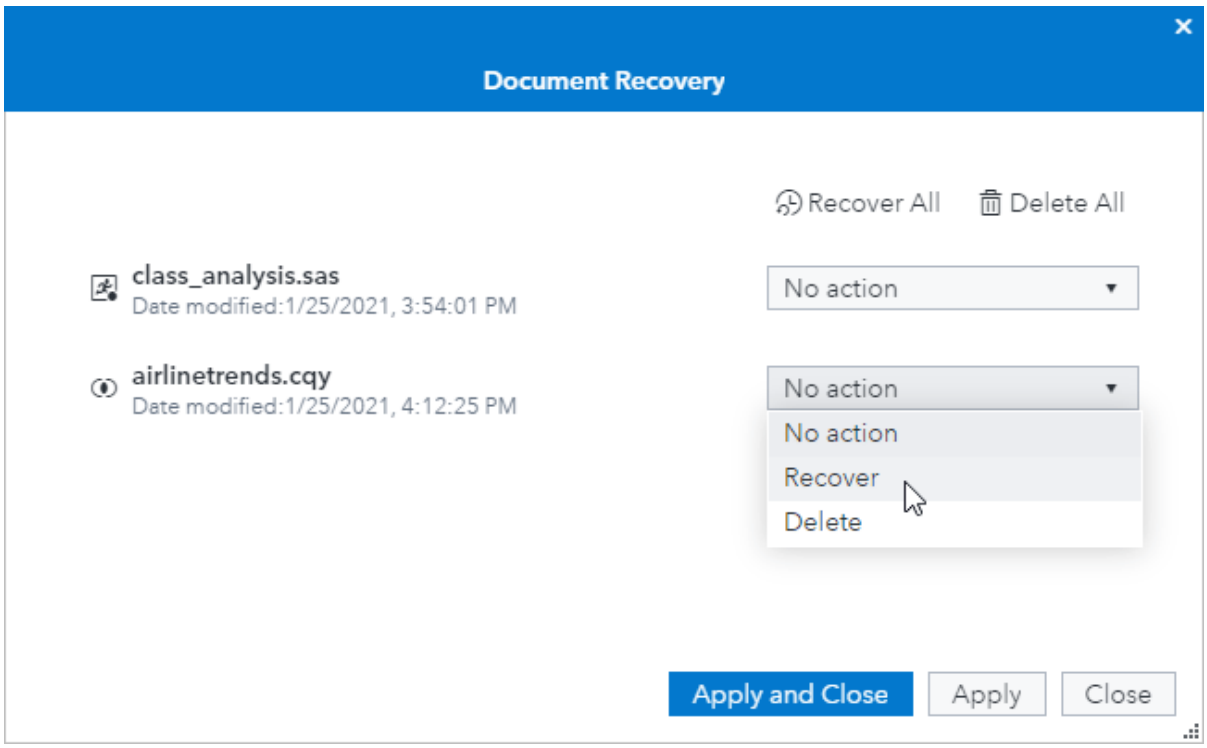
- Use the Filter pane to subset the displayed items by submission status.
- Use the **Filter submissions list** box to specify filter criteria for the Name column.
- To remove an individual item from the submission list, right-click the item and select **Clear**.
- To remove all items from the submission list, click **Clear All** on the Submission Status window toolbar. Items that are in the process of being submitted are not cleared from the list.

Using the Document Recovery Window

The Document Recovery window enables you to recover or delete the files that SAS Studio has automatically saved. By default, SAS Studio automatically saves your open files, such as SAS programs, flows, queries, and tasks, if SAS Studio times out while you are away or closes unexpectedly due to a situation such as a power loss.

To open the Document Recovery window, select **View** ⇨ **Document recovery**, or click **Recover** in the lower right corner of your SAS Studio browser window.

Note: The Document Recovery window is available only if you have recovered documents. You can use the **Enable document recovery** option to manage the document recovery feature. For more information, see "Setting General Preferences" on page 261.



For each file in the Document Recovery window, you can choose from among these options:

- **No action** - retains the file in the list of recovered documents. The file continues to be available from the Document Recovery window until you recover it or delete it.
- **Recover** - opens the file in a new tab in the work area. The file is deleted from the list of recovered documents.
- **Delete** - deletes the file from the list of recovered documents.

Note: You can also recover or delete all of the documents by clicking **Recover All** or **Delete All**.

When you have specified actions for the recovered files, choose one of these options:

- **Apply and Close** - applies the selected actions to the files and closes the Document Recovery window.
- **Apply** - applies the selected actions to the files, but does not close the Document Recovery window.
- **Close** - closes the Document Recovery window without applying any actions to the files. The files are still available the next time you open the Document Recovery window.

Scheduling a Program, Task, Query, or Flow as a Job

You can schedule saved items, including programs, tasks, queries, and flows, to run at a specified time and frequency. Scheduled items still run even when you are not using SAS Studio.

Scheduling a Program, Task, Query, or Flow

To schedule an item to run at a specified time:

1. Right-click the item in the **Explorer** section of the navigation pane and select **Schedule as a job**.

   Note: You can also schedule items by clicking ⋮ on the item toolbar and selecting **Schedule as a job**.

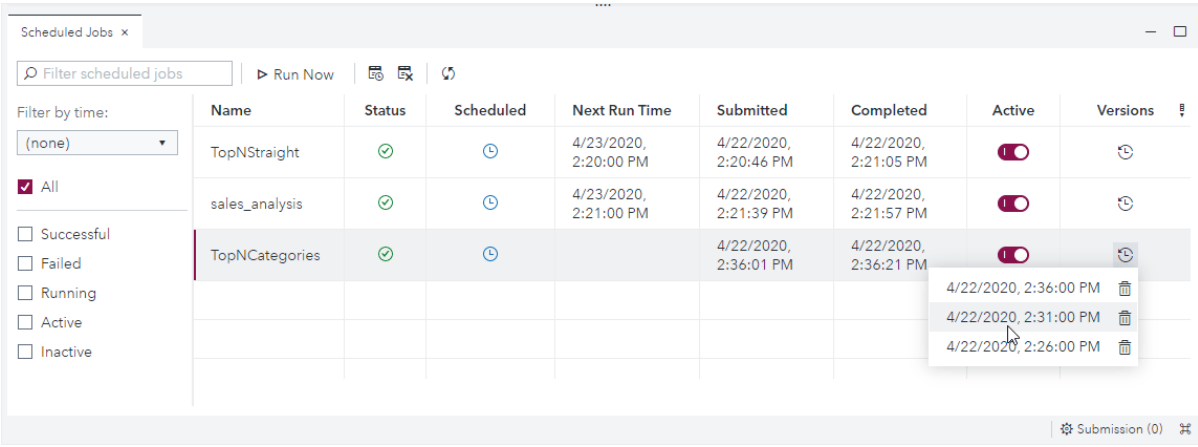
2. In the New Trigger window, specify how frequently to run the job. You can also specify the time at which the job should run and the start date and the end date for running the job.
3. Click **Save**. You can use the Scheduled Jobs window to view and manage your scheduled jobs.

Using the Scheduled Jobs Window

The Scheduled Jobs window displays information about all of your scheduled jobs in SAS Studio. You can schedule saved programs, tasks, queries, and flows, and you can also create and schedule SAS Viya jobs. SAS Viya jobs consist of a program and its definition and can include an HTML form or task prompt to provide a user interface to the job.

To open the Scheduled Jobs window, select **View** ⇨ **Scheduled Jobs**. For information about SAS Viya jobs, see "Managing Jobs" in *SAS Studio Developer's Guide: Working with Jobs*.

The Scheduled Jobs window includes an entry with information about each job that you have scheduled, including the next time the job is scheduled to run as well as the latest submission and completion times of the job.



You can change when a scheduled job runs in the following ways:

- To run a scheduled job immediately, select the job and click **Run Now**. The job opens in a separate browser window.
- To deactivate a scheduled job, click in the row for that job. To reactivate the job, click
- To change when a job is scheduled, click button. The Edit Trigger window opens so that you can change when the job runs.
- To remove a job from the schedule, select the job and click . The job remains in the Scheduled Jobs window, but is no longer scheduled to run. To reschedule the job, right-click the job and select **Schedule**.
- To delete a job, right-click the job and select **Delete job**.

To view a list of all submitted versions of a job, click . You can open the code, log, and results that are associated with the job by clicking the appropriate version in the list. To delete a specific version of a job, click beside the appropriate version.

Note: The code that you open from the Scheduled Jobs window is a copy of the code that SAS Studio ran and is no longer associated with the original program,

task, query, or flow. Editing this code does not affect the original program, task, query, or flow.

You can customize the Scheduled Jobs window in several ways:

- Use the **Filter scheduled jobs** box to specify filter criteria for the Name column.
- Use the **Filter by time** drop-down list to select a specific time period that you want to view.
- Specify whether you want to view all or only selected types of submission status.
- Click to open the Manage Columns window and specify the columns to display in the window.

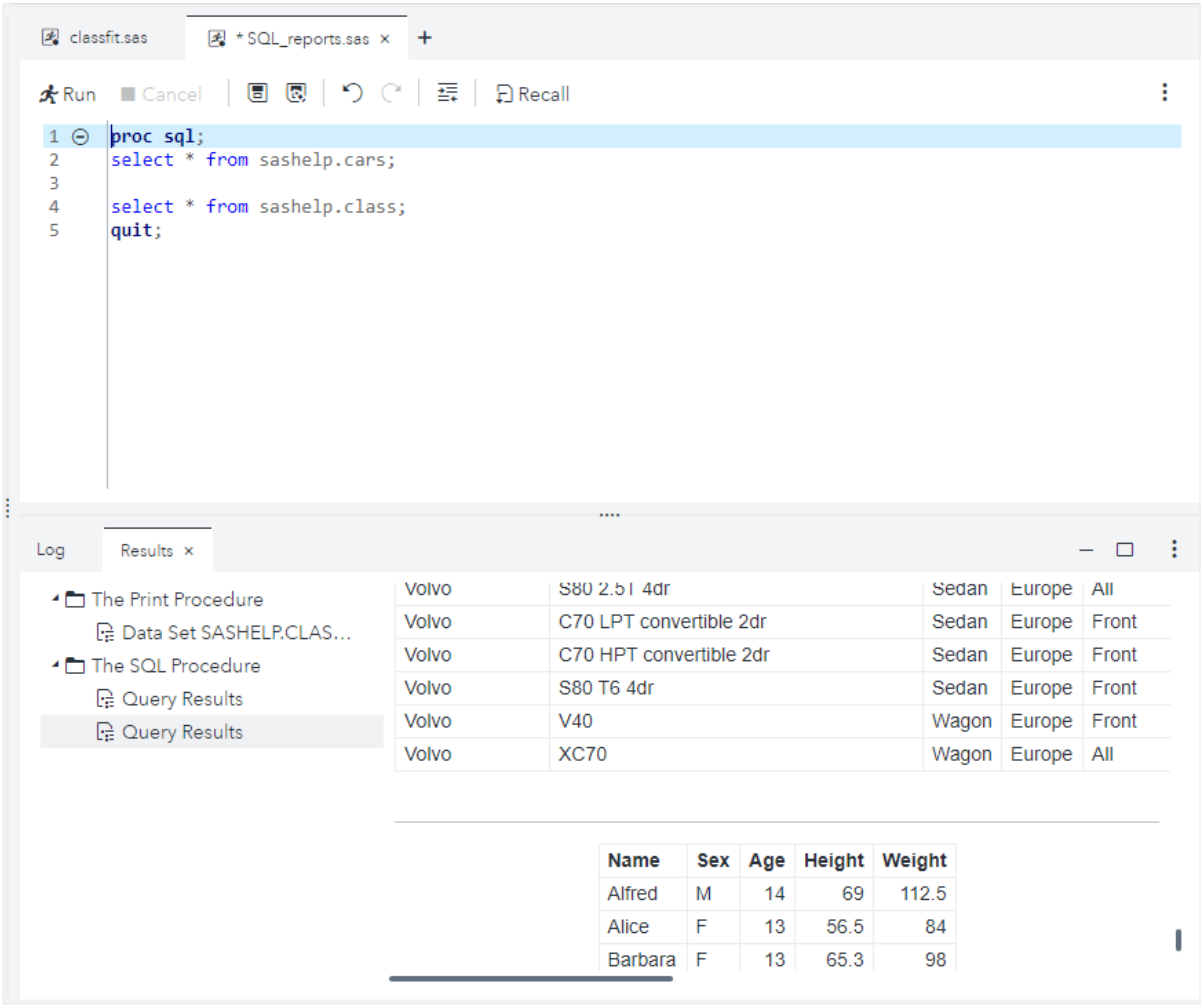
Note: The SAS Studio scheduling feature provides basic scheduling functionality for a single user. Administrators can access more advanced scheduling options by using SAS Environment Manager. SAS Environment Manager enables administrators to view and manage jobs for multiple users. For more information, see "Jobs and Flows Page" in *SAS Environment Manager: User's Guide*.

Understanding Perspectives

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 perspective, that meets your needs best. Perspectives are sets of functionality that are customized to meet the needs of different types of users. SAS Studio includes two perspectives: the Standard perspective and the Interactive perspective.

The Standard perspective is for users who want to be able to write SAS programs, use the predefined tasks, and create flows. In the Standard perspective, SAS program files are run as a whole. By default, the Standard perspective opens with the **Start Page** tab and includes these sections of the navigation pane: Open Files, Explorer, Steps, Tasks, Snippets, Libraries, and Git Repositories.

The Interactive perspective is designed for users who intend to use SAS Studio mainly for writing and editing SAS programs that use interactive SAS procedures. By default, the Interactive perspective opens with the **Start Page** tab, and the **Log** and **Results** tabs are displayed in a separate horizontal tab group. In addition, the log data and results from all of your program tabs are appended together in one log tab and one results tab. In the Standard perspective, there are separate log and results tabs for each program. For more information, see "Working with the Interactive Perspective" on page 44.



The differences between the perspectives can be viewed in the following table:

Element in SAS Studio	Available in Standard?	Available in Interactive?
Navigation Pane sections		
■ **Open Files**	Yes	Yes
■ **Explorer**	Yes	Yes
■ **Steps**	Yes	No
■ **Tasks**	Yes	No
■ **Snippets**	Yes	Yes
■ **Libraries**	Yes	Yes
■ **Git**	Yes	No

Element in SAS Studio	Available in Standard?	Available in Interactive?
■ **File References**	Yes, but not displayed by default	Yes, but not displayed by default
Ability to create and open queries	Yes	No
Ability to create and run a flow	Yes	No
Ability to create and run job definitions	Yes	No
Options to recall code and clear code on submit	No	Yes
Options to generate PDF, Word, RTF, Excel, and PowerPoint output	Yes	No
Ability to import data	Yes	No
Ability to export data	Yes	No
Ability to print data	Yes	No
Ability to validate an expression in the Filter Table Rows window	Yes	No
Ability to validate the autoexec.sas file	Yes	No
Ability to insert custom code	Yes	No
Ability to update column properties	Yes	No

After you have started SAS Studio, you can change the perspective that you are using by selecting **Options** ⇨ **Change perspective** and then selecting the perspective that you want to use.

You can specify which sections of the navigation pane are displayed in SAS Studio by selecting **View** ⇨ **Navigation panes**.

Editing the Autoexec File

The autoexec.sas file includes SAS statements that run each time you start SAS Studio and connect to your SAS server. For example, you can use the autoexec.sas file to assign libraries that you want to be available every time you use SAS Studio in both the Standard and Interactive perspectives.

Note: If you create a library by using the New Library window, you can select the **Persist this library beyond the current session** option to automatically add the LIBNAME statement to the autoexec.sas file. For more information, see "Working with Libraries" on page 8.

To edit the autoexec.sas file:

1. Select **Options** ⇨ **Autoexec file**.
2. Enter the code that you want to include in the autoexec.sas file.
3. To validate your syntax, click **Run**. The **Log** tab opens so that you can view the log.

   Note: The option to run the autoexec file is available only in the Standard perspective. For more information, see "Understanding Perspectives" on page 25.
4. Click **Save** to save and close the autoexec file.

Inserting Custom SAS Code

You can specify the SAS code to run before or after the code for programs, tasks, queries, and imports. This custom code is always run and persists between sessions.

To include custom SAS code:

1. Select **Options** ⇨ **Custom code**.
2. Specify the preamble and postamble SAS code. Click **Run** to validate the syntax.
3. On the **Options** tab, specify where to insert the custom code. You can choose from programs, tasks, queries, and imports. You can also specify whether to include the code if you are running a background submit.
4. Click **Save**.

Resetting Your SAS Session

You can terminate your current SAS session and create a workspace session by using the **Reset SAS Session** option. The **Reset SAS Session** option terminates your current workspace session by running the ENDSAS statement and then creates a new workspace session. When the new workspace session is started,

SAS Studio runs the autoexec file to initialize the session. To reset your SAS session, select **Options** ⇨ **Reset SAS session**.

Note: All files in the Work library are deleted when you reset your SAS session.

Managing Keyboard Shortcuts

SAS Studio includes many keyboard shortcuts that you can use to perform a task. For a list of keyboard shortcuts, see "Keyboard Shortcuts" in *SAS Studio: Accessibility Features*.

Some keyboard shortcuts can be customized. To manage your keyboard shortcuts, select **Options** ⇨ **Manage keyboard shortcuts**. When you enter a new keyboard shortcut, you are prompted to confirm your changes.

Changing Your Compute Context

If you have access to more than one compute context, you can change the context that SAS Studio uses. To change the compute context, click the current compute context that is displayed in the upper right of the application window. Select the compute context that you want to use. When you change the compute context, note the following information:

- Any libraries and file shortcuts that you created are deleted.
- Scheduled jobs and background submissions are not canceled or cleared.
- All open tabs remain open.

For more information, see "Server Contexts: Concepts" in *SAS Viya: Server Contexts*.

Note: You can work on only one compute context at a time.

2

Working with Programs

About the Code Editor

SAS Studio includes a color-coded, syntax-checking editor for editing new or existing SAS programs. The editor includes a wide variety of features such as autocompletion, automatic formatting, and pop-up syntax help. With the code editor, you can write, run, and save SAS programs. You can also modify and save the code that is automatically generated when you run a task.

SAS Studio also includes several sample code snippets that you can use to make programming common tasks easier.

Opening and Creating Programs

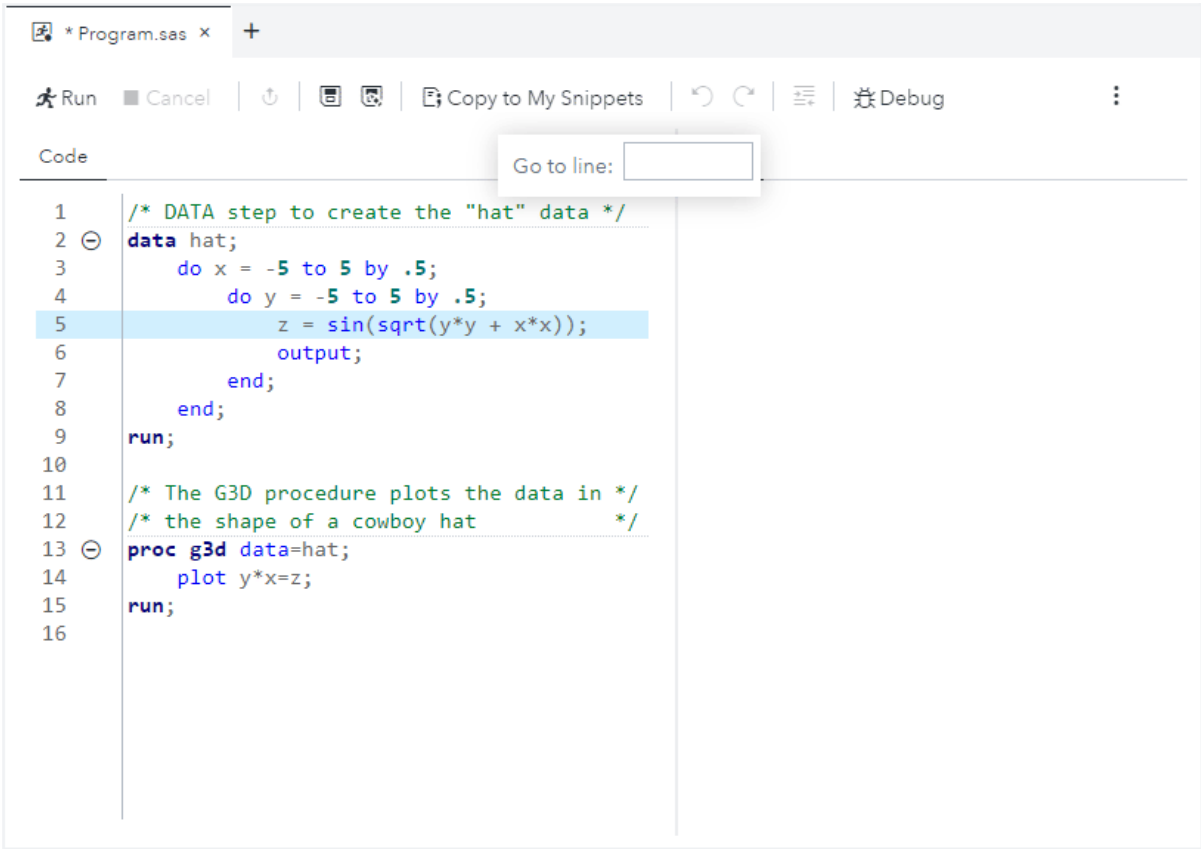
Opening a Program

You can open SAS programs from the **Explorer** section of the navigation pane. To open a program, expand the appropriate folder and double-click the program that you want to open, or right-click the program in the navigation pane and select **Open**. The program opens in a new tab in the work area.

Note: You can also open a file by clicking **Open** on the toolbar and selecting a file from your available folders. If you are opening a program or text file, then you can specify the text encoding that you want to use by selecting **Open with encoding** and clicking **Open**. Use the Select Encoding window to specify the encoding that you want.

Note: Opening very large program files can affect your performance. By default, if you open a program file that is greater than 10 MB, you are prompted to confirm whether you want to continue opening the file. You can use the **Display warning if text files are larger than specified size** option to change the file size that determines when a warning is displayed. For more information, see “Setting General Preferences” on page 261.

In the lower left corner of the Program Editor, you can view the current line and column position of the cursor. If you are viewing a *.sas or *.txt file, the text encoding is displayed as well. To go to a specific line in the program, right-click in the program and select **Go to line**. Enter the line number in the **Go to line** box. You can also right-click in the program and select **Encoding** and select another character-set encoding option.



Creating a Program

To create a program, select **New** ⇨ **SAS Program**.

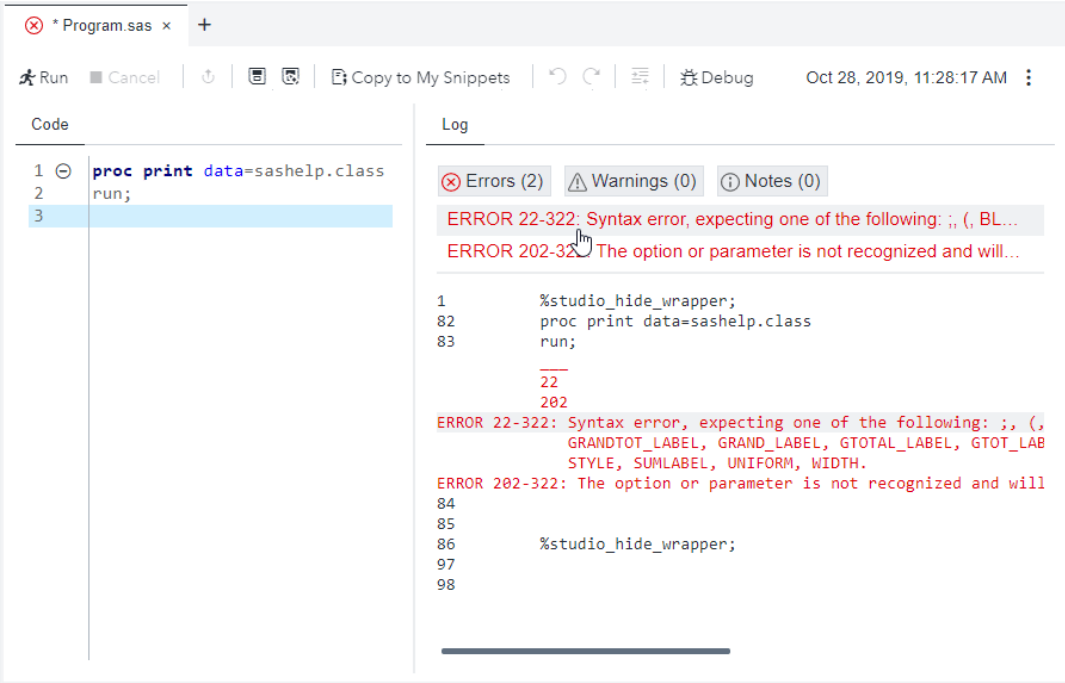
Note: You can also use the **New** menu to begin importing a file as well as create flows, queries, tasks, jobs, text files, and XML files.

Running a Program

After you have written your program, you can run the entire program or you can select specific lines of code to run. To run the entire program, click **Run** on the toolbar. To run a portion of the program, select the lines of code that you want to run and then click **Run**.

By default, if there are no errors, the results open automatically, and if there are errors, the **Log** tab opens. You can click one or more of the **Errors**, **Warnings,** and **Notes** sections to view the messages. When you click a message, SAS Studio highlights it for you in the log so that you can see exactly where the message occurs in the log.

Note: You can specify which tab you want to display after you run a program or task by using the General preferences. For more information, see “Setting General Preferences” on page 261.



Note: Because you are working in a server environment, do not include the ENDSAS statement in your SAS programs. If you run a program that contains ENDSAS, reset your SAS session by selecting **Options** ⇨ **Reset SAS session**.

Saving a Program

To save updates to an existing program in its current location, click on the toolbar. To save the entire program with a new name or location, click . If you want to specify the text encoding when you save a program or text file, right-click in the file, select **Encoding**, and specify the new encoding option before you save the file.

Note: You can also create a SAS program package or a program summary when you save your program. For more information, see “Creating a SAS Program Package” on page 41 and “Creating a Program Summary ” on page 42.

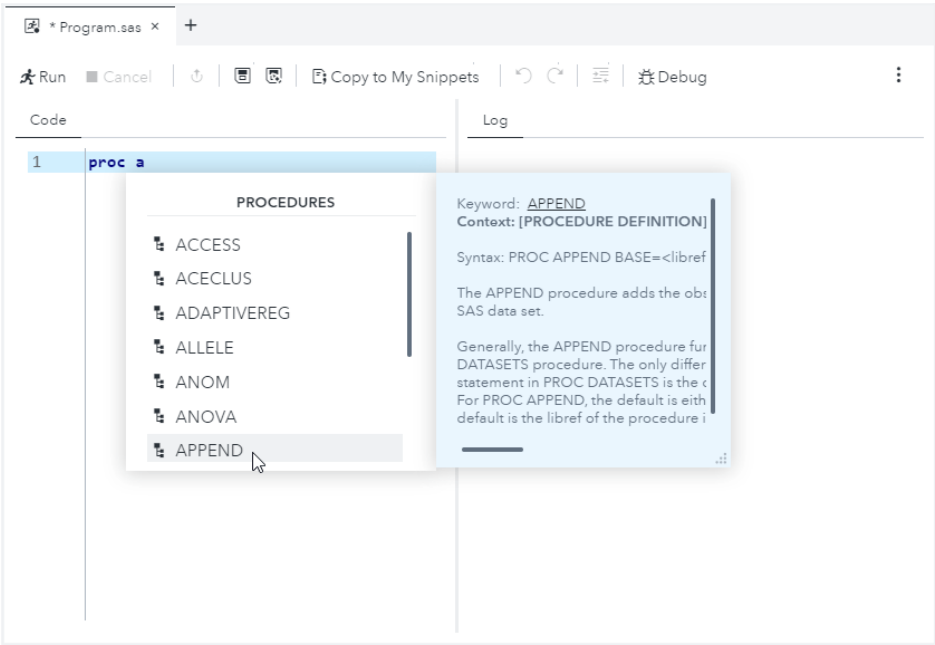
Using the Autocomplete Feature

About the Autocomplete Feature

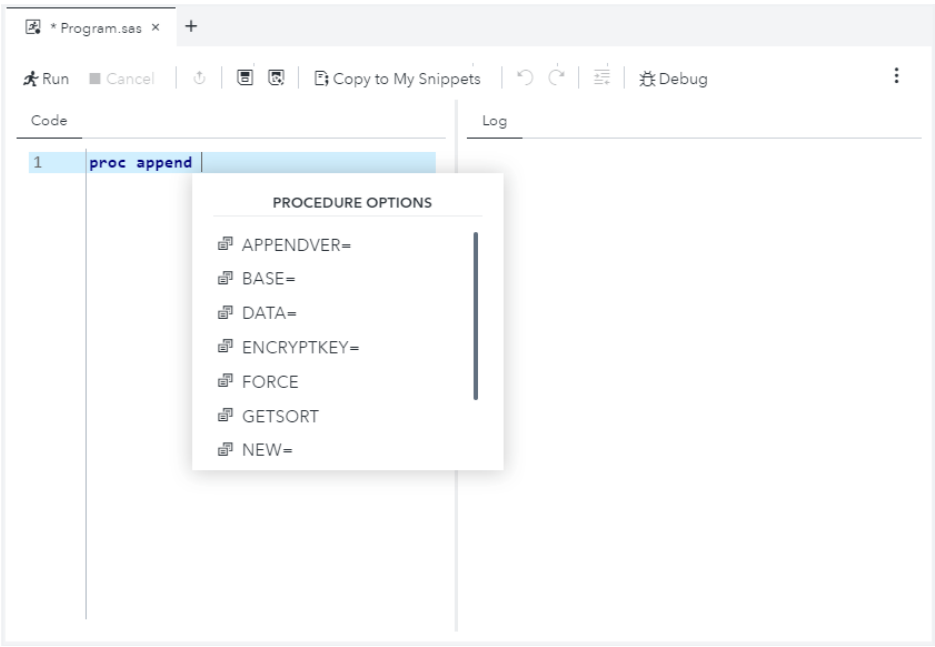
The autocomplete, or code completion, feature in the code editor can predict the next word that you want to enter before you actually enter it completely. The autocomplete feature can complete keywords that are associated with SAS procedures, statements, macros, functions, CALL routines, formats, informats, macro variables, SAS colors, style elements, style attributes, and statistics keywords, and various SAS statement and procedure options. The autocomplete feature can also complete librefs and table names.

Note: The autocomplete feature is available only for editing SAS programs.

This example shows the keywords and help that appear when you enter `proc a` in the code editor.



In this example, you select **APPEND** from the list of procedures so that `proc append` appears in the code editor. When you enter a space, the code editor displays a list of options for the APPEND procedure.



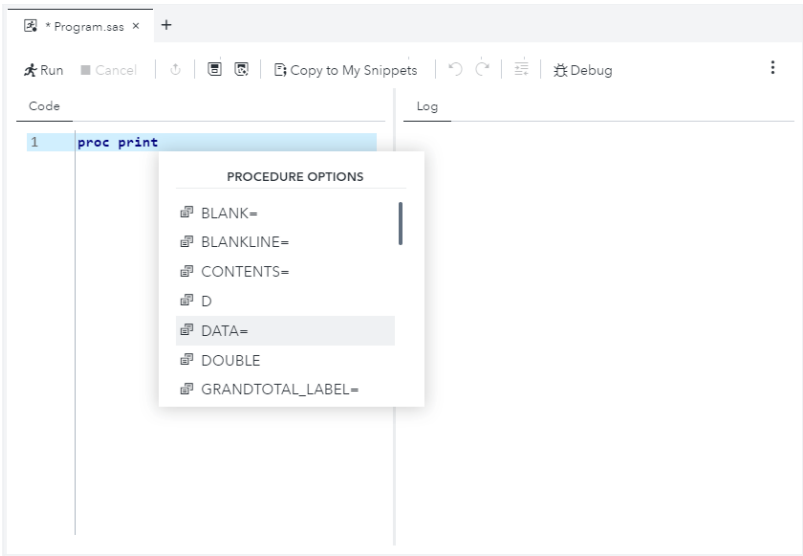
How to Use the Autocomplete Feature

To use the autocomplete feature:

1 How you open the autocomplete list depends on the keyword that you want to add.

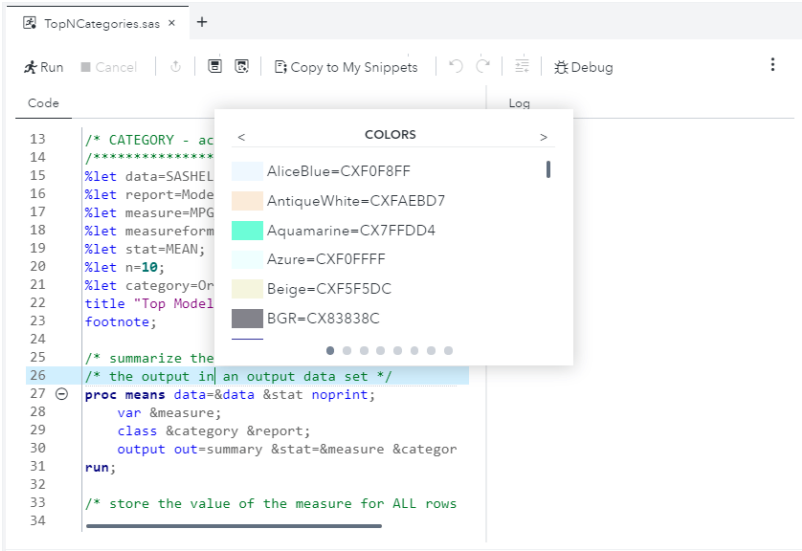
- If you want to add a global statement, DATA step statement, CALL routine, procedure, macro statement, or automatic macro variable, enter the first one or more letters of the keyword that you want to use.

  A window opens with a list of suggested keywords that begin with those letters.



- If you want to specify colors, formats, informats, macro functions, SAS functions, statistics keywords, style elements, or style attributes, position your mouse pointer in a comment and press Ctrl+spacebar. To navigate through the list of options backward, press Ctrl+Shift+spacebar or use the arrow keys at the top of the pop-up window.

  Note: These shortcuts work even if you have deselected the **Enable autocomplete** option in the Preferences window. For more information, see "Customizing the Code Editor" on page 67.



2. You can navigate to the keyword that you want to use in several ways:
   - Continue to type until the correct keyword is selected (because the matching improves as you type).
   - Scroll through the list by using the up and down arrow keys, the Page Up and Page Down keys, or your mouse.
3. You can add the keyword to your program by double-clicking the selected keyword or by pressing the Enter key.

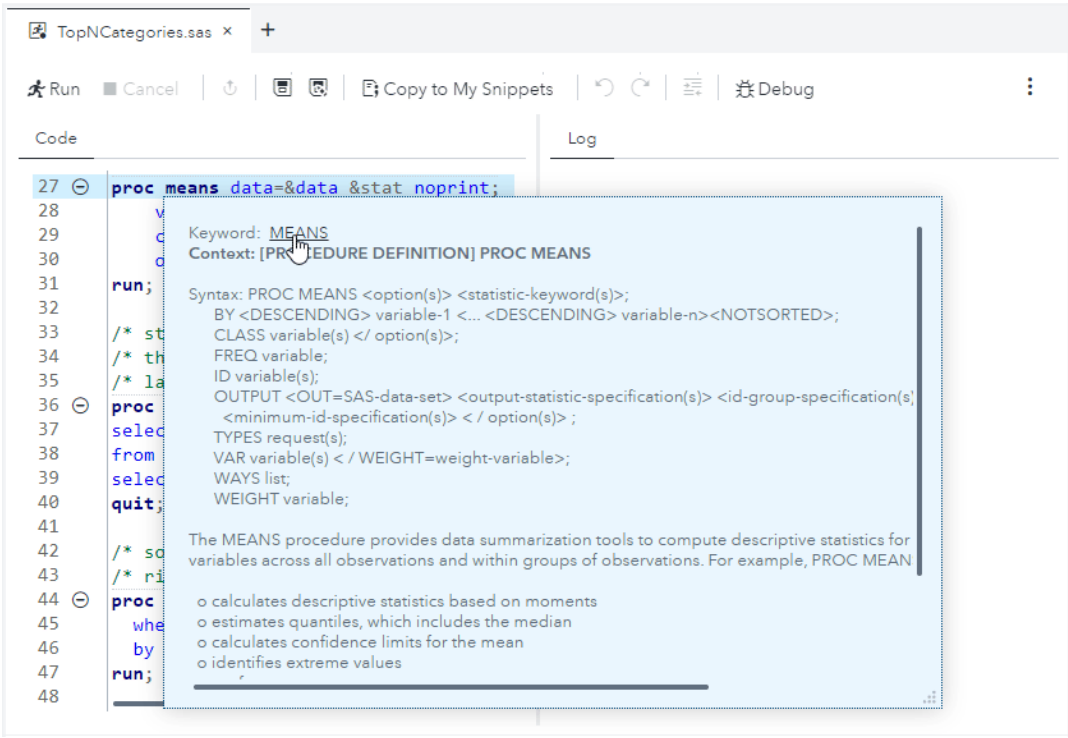
Using the Syntax Help

The code editor displays brief SAS syntax documentation as you write and edit your programs. You can display the Help in the following ways:

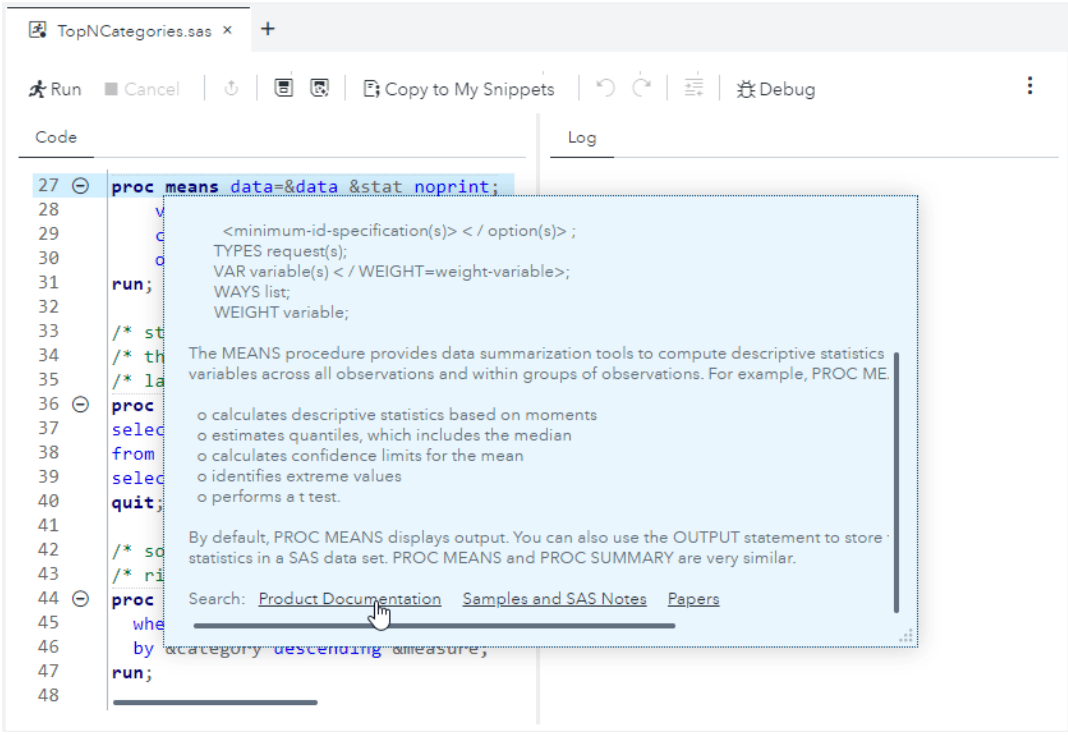
- Right-click a keyword in your program and select **Syntax help**.
- Start typing a valid SAS keyword, and then click a suggested keyword in the autocomplete window.
- Position the mouse pointer over a valid SAS keyword in your program. This works only if you have selected the **Enable hint when you hover over keywords** option in the Editors preferences. For more information, see "Customizing the Code Editor" on page 67.

The SAS Product Documentation provides more comprehensive usage information about the SAS language, but the syntax help in the code editor can get you started with a hint about the syntax or a brief description of the keyword. You can get additional help by clicking links in the syntax help window as follows:

- Click the keyword link at the top of the window to search the support.sas.com website for the keyword.



- Click the links at the bottom of the window to search for the keyword in the SAS Product Documentation, Samples and SAS Notes, and SAS Technical Papers.



Matching Parentheses

You can use the parenthesis-matching feature to track nested parentheses within a program. The code editor highlights both the open and close parentheses. If only one parenthesis is highlighted, then you know that you are missing a parenthesis. This feature can be used to match parentheses, square brackets, angle brackets, and braces.

```
do y = -5 to 5 by .5;
   z = sin(sqrt(y*y + x*x));
```

To match parentheses, position the cursor in front of the open parenthesis or just after the close parenthesis that you want to match. The parenthesis and its match are highlighted. If the parenthesis does not have a match, it is not highlighted.

Note: You can use the Code Editor Settings page to turn the **Highlight matching brackets** option off. You can also select an option to automatically insert a closing bracket or parenthesis. For more information, see "Setting General Code Editor Preferences" on page 267.

Selecting Columns of Text

You do not have to select entire horizontal lines of text. You can select columns or vertical blocks of text.

To select a column or vertical block of text:

- In Window environments, press the Alt key while you select the text with the left mouse button.
- In Mac OS X environments, press the Option key while you select the text with the left mouse button.

Adding Table Names and Column Names

From the **Libraries** section of the navigation pane, you can use a drag-and-drop operation to move table names and column names into the SAS code. For example, you can move the Sashelp.Cars table into the DATA option for the PRINT procedure. When you release the mouse, the fully qualified name for the table appears in your code.

Note: If the table or column name is a reserved word in a database, SAS Studio creates a name literal by automatically adding quotation marks and a lowercase letter `n` to ensure that the name is evaluated correctly by your program.

Inserting a Pathname

From the **Explorer** section of the navigation pane, you can right-click a folder or a *.sas file and select **Insert as path** to add the pathname of the folder or the pathname and file name of the file to your program. For example, you can right-click a folder such as

`/products/salesdata/` and select **Insert as path** to add it to a LIBNAME statement.

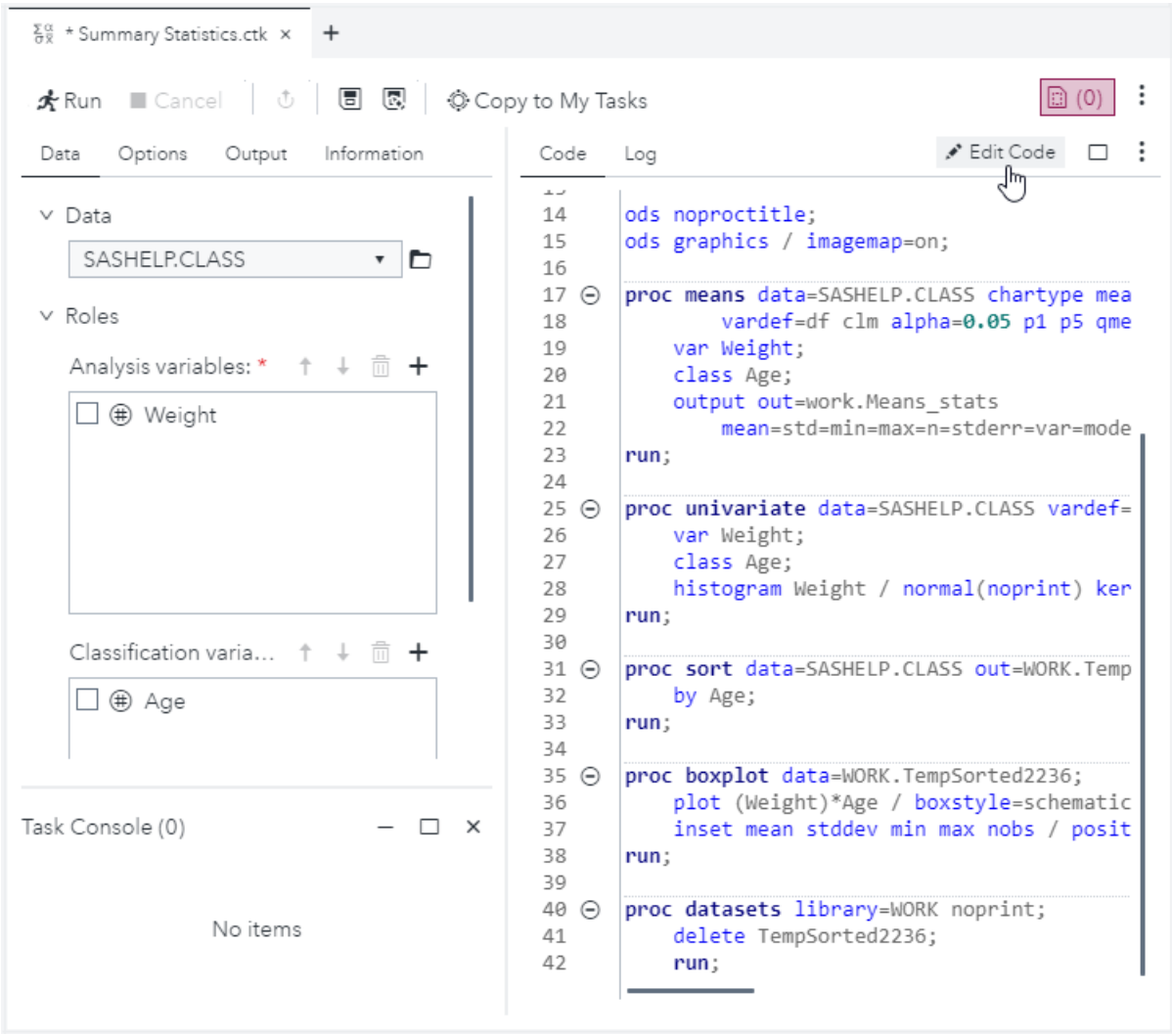
Note: This option is available only for stand-alone programs. It is not available for SAS Program nodes in a flow.

Editing the Code from a Task or Query

You can edit the code that is generated automatically when you run a task or query and then run it with your modifications. When you edit the code, SAS Studio opens it in a separate program window. The code is no longer associated with the original task or query.

To edit a program generated by a task or query:

1. On the appropriate task or query tab in the work area, click the **Code** subtab to display the code that is associated with the task or query.
2. On the toolbar, click **Edit Code**. The code is opened in a new program window.



Creating a SAS Program Package

A SAS Program Package is a file that contains a snapshot of a SAS program along with its log and HTML results. You can create a program package from code that you have written as well as code that is automatically generated when you run a task. When you open a program package in SAS Studio, you can access the code, log, and results without running the program again. If you make changes to the code and rerun it, the package is not automatically updated. You must save the package again to keep the changes.

Note:

- The program package does not include PDF or RTF results.
- You cannot create a program package if you are using the Interactive perspective.

To create a program package file, open the code that you want to use and click . Specify the file location and name, and then select `Package` as the **Type** option.

Note: If you open a program package file and want to save the program or log individually or download the results as an HTML, PDF, or RTF file, you must rerun the program after you open the program package file.

Creating a Program Summary

You can create a summary page for code that you have written as well as code that is automatically generated when you run a task. The Program Summary page is an HTML file that opens in a separate browser tab and includes information about the program execution, the complete SAS source code, the complete SAS log, and the results. To view the Program Summary page for a program, click ⋮ and select **Open in a browser tab** ⇨ **Summary**.

Note: The Program Summary is available only after you have run the program.

You can also save a Program Summary file to a folder that you specify by clicking and selecting `Summary` as the **Save as type** option. The Program Summary is saved as an HTML file.

Using Macro Variables

Macro variables can be used to add information that is obtained when a program or task is run, such as the name and version number of the application. You can reference these items within code, titles, or footnotes by preceding them with "&".

For example, you can use macros in a footnote like the following:

```
Generated with &_CLIENTAPP &_CLIENTVERSION
```

Note: In addition to the following macro variables, you can run this code to see other user-generated and automatic macro variables that are available:

```
%PUT _ALL_;
```

For information about SAS macro functions and variables, see *SAS Macro Language Reference*.

Macro Variable	Description
_CLIENTAPP	Name of the client application.
_CLIENTAPPABBREV	Abbreviated name of the client application.

Macro Variable	Description
_CLIENTMACHINE	Node name of the client machine.
_CLIENTMODE	Type of SAS Studio mode: `viya` (Standard perspective) or `interactive` (Interactive perspective).
_CLIENTVERSION	Application version.
_SASHOSTNAME	Server node name (IP address or DNS name).
_SASPROGRAMFILE	The full path and file name of the SAS program that is currently being run. This macro variable is available only for SAS program files that are saved on the same server on which your SAS Studio code is running.
_SASPROGRAMFILEHOST	The server node name on which the current SAS program is being run.
_SASWORKINGDIR	Current working directory.
SASWORKLOCATION	Location of the Work library.
SYSPROCESSNAME	Name of the current SAS process.
SYSPROCESSMODE	Current SAS session run mode or server type name.
SYSVLONG4	SAS software release number, maintenance level, and four-digit year.

Note: If you specify `%put _all_` or `%put _global_` in your SAS program, the output does not include any special characters. For example, slashes are not included in directory paths. To view the output with these special characters, you must specify the individual macro variable by name, such as `%put &_sasprogramfile;`.

Using Your Submission History

You can use the Submission Status window to access prior versions of your submitted code. Select **View** ⇨ **Submission Status** on the toolbar, or click **Submission** in the lower right corner of your SAS Studio browser window. The Submission Status tab includes an entry for each time you have run a program. When you click a program from the Submission Status window, you open the version of the program that was run at that time. The program opens in a new tab in the work area from which you can make and save changes.

Note: The submission history is cleared automatically when you sign off from SAS Studio.

For more information, see "Using the Submission Status Window" on page 21.

Automatically Formatting Your SAS Code

You can use the code editor to make your programs easier to read by automatically formatting your code. When you automatically format your code, line breaks are added, and each line is correctly indented according to its nesting level. To format the code in the code editor, click on the toolbar. You can also choose to format a selected section of your code.

For example, the following code is difficult to read because it lacks indention and logical line breaks:

```
data topn;
length rank 8; label rank="Rank";
set topn; by &category descending &measure;
if first.&category then rank=0; rank+1;
if rank le &n then output;
run;
```

After you use the automatic code-formatting feature, the program looks like this:

```
data topn;
    length rank 8;
    label rank="Rank";
    set topn;
    by &category descending &measure;

    if first.&category then
        rank=0;
    rank+1;

    if rank le &n then
        output;
run;
```

Working with the Interactive Perspective

What Is the Interactive Perspective?

Some SAS procedures are interactive, which means they remain active until you submit a QUIT statement, or until you submit a new PROC or DATA step. In SAS Studio, you can use the code editor to run these procedures, as well as other SAS procedures, in the Interactive perspective.

By using the Interactive perspective, you can run selected lines of code from your SAS program and use the results to determine your next steps. For example, the OPTMODEL procedure in SAS/OR enables you to model and solve mathematical programming models. By running this procedure interactively, you can quickly check results for parts of the program and determine whether you need to make any modifications without running the entire program.

Running a Program in the Interactive Perspective

When you run a program in the Interactive perspective, SAS Studio does not add any automatically generated code, such as ODS and %LET statements, to your program. In addition, results are generated only in HTML. In the Interactive perspective, the log and results are appended to the existing log and results. Previously submitted code remains active until you terminate it.

For example, suppose you have the following program:

```
proc sql;
  select * from sashelp.cars;

  select * from sashelp.class;
  quit;
```

In the Standard perspective, if you select the first two lines of code and run them, the code runs successfully. If you then select the last two lines of code and run them, the code fails because the PROC SQL statement is missing.

If you switch to the Interactive perspective and follow the same steps, the last two lines of code run successfully because the PROC SQL statement is still active.

Note: For documentation about specific procedures, see the SAS Programmer's Bookshelf on support.sas.com.

Using the DATA Step Debugger

About the DATA Step Debugger

The DATA Step Debugger is a tool that enables you to find logic errors in a DATA step program. With the DATA Step Debugger, you can watch the variable values in a program change as the program runs. You can execute the program line by line, and you can also set specific breakpoints in the program.

Note: The DATA Step Debugger works only with DATA step programs and only if your program does not have a syntax error.

Getting Started with the DATA Step Debugger

What Is Debugging?

Debugging is the process of removing logic errors from a program. Unlike syntax errors, logic errors do not stop a program from running. Instead, logic errors cause the program to produce unexpected results. For example, if you create a DATA step that tracks inventory, and your program shows that you are out of stock when your warehouse is actually full, you have a logic error in your program.

Without the DATA Step Debugger, you could debug your program by adding code, such as PUT statements, to your program and examining the results and log. Although the SAS log helps you identify data errors, the DATA Step Debugger offers you an easier, interactive way to identify logic errors, and sometimes data errors, in DATA steps.

How Debugging Works

When you open a DATA step in the DATA Step Debugger, SAS compiles the step, displays the code in the DATA Step Debugger window, and pauses until you begin execution. Each time execution is paused, the current values of each variable are displayed in the variables pane. For more information, see “Inspecting Variable Values” on page 52.

You can execute your program in the DATA Step Debugger in several ways:

- If you begin execution by clicking ▷, SAS executes each statement in the DATA step.
- To suspend execution at a particular line in the DATA step program, select one or more statements on which to set breakpoints. When you click ▷, SAS executes the program until a breakpoint is reached. For more information, see “Setting and Clearing Breakpoints” on page 50.
- To execute the DATA step one statement at a time, click ⇩. You can examine the variable values after each step in the program. For more information, see “Executing the Program Step by Step” on page 51.
- To execute the DATA step until the value of a specific variable changes, select the variable to watch and click ▷. SAS executes the program until the value of the specified variable changes. For more information, see “Watching a Variable Value” on page 52.

Note: When you use the DATA Step Debugger, note these reminders:

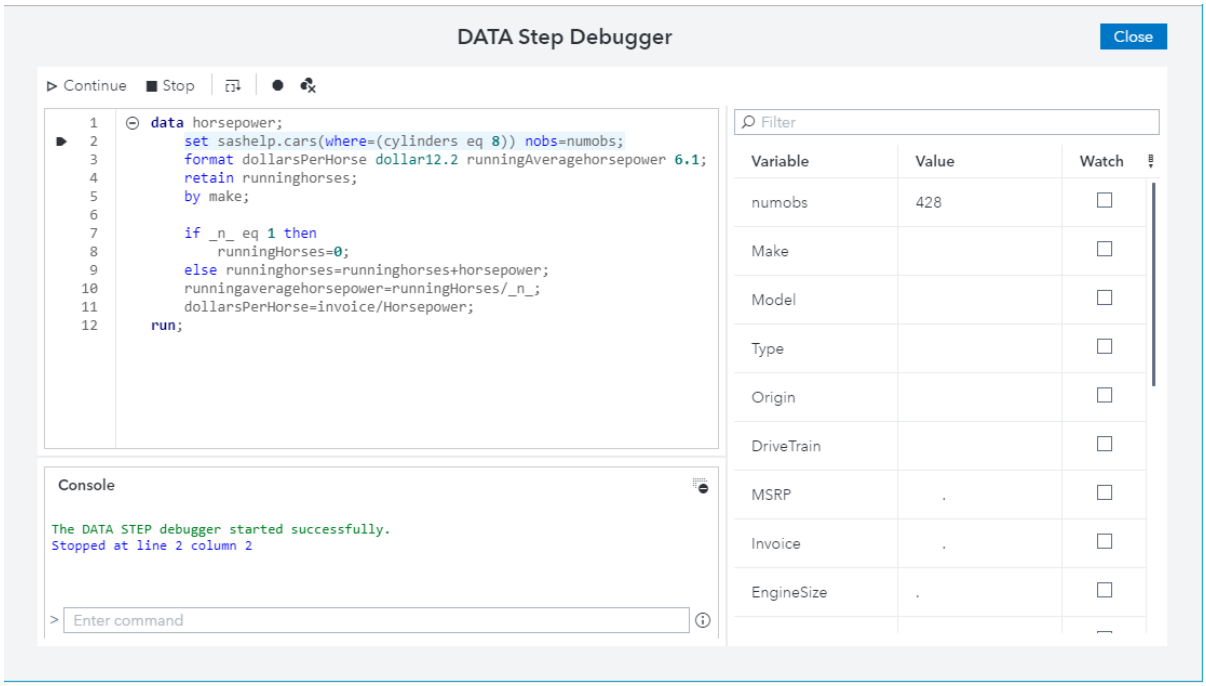
- The DATA Step Debugger is available only in the Standard perspective. For more information, see “Understanding Perspectives” on page 25.
- When you are debugging a DATA step, statements in the DATA step can iterate as many times as they would outside the debugging session. When the last iteration has finished, a message appears in the Debug Console.
- DATA steps that include the DATALINES and CARDS statements cannot be debugged. You can work around this limitation in these ways:
  - Store your data in a flat file, and use the FILENAME statement to read the data into the DATA step.
  - Create two separate DATA steps: one DATA step that includes the DATALINES or CARDS statement and another debuggable DATA step to process the data.
- You can debug only one DATA step at a time. You can use the DATA Step Debugger only with a DATA step and not with a PROC step.

Using the DATA Step Debugger

About the DATA Step Debugger Window

The DATA Step Debugger window includes many features that make it easier to debug your DATA step program:

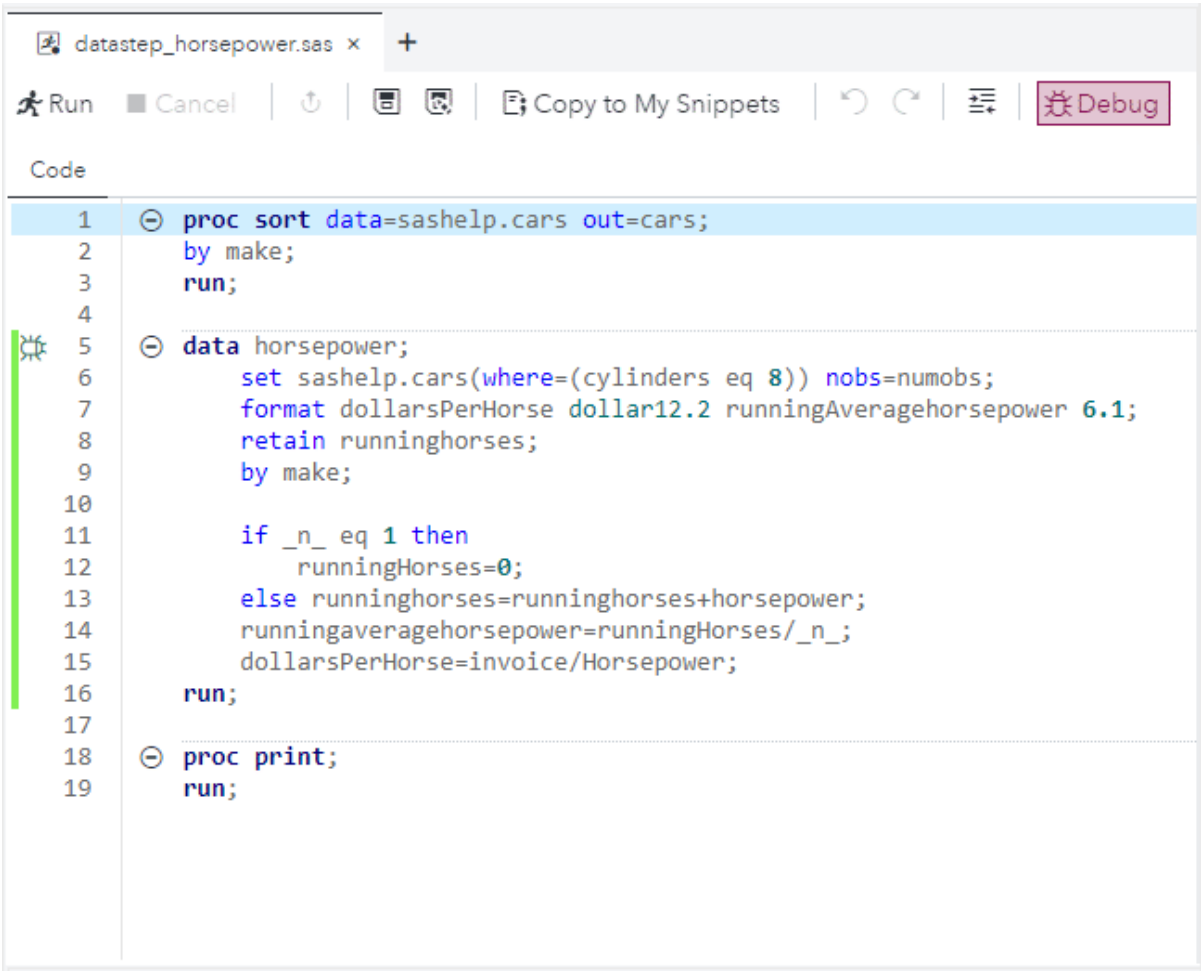
- The currently executing line is highlighted in blue.
- The variables pane on the right side of the window lists each variable in the program with its current value.
- You can use the filter and search box to narrow the list of variables in the variables pane.
- As you step through a program, variable values that change are displayed in red.
- The Debug Console at the bottom of the window displays logging information as you execute the program.
- You can enter debugger commands directly on the command line at the bottom of the window. To view a list of valid commands, click ⓘ.



Opening the DATA Step Debugger

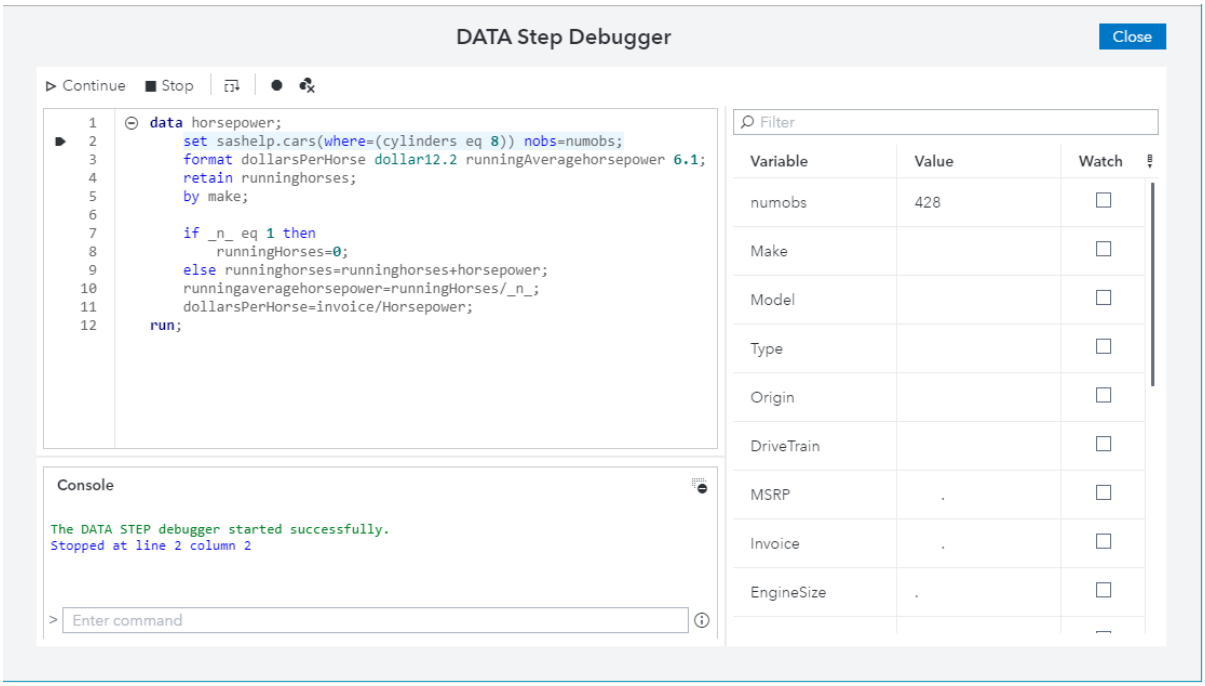
You can run the DATA Step Debugger on any program that contains a DATA step. To get started using the DATA Step Debugger, you must first enable the DATA Step Debugger and select the valid DATA step code that you want to debug.

1 Open your program in the Program Editor and click **Debug** on the toolbar. All sections of DATA step code in the program are highlighted with a green bar in the margin to indicate that they can be debugged.



2 Click in the margin beside the section of code that you want to debug. The DATA step code opens in the DATA Step Debugger window.

Note: You can also open the DATA Step Debugger window by right-clicking in the DATA step code and selecting **Debug DATA step**.

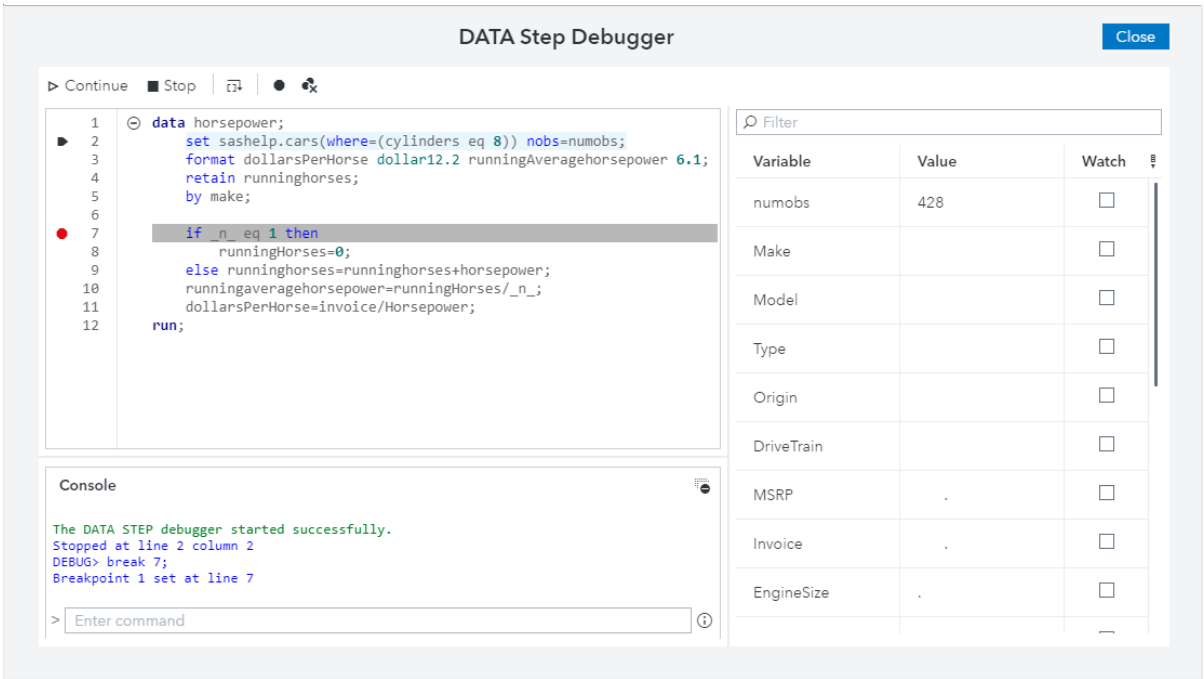


Setting and Clearing Breakpoints

You can set or clear a breakpoint on any executable line in the program. A breakpoint suspends execution of the program at the specified line. Each time you set or clear a breakpoint, information about the breakpoint is displayed in the Debug Console at the bottom of the window.

To set and clear breakpoints:

- You can toggle a breakpoint on and off by selecting the line and clicking ● on the toolbar.
- To clear all breakpoints in the program, click .



Executing the Program Step by Step

You can execute the statements in the program one at a time and view the variable values with each step. As you step through your program, any variable values that change are displayed in red.

To execute the statement starting at the point at which execution was suspended, click . Only one statement is executed at a time.

Starting and Stopping Execution of the Program

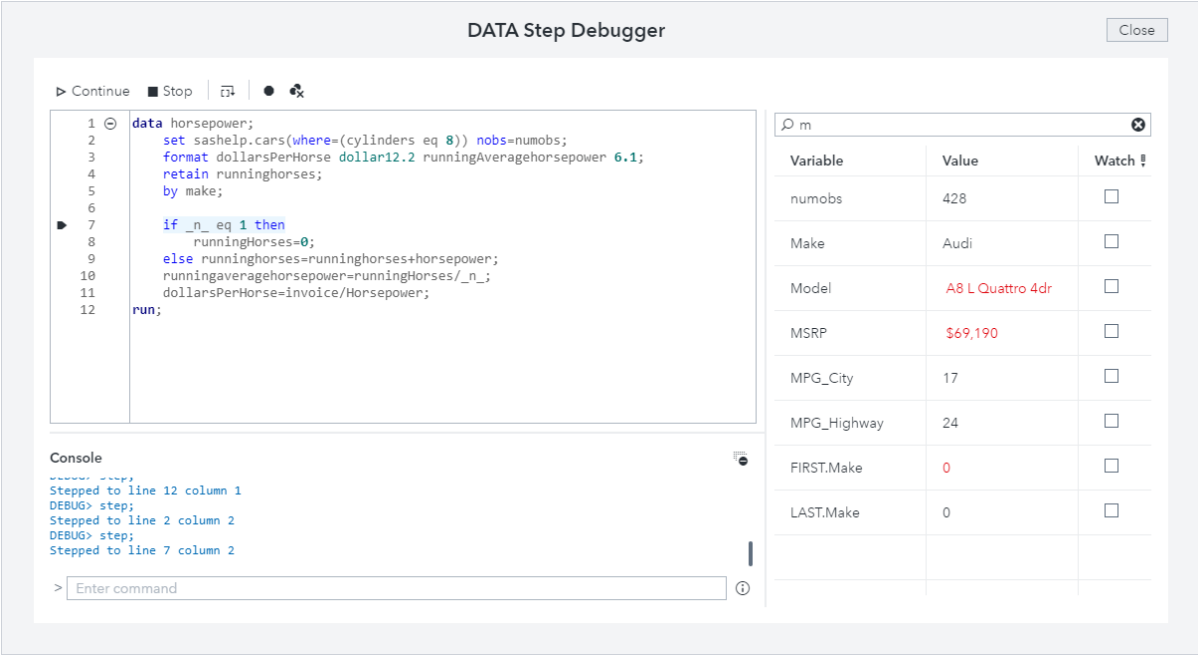
When you start execution of the program in the DATA Step Debugger, the program runs until it reaches a breakpoint or until the value of a watched variable changes. If no breakpoints or watched variables are specified, the program runs to completion.

- To start or continue execution of the program, click .
- To stop execution of the program, click .

Note: You can specify a line in the program at which to restart execution by right-clicking the appropriate line in the program and selecting **Jump**.

Inspecting Variable Values

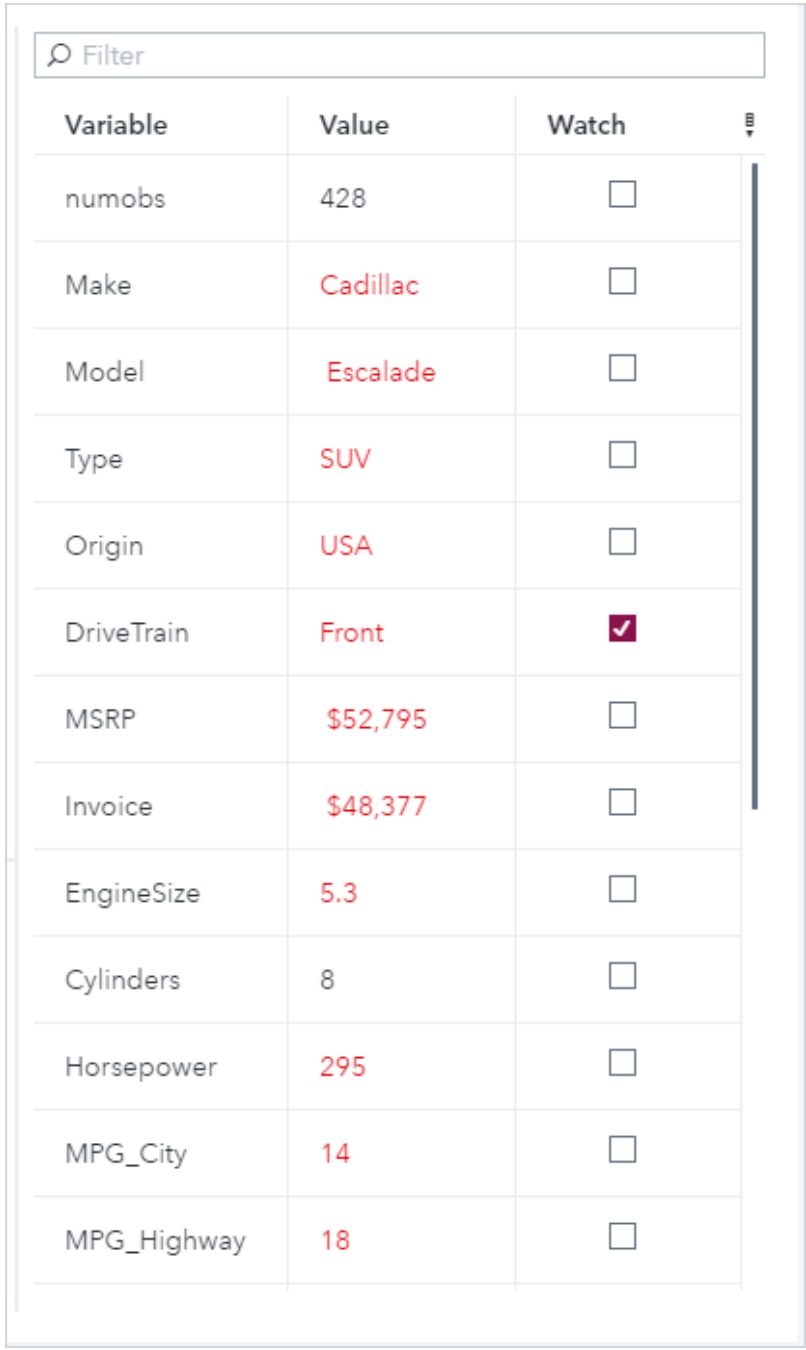
As you debug a program, you can view the values of each variable in the program. You can view the complete list of variables in the variables pane, and you can use the filter and search box to narrow the list of variables. Variable values that have changed are displayed in red.



Watching a Variable Value

You can watch any variable in the program. SAS suspends execution of the program whenever the value of a watched variable changes. Variable values that have changed are displayed in the variables pane in red.

To watch a variable, select the check box for the variable in the variables pane.



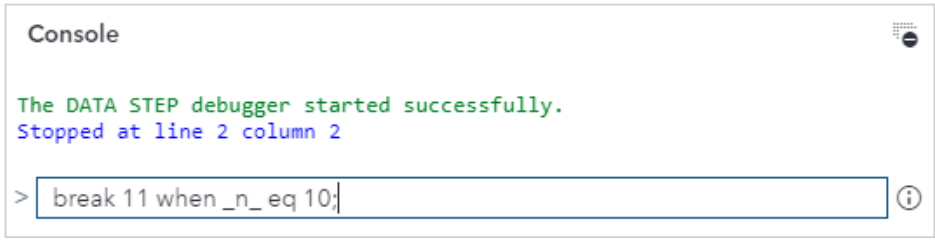
Debugging with the Command Line

You can use the command line to enter debugger commands directly. To view a list of valid commands, click ⓘ.

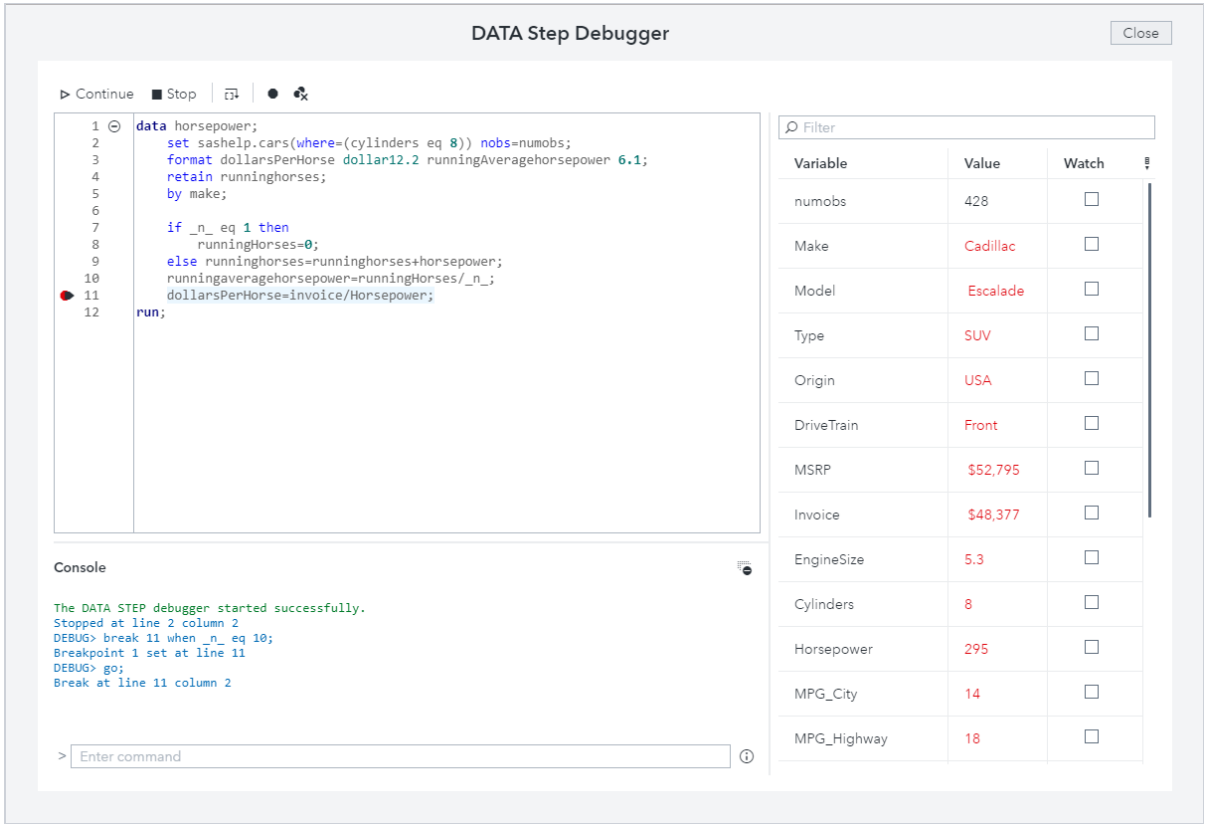
Note: For more information about DATA Step debugger commands, see "Debugger Commands Overview" in *SAS Code Debugger: User's Guide*.

For example, you can use the command line to set a conditional breakpoint so that execution of the program is stopped when a specified condition is met. The following example uses this debugger command to set a breakpoint at line 11 when the value of the variable _n_ is `10`:

```
break 11 when _n_ eq 10
```



A breakpoint is set at line 11, and the program execution stops when the value of _n_ is `10`.



Working with Snippets

Why Use Snippets?

Snippets are lines of commonly used code or text that you can save and reuse. Snippets can contain SAS code, XML code, and text. You can insert snippets in locations including the Program Editor, XML editor, text editor, flows, job definitions, job forms, and job prompts. SAS Studio is shipped with several code snippets.

Snippet Name	Description
Data	
DS2 Code	Provides a template for a DS2 program. DS2 is a SAS programming language that is appropriate for advanced data manipulation. DS2 is included with Base SAS and shares core features with the SAS DATA step. DS2 exceeds the DATA step by adding variable scoping, user-defined methods, ANSI SQL data types, and user-defined packages. The DS2 SET statement accepts embedded FedSQL syntax, and the run-time-generated queries can exchange data interactively between DS2 and any supported database. This allows SQL preprocessing of input tables, which effectively combines the power of the two languages. For more information, see *SAS DS2 Language Reference*.
DS2 Package	Provides a template for a DS2 package. A package is similar to a DS2 program. The package body consists of a set of global declarations and a list of methods. The main syntactical differences are the PACKAGE and ENDPACKAGE statements. These statements define a block with global scope. For more information, see *SAS DS2 Language Reference*.
DS2 Thread	Provides a template for a DS2 threaded program. Typically, DS2 code runs sequentially. That is, one process runs to completion before the next process begins. It is possible to run more than one process concurrently, using threaded processing. In threaded processing, each concurrently executing section of code is said to be running in a thread. For more information, see *SAS DS2 Language Reference*.

Snippet Name	Description
Generate CSV File	Creates a CSV version of a SAS data set in the file that is specified in the FILENAME statement. The file name should be fully qualified and end with .csv.
Generate PowerPoint Slide	Uses PROC SGPLOT to create Microsoft PowerPoint output in the file that is specified in the FILENAME statement. The file name should be fully qualified and end with .ppt.
Generate XML File	Creates an XML version of a SAS data set in the directory that is specified in the FILENAME statement. The directory name should be fully qualified.
Import CSV File	Imports a comma-separated file from the location specified on the FILENAME statement and writes the output to a SAS data set.
Import XLSX File	Imports an XLSX file from the location specified on the FILENAME statement and writes the output to a SAS data set.
Simulate Linear Regression Data	Creates an input data source that you can use for linear regression analysis. Linear regression analysis tries to assign a linear function to your data by using the least squares method.
Simulate One-Way ANOVA Data	Creates an input data source that considers one treatment factor with three treatment levels. When you analyze this data by using the One-Way ANOVA task, the goal is to test for differences among the means of the levels and to quantify these differences.
Data Quality **Note:** Data preparation is required to use the data quality snippets. For more information, see *SAS Data Quality: Getting Started*	
Apply Schemes	Shows how to apply a scheme stored in SAS format using the DQSCHEMEAPPLY function, the DQSCHEMEAPPLY CALL routine, or the DQSCHEME procedure. Schemes enable you to standardize irregular entries in the data so that this data can be used for reporting.
Calculate Matchcodes	Uses the DQMATCH function to generate a match code from a character value. The match codes reflect the relative similarity of data values. Match codes are based on a specified match definition in a specified locale. The match codes are written to an output SAS data set. Values that generate the same match codes are candidates for transformation or standardization.

Snippet Name	Description
Case Data	Uses the DQCASE function to transform a character constant (according to the specified case definition) and return a character value with standardized capitalization.
Cluster (Entity Resolution) – Proc DQMATCH	Runs the DQMATCH procedure to generate match codes and create cluster numbers with multiple criteria statements. PROC DQMATCH creates match codes as a basis for standardization or transformation. The match codes reflect the relative similarity of data values. Match codes are created based on a specified match definition in a specified locale. The match codes are written to an output SAS data set. Values that generate the same match codes are candidates for transformation or standardization.
Extract Data	Uses the DQEXTRACT and DQEXTTOKENGET functions. The DQEXTRACT function returns a delimited string of extraction token values from an input character value that are detected by the extraction definition. The DQEXTTOKENGET function returns an extraction token value from a delimited string of extraction token values.
Guess Locale	Uses the DQLOCALEQUSS function to apply the input character value to the locale-guess definition that are listed in the DQLOCALE= system option or session option. The function returns the name of the locale.
Identify Gender	Uses the DQGENDER function to return a gender from the name of an individual.
Parse Data	Uses the DQPARSE function to return a delimited string of parse token values and the QPARSETOKENGET function to return a parse token value from a delimited string of parse token values.
Perform Identification Analysis	Shows how to perform identification analysis using DQIDENTIFYINFOGET, DQIDENTIFYMULTI, and DQIDENTIFYIDGET functions. ■ The DQIDENTIFYINFOGET function returns a list of names that are supported by a given identification analysis definition. ■ The DQIDENTIFYMULTI function returns all of the identification analysis of a character value. ■ The DQIDENTIFYIDGET function returns a list of identity names.

Snippet Name	Description
Run Data Profiling – Proc DATAMETRICS	Shows how to apply data profiling to a data set using the DATAMETRICS procedure with or without the MULTIIDENTITY= argument specified on the IDENTITIES statement.
Standardize Data	Uses the DQSTANDARDIZE function to standardize the casing, spacing, and format of an input character value and return an updated character value.
Descriptive	
Custom ODS Output	Provides a template for creating HTML, PDF, and RTF output by using the SAS Output Delivery System. For more information, see *SAS Output Delivery System: User's Guide*.
PROC SQL	Provides a template for writing SQL queries. For more information, see *SAS SQL Procedure User's Guide*.
Graph **Note:** For more information about the SGPLOT, SGPANEL, and SGSCATTER procedures, see *SAS ODS Graphics: Procedures Guide*.	
Bar Panel	Uses the VBAR statement in the SGPANEL procedure and enables you to create multiple bar charts.
Box Panel	Uses the VBOX statement in the SGPANEL procedure and enables you to create multiple box plots.
Comparative Scatter Plot	Uses the COMPARE statement in the SGSCATTER procedure. This code snippet creates a comparative panel of scatter plots with shared axes.
Dot Plot	Uses the DOT statement in the SGPLOT procedure. Dot plots summarize horizontally the values of a category variable. By default, each dot represents the frequency for each value of the category variable.
Fit Plot	Uses the REG statement in the SGPLOT procedure. This code snippet produces a regression plot with a quadratic fit and includes confidence limits.
HBar Plot	Uses the HBAR statement in the SGPLOT procedure. This code snippet creates a horizontal bar chart that summarizes the values of a category variable.
HighLow Plot	Uses the HIGHLOW statement in the SGPLOT procedure. High-low charts show how several values of one variable relate to one value of another variable.

Snippet Name	Description
	Typically, each variable value on the horizontal axis has several corresponding values on the vertical axis.
Histogram Plot	Uses the HISTOGRAM statement in the SGPLOT procedure. This code snippet produces a histogram with two density plots. In this snippet, one density plot uses a normal density estimate and the other density plot uses a kernel density estimate.
Scatter Plot Matrix	Uses the MATRIX statement in the SGSCATTER procedure. This code snippet creates a scatter plot matrix.
VBox Plot	Uses the VBOX statement in the SGPLOT procedure. A box plot summarizes the data and indicates the median, upper and lower quartiles, and minimum and maximum values. The plot provides a quick visual summary that easily shows center, spread, range, and any outliers. The SGPLOT and the SGPANEL procedures have separate statements for creating horizontal and vertical box plots.
IML **Note:** These snippets are available only if your site licenses SAS/IML.	
Find Roots of Nonlinear Equation	Enables you to find the roots of a function of one variable. Finding the root (or zero) of a function enables you to solve nonlinear equations.
Fit by using Maximum Likelihood	Uses maximum likelihood estimation to estimate parameters for the normal density estimate.
Generate a Bootstrap Distribution	Uses the IML procedure to create and analyze a bootstrap distribution of the sample mean.
Integrate a Function	Enables you to numerically integrate a one-dimensional function by using the QUAD subroutine in SAS/IML software. Use the QUAD subroutine to numerically find the definite integral of a function on a finite, semi-infinite, or infinite domain.
Simulate Multivariate Normal Data	Simulates data from a multivariate normal distribution with a specified mean and covariance.
Macro **Note:** For more information about SAS macros, see *SAS Macro Language: Reference*.	
SAS Macro	Provides a basic template for working with SAS macros. Macros enable you to perform many tasks, including substituting text in a program. A SAS

Snippet Name	Description
	program can contain any number of macros, and you can invoke a macro multiple times in a single program. For more information, see *SAS Macro Language: Reference*.
SAS Macro Char Functions	Provides several examples of these SAS macros that work with character values: ■ The %EVAL function evaluates arithmetic and logical expressions by using integer arithmetic. This function operates by converting its argument from a character value to a numeric or logical expression. After the expression is evaluated, the result is converted back to a character value. This function is useful because the SAS Macro Facility is basically a text generator. As a result, an arithmetic expression is first converted to a numeric expression. After this numeric expression is evaluated, it is converted back to an arithmetic expression. ■ The %INDEX function returns the position of the first character of a string. ■ The %LENGTH function returns the length of a string. ■ The %SCAN function searches for a word that is specified by its position in a string. ■ The %SUBSTR function produces a substring of a character string. ■ The %UPCASE function converts values to uppercase.
SAS Macro Do Statement	Designates the beginning of a section of a macro definition that is treated as a unit until a matching %END statement is encountered. This macro section is called a %DO group. A simple %DO statement often appears in conjunction with %IF-%THEN-%ELSE statements to designate a section of the macro to be processed depending on whether the %IF condition is true or false. **Note:** SAS also provides a %DO iterative statement, which is different from the code that is generated by this snippet. For more information, see *SAS Macro Language: Reference*.
SAS Macro If Statement	Conditionally processes a portion of a macro. The expression that is the condition for the %IF-%THEN-%ELSE statement can contain only operands that are constant text or text expressions that generate text.
SAS Macro Parameters	Names one or more local macro variables whose values you specify when you invoke the macro. There

Snippet Name	Description
	are two types of macro variables: positional and keyword. Parameters are local to the macro that defines them. You must supply each parameter name. You cannot use a text expression to generate it. A parameter list can contain any number of macro parameters separated by commas. The macro variables in the parameter list are usually referenced in the macro.
SAS Macro Quoting	Provides examples of macro functions that tell the macro processor to interpret special characters and mnemonics as text rather than as part of the macro language. ■ The %STR function masks special characters and mnemonic operators in constant text at macro compilation. This function masks these special characters and mnemonic operators:`+ - * / < > = ¬ ^ ~ ; , # blank AND OR NOT EQ NE LE LT GE GT IN` This function also masks these characters when they occur in pairs and when they are not matched and are marked by a preceding %:`' " ()` ■ The %NRSTR function masks special characters and mnemonic operators in constant text at macro compilation. This function masks all of the special characters and mnemonic operators listed for the %STR function. In addition, the %NRSTR function masks these characters:`& %` ■ The %BQUOTE function masks special characters and mnemonic operators in a resolved value at macro execution. This function masks these special characters and mnemonic operators:`' " () + - * / < > = ¬ ^ ~ ; , # blank AND OR NOT EQ NE LE LT GE GT IN` ■ The %SUPERQ function masks all special characters and mnemonic operators at macro execution but prevents further resolution of the value. This function masks these special characters and mnemonic operators:`& % ' " () + - * / < > = ¬ ^ ~ ; , # blank AND OR NOT EQ NE LE LT GE GT IN` ■ The %QSCAN function searches for a word and masks special characters and mnemonic operators. ■ The %QSUBSTR function produces a substring and masks special characters and mnemonic operators. ■ The %QUPCASE function converts a value to uppercase and returns a result that masks special characters and mnemonic operators.

Snippet Name	Description
	■ The %UNQUOTE function unmasks a value during macro execution so that any special characters and mnemonic operators are interpreted as macro language elements instead of text. For more information about macro complication and macro execution, see *SAS Macro Language: Reference*.
SAS Macro Variables	Provides examples of how to create user-defined global and local macro variables. Macro variables are tools that enable you to dynamically modify the text in a SAS program through symbolic substitution. You can assign large or small amounts of text to macro variables. Then you can use that text by simply referencing the variable that contains the text. Macro variables that are defined by the macro programmer are called user-defined macro variables. Macro variables that are defined by the macro processor are called automatic macro variables. You can define and use macro variables anywhere in SAS programs, except within data lines.
SAS Viya Cloud Analytic Services	
Create CAS Connection	Creates a connection to a CAS server. You must specify values for the CASHOST= and CASPORT= system options. The CAS statement connects the default session to the specified CAS server and CAS port.
Delete caslib	Deletes the specified CAS library.
Delete Table or File from caslib	Deletes a table or file from the CAS library. You can also remove an in-memory table.
Disconnect CAS Session	Disconnects from the CAS session named mySession. Before you disconnect, you can specify a value for the time-out (in seconds). You can reconnect to the session before the time-out expires. Otherwise, the session is terminated.
Generate SAS librefs for caslibs	Creates a default CAS session and generates SAS librefs for existing CAS libraries so that the librefs are visible in the Libraries pane.
List CAS Session Options	Lists session options for the specified CAS session.
List CAS Sessions for SAS Client	Lists all the CAS sessions and session properties that are created by the SAS client or reconnected to by the SAS client.

Snippet Name	Description
List CAS Sessions for User ID	Lists all the CAS sessions that are known to the CAS server for the user ID that is associated with yourSessionName.
Load Data to caslib	Loads a table to the specified CAS library. The snippet includes the PROMOTE option so that the loaded data is available to all your active sessions.
New CAS Session	Starts a new CAS session named mySession using the existing CAS server connection. When starting a new session, you can specify the CAS library to use, the time-out (in seconds) for the session, and the locale of the session.
New caslib for Path	Creates a CAS library (myCaslib) for the specified path (`/filePath/`) and session (mySession). If you omit the SESSREF= option, the caslib is created and activated for the current session. The SUBDIRS option extends the scope of myCaslib to include subdirectories of (`/filePath/`). The LIBNAME statement creates a SAS libref for the CAS library.
Reconnect CAS Session	Reconnects to a CAS session named mySession.
Save Table to caslib	Creates a permanent copy of an in-memory table (*table-name*) from *sourceCaslib*. The in-memory table is saved to the data source that is associated with the target caslib *targetCaslib* using the specified name (*filename*). **Note:** You can determine the caslib that is associated with a CAS engine libref by right-clicking the libref in the Libraries pane and selecting Properties. The Server Session CASLIB field displays the caslib.
Terminate CAS Session	Terminates the CAS session named mySession. No reconnection is possible.
SAS Viya Image Processing You must license and install SAS Visual Data Mining and Machine Learning to use these snippets.	
Convert Color	Converts the color space of one or more images. You can specify any of these values for the TYPE parameter: 'BGR2HSV', 'BGR2RGB', 'BGR2YUV', 'COLOR2GRAY', 'GRAY2COLOR', 'HSV2BGR', 'RGB2BGR', or 'YUV2BGR'. The input is a CAS table that contains the images that you want to convert, and the output is a CAS table that contains the converted images.

Snippet Name	Description
Display Image	Displays an image that is saved from SAS Studio to your computer. This snippet creates an annotation data set that contains a reference to the image and uses PROC SGPLOT to display the image. Use the IMAGE= variable to specify the fully qualified name and location of the image that you want to display. The SAS server must be able to access this location.
Load Images	Loads the image action set from the specified path and creates a CAS table. The path parameter points to the directory that contains the images that you want to load. The path parameter is a subdirectory in the caslib directory where the images are stored. A typical workflow includes loading images and then processing them by using one of the other snippets to resize, rescale, or mutate them. The output from one snippet can be used as input for another snippet. Images can be saved to your computer for further processing by using the Save Images snippet.
Mutate Images	Mutates one or more images using different augmentation techniques. You can specify any of these values for the TYPE parameter: 'COLOR_JITTERING', 'COLOR_SHIFTING', 'DARKEN', 'HORIZONTAL_FLIP', 'INVERT_PIXELS', 'LIGHTEN', 'ROTATE_LEFT', 'ROTATE_RIGHT', 'SHARPEN', or 'VERTICAL_FLIP'. The input is a CAS table that contains the images that you want to change, and the output is a CAS table that contains the mutated images.
Rescale Images	Changes the depth of one or more images based on the options that you specify. You can specify any of these values for the TYPE parameter: "TO_8U", "TO_32F", or "TO_64F". You can also use the ALPHA and BETA parameters to scale the values. The input is a CAS table that contains the images that you want to rescale, and the output is a CAS table that contains the rescaled images.
Resize Images	Resizes one or more images based on the HEIGHT and WIDTH parameters that you specify. The HEIGHT parameter corresponds to the number of rows, and the WIDTH parameter corresponds to the number of columns. The input is a CAS table that contains the images that you want to resize, and the output is a CAS table that contains the resized images.
Save Images	Saves the specified images to the specified subdirectory of a previously defined caslib, such as CASUSER. You must specify the name of a subdirectory in the caslib root directory. The names of

Snippet Name	Description
	the saved images start with the value of the PREFIX parameter.
SAS Viya Machine Learning You must license and install SAS Visual Statistics to use these snippets.	
Compare Several ML Algorithms	Demonstrates fitting and comparing several Machine Learning algorithms for predicting the binary target in the Hmeq data set, which is included in the Machine Learning sample data library. You must run the Load Data and Prepare and Explore Data snippets before you run this snippet.
Compare Two ML Algorithms	Demonstrates fitting and comparing two Machine Learning algorithms for predicting the binary target in the Hmeq data set, which is included in the Machine Learning sample data library. You must run the Load Data and Prepare and Explore Data snippets before you run this snippet.
Generalized Linear Models	Demonstrates fitting and assessing generalized linear models using the GENSELECT procedure. You must run the Load Data and Prepare and Explore Data snippets before you run this snippet.
Load Data	Demonstrates how to load local data into CAS.
Prepare and Explore Data	Modifying, and preparing data prior to modeling. This snippet uses the Hmeq data set from the Machine Learning sample data library. The Hmeq data set is used as input and creates the Hmeq_prepped data set. The Hmeq_prepped data set is used in subsequent examples. **Note:** You might see a warning message about missing values when you run this snippet. The sample data includes some missing values that can be used by other procedures.
Supervised Learning	Shows the entire workflow process, including data exploration and preparation, modeling, and evaluation. This snippet uses the Sampleml.Hmeq data set, which is included in the Machine Learning sample data library.
Unsupervised Learning	Shows the entire workflow process, including data preparation, analysis, and visualization of the results. This snippet uses the Sashelp.Iris data set.

Create a Snippet

To create your own snippet:

1. Open a file in SAS Studio and select the lines or code or text that you want to save as a snippet. If you do not select any lines of code or text, the entire file is saved as a snippet.
2. On the toolbar, click **Copy to My Snippets**.

   Note: You can also copy a snippet to the **Explorer** section in the navigation pane. Right-click an existing snippet in the **Snippets** section in the navigation pane and **Copy snippet to** ⇨ **Explorer**. The snippet is saved as a *.csm file in the folder that you specify.
3. Enter a name for the snippet.
4. If you want to quickly access the snippet by entering *@snippet-abbreviation* in the Program Editor, enter an abbreviation for the snippet. You can also add a description.
5. Select the folder in which you want to save the snippet and click **OK**. The snippet is added to the folder that you specified.

Note: You can also upload existing snippet files to your `My Snippets` folder and subfolders. The file type must be *.csm. Select the folder that you want to use and click . Click + to browse for the file, and then click **Upload**.

To add a saved *.sas file to your `My Snippets` folder, right-click the .sas file in the **Explorer** section in the navigation pane and select **Copy to My Snippets**.

How to Insert a Code Snippet

To insert a snippet:

1. Click the location where you want to insert the snippet.
2. In the navigation pane, click to open the Snippets section.
3. You can add a snippet to your file in these ways:
   - use a drag-and-drop operation to insert the snippet into your file.
   - right-click the name of the snippet and select **Insert into code**.
   - right-click in the file and select **Insert snippet**, and then select the snippet that you want to use.

- enter @*snippet-name* in your code and press Enter. You can also enter @ and use the pop-up window to select from a list of snippets in your My Snippets folder.

  Note: In order to use this functionality, you must specify an abbreviation for the snippet. To verify whether your snippet has an abbreviation, select the snippet and click . To add or edit the abbreviation, right-click the snippet and select **Edit**. For more information, see “Create a Snippet” on page 66. You must also select the **Enable snippet abbreviations** preference. For more information, see “Setting Editors Preferences” on page 266.

Customizing the Code Editor

The Preferences window enables you to change several options that affect the features in the code editor, including autocompletion and color coding.

To access the editor options, select **Options** ⇨ **Preferences**. Click **Editors**.

For more information, see “Setting Editors Preferences” on page 266.

3

Working with Queries

What Is a Query?

A *query* enables you to extract data from one or more tables according to criteria that you specify. You can create a query that is based on only one table, or you can join tables. When you create a query, SAS Studio generates Structured Query Language (SQL) code, which you can view and edit. You can also choose to generate your query using the FEDSQL procedure. For more information, see "Generating a FedSQL Query" on page 98.

Note: Queries are available only in the Standard perspective. For more information, see "Understanding Perspectives" on page 25.

Understanding the Differences between a Stand-Alone Query and a Flow Query

You can create both stand-alone queries that you can run individually as well as queries within a flow that can be run only as part of the flow. Not all query functionality is available if you are running a query in a flow. For more information, see "What Is a Flow?" on page 102.

The differences between stand-alone and flow queries can be viewed in the following table:

Query Feature	Stand-Alone Query	Flow Query
Output options		
Table	Yes.	Yes.
View	Yes, unless the query uses PROC FEDSQL.	No.
Report	Yes.	No.
Options to specify library and table name	Yes.	No. To specify the library and table name, you must add a Table node to the flow.

Query Feature	Stand-Alone Query	Flow Query
Columns area		
Options to add and remove tables	Yes.	No.
Option to view table properties	Yes.	No. You can update only the table alias.
Select tab		
Ability to specify a format for a column	Yes.	Yes, unless the query uses PROC FEDSQL.
Expression Builder		
Ability to display a list of column values	Yes.	No.
Option to validate syntax and view validation log	Yes.	No.
Preferences		
Ability to apply preferences to a query	Yes.	No.
Save options		
Ability to save the query as a separate file	Yes. The query can be saved as a *.cqy file.	No. You can save the query only as part of the flow.

Migrating a Saved Query from an Earlier Version of SAS Studio

Saved queries from earlier versions of SAS Studio can be run in SAS Studio 2020.1 and later if the following criteria are met:

- The data that was used in the original query is available in SAS Studio 2020.1 and later.
- The data that is available in SAS Studio 2020.1 and later is not missing any columns that were used in the original query.
- All tables that are included in the original query must have a defined join.

Note: When you first save a query from an earlier version of SAS Studio in SAS Studio 2020.1 and later, the Save As window opens. If you save the migrated query with the same name as the earlier version, you can no longer open the query in an earlier version of SAS Studio.

If a saved query cannot be migrated, SAS Studio converts the query to a program file and includes a note in the program that explains why the query could not be migrated.

Creating a Stand-Alone Query

Creating a Query

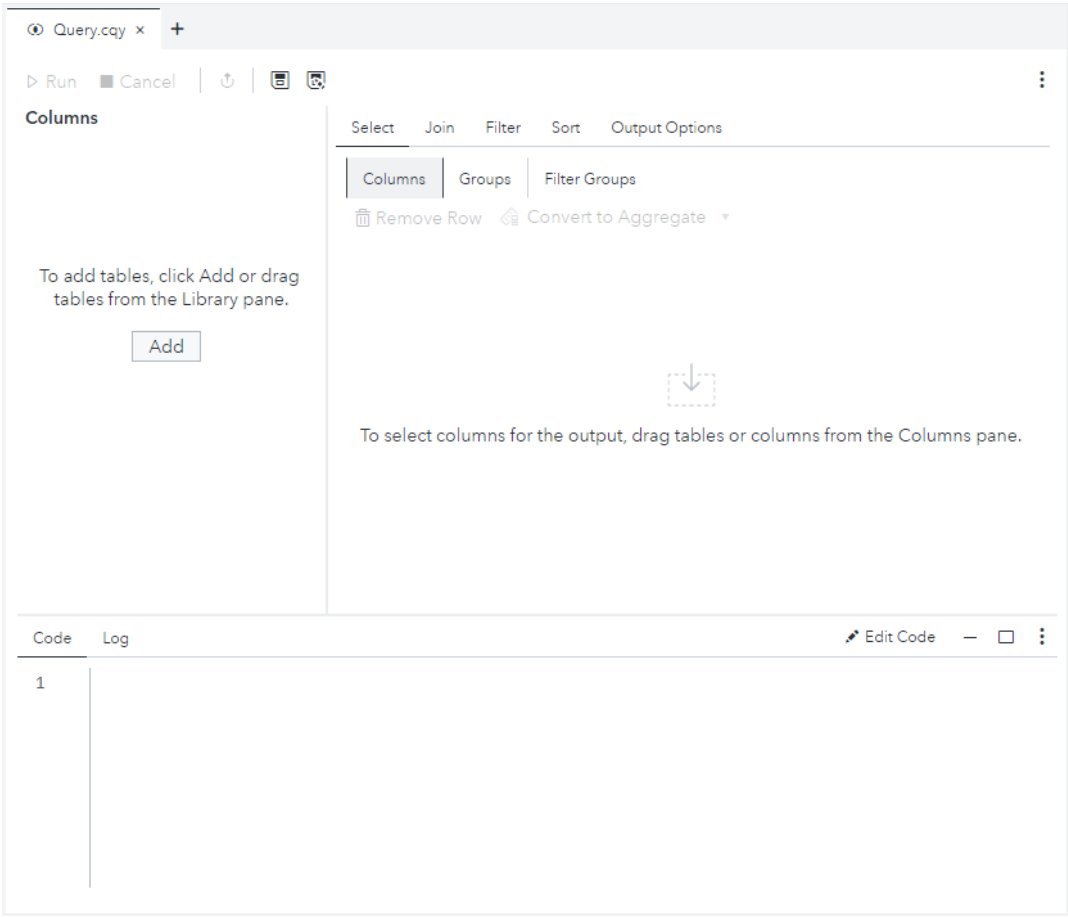
On the main SAS Studio toolbar, select **New** ⇨ **Query**.

Note: You can also create a query by right-clicking one or more tables in the **Libraries** section in the navigation pane and selecting **Create query**.

Note: For information about creating a query in a flow, see “Creating a Query in a Flow” on page 141.

By default, the Query tab contains three main areas:

Columns area	provides access to the tables and columns in your query.
query definition area	enables you to specify and group columns in the query, create joins between tables, filter the data in the query, sort the output, and specify output options.
Code, Log, and Results area	the **Code** tab displays the code that is automatically generated as you build the query. The **Log** tab displays the query log when you run the query. If your output type is `Table` or `View` and you are generating your code using PROC SQL, an **Output Data** tab is added to this area when you run the query. If your output type is `Report`, a **Results** tab is added to this area when you run the query.



Adding Tables to a Query

In the Columns area of the Query tab, click **Add**. Use the Select Table window to browse for the library and one or more tables that you want to use, and click **OK**.

Note: You can also add tables to the query by using the **Libraries** section in the navigation pane. Click in the navigation pane to open the **Libraries** section and expand the appropriate library. Drag one or more tables to the Columns area of the Query tab.

Understanding Joins

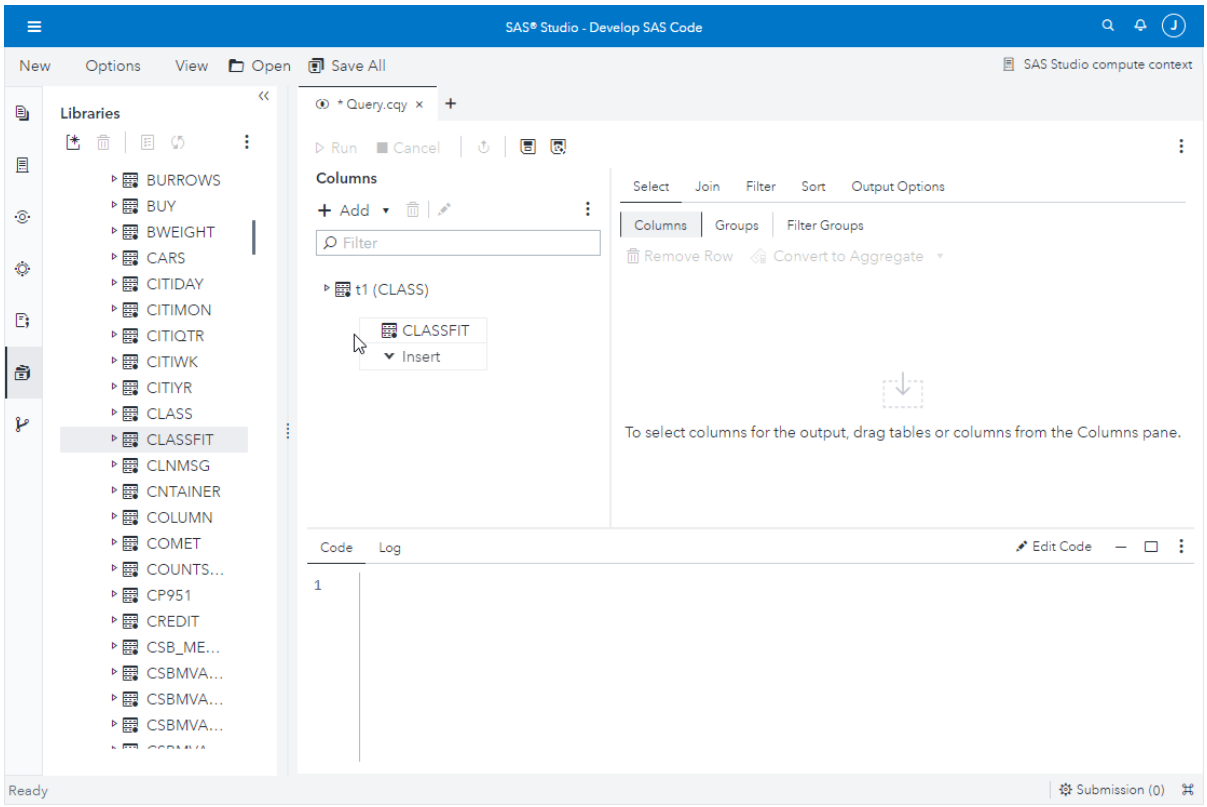
Joining Tables

When you create a query, you can join multiple tables. SAS Studio can automatically join the tables for you, or you can manually create the join. When you join two tables, SAS Studio attempts to join the tables by columns that have the same name and type. If no matches for column name and type are found, the tables are joined using the first column in each table. If this is not the correct join for your query, you can edit the join and specify the join criteria that you want to use.

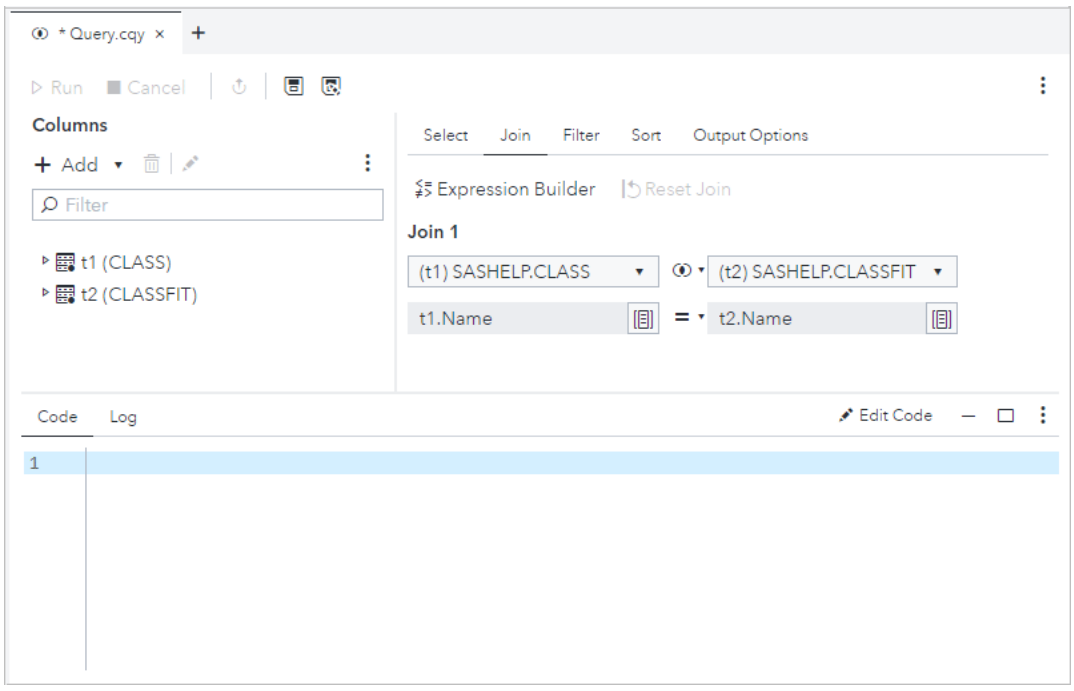
Creating a Join

When you add a table to an existing table in your query, SAS Studio automatically creates a join for you.

To add another table to your query, click **Add** in the Columns area of the Query tab. Use the Select Table window to browse for the library and table that you want to use and click **OK**. You can also drag a table from the **Libraries** section in the navigation pane to the Columns area.



Click the **Join** tab to view the join criteria. In the following example, the Classfit table is automatically joined to the Class table by using the Name column in both tables.



If a suitable join cannot be created automatically, you can specify the join condition manually. For more information, see “Modifying an Existing Join” on page 77.

Editing Table Properties in a Query

You can create an alias for a table in a query.

To specify an alias:

1. In the left pane of the Query tab, right-click the table that you want to change and select **Properties**.
2. In the **Alias** box, enter the alias for the table.

   Note: Alias names must be unique within a query.

3. Click **OK** in the Properties window to close the window and save your changes. The alias is displayed in the list of tables on the Query tab.

Understanding the Types of Joins

SAS Studio supports four different types of joins. You can select the type of join you want by modifying an existing join.

You can select the join option that you want to use in the Join window.

SAS Studio Join Type	Join Icon	Description
Inner join		The output rows include only those for which the column in the first table matches the joining criterion of the column in the second table. Joins are inner joins by default.
Left join		The output rows include all rows from the first table and the rows from the second table in which the joining criterion is met.
Right join		The output rows include all rows from the second table and the rows from the first table in which the joining criterion is met.
Full join		The output rows include all matching and nonmatching rows from both tables.

Modifying an Existing Join

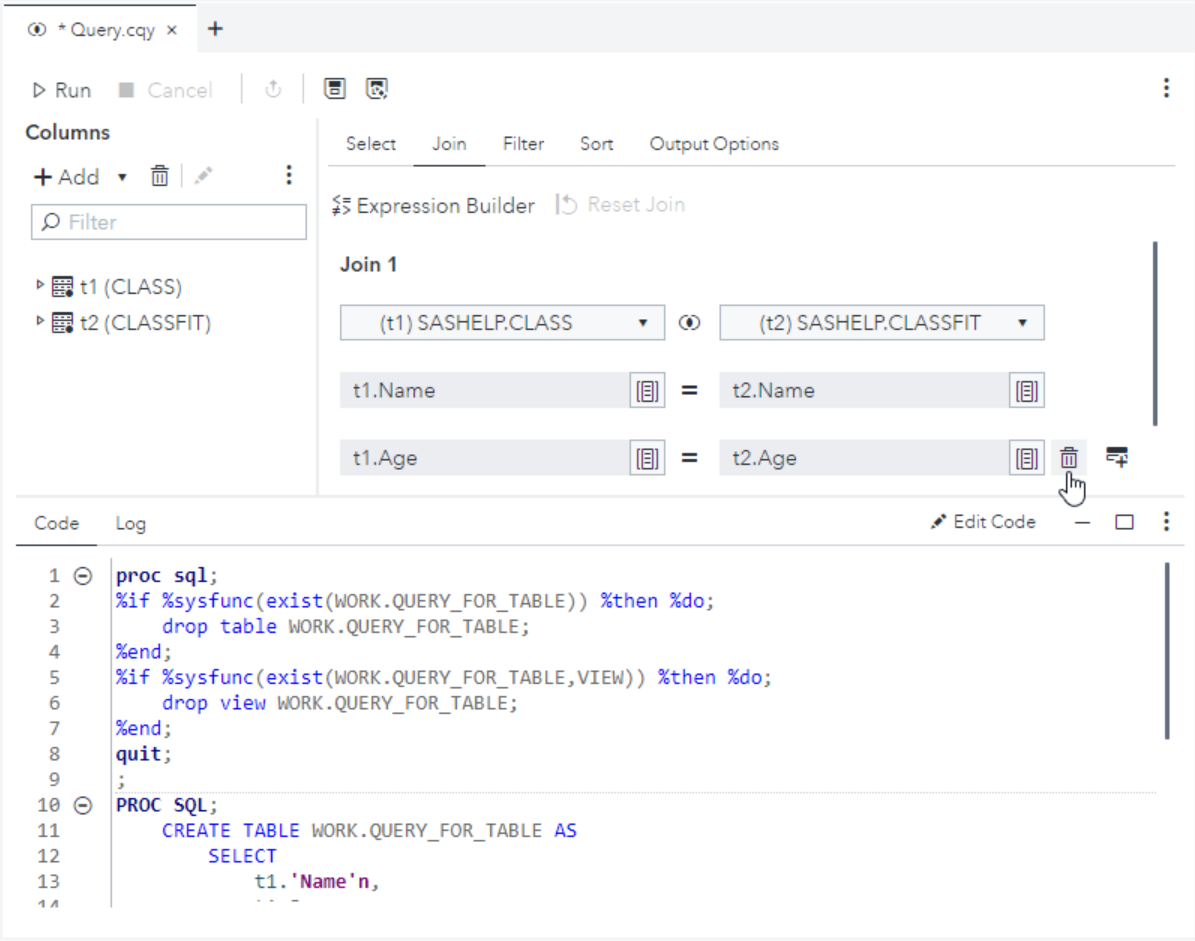
You can modify an existing join by selecting a different type of join or by changing the columns that are used in the join condition. You can also add and remove join conditions or remove the entire join.

Note: You must have at least one join and one join condition for each additional table that you add to the query.

To modify a join:

1. Click the **Join** tab of the query window. You can edit the join in the following ways:
   - Use the table drop-down lists to change the tables that are used in the join.
   - Click the join indicator between the table names to edit the join type. The default join type is `Inner join`. For more information, see “Understanding the Types of Joins” on page 76.
   - Click the operator between the column names to select a new operator. The default operator is =.
   - Click on each side of the join to change the columns that are used in the join.
2. To add a new join condition, position the mouse pointer over the existing join condition, and click and specify the columns to use in the join. To remove a join condition, position the mouse pointer over the join condition, and click next to the appropriate condition.

   Note: If the join has only one join condition, you cannot delete the join condition.



Note: To edit and create joins using SQL code, click **Edit Expression** on the **Join** tab. For more information, see "Building an Expression" on page 273.

Selecting Data

Specifying Columns in the Output

By default, no columns are included in the output. You must specify the columns that you want to appear in the output table. You can also specify an alias to use in place of the column name in the output table.

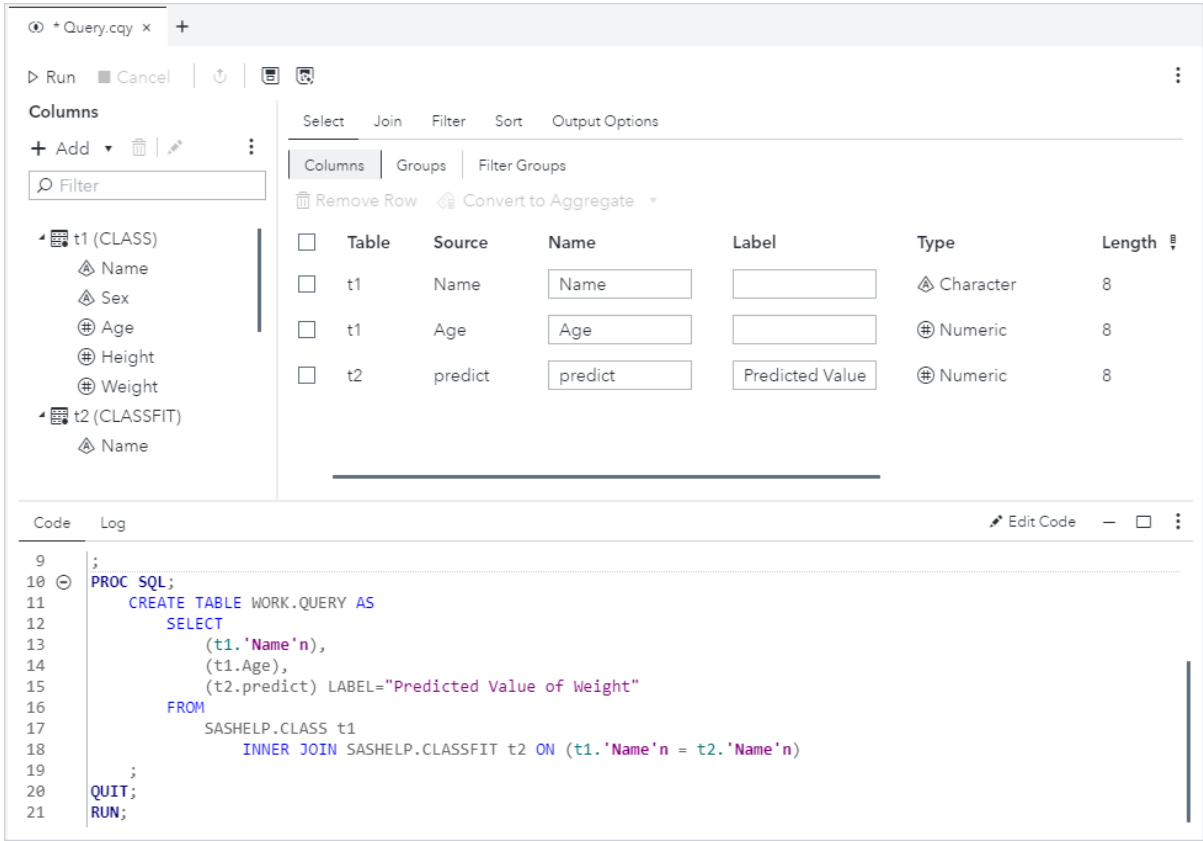
The order in which the columns are listed on the **Select** tab is the order in which they appear in the output table. You can move columns up, down, and to the top or bottom of the list by right-clicking a column and selecting the appropriate option. You can move multiple adjacent columns in the list by selecting the columns that you want to move, right-clicking one of the columns, and then selecting the appropriate option.

Note: The Length, Label, and Informat columns are not available on the **Columns** subtab of the **Select** tab if you are generating a FedSQL query. The Format column is not available if you are generating a FedSQL query in a flow. If you are generating a SQL query, you can manage the columns that are displayed on the **Columns** subtab by clicking . Use the Manage Columns window to hide and display columns.

You can select columns individually for the output, or you can use different options to select all columns in the input tables.

To select individual columns for the output table:

- In the Columns area of the Query tab, expand the tables and drag the columns that you want to include in the output to the **Columns** subtab of the **Select** tab.



To select all columns from a table by using column names:

- In the Columns area of the Query tab, right-click the table and select **Add columns**. This option creates a SELECT clause that includes all individual column names:

  `SELECT table-alias.column-name1 <, table-alias.column-name2,...>`

To select all columns from a table by using the asterisk (*) notation:

- In the Columns area of the Query tab, right-click the table and select **Select all columns**. This option creates a SELECT clause that uses the asterisk notation (*) to represent all columns from the table:

  `SELECT table-alias.*`

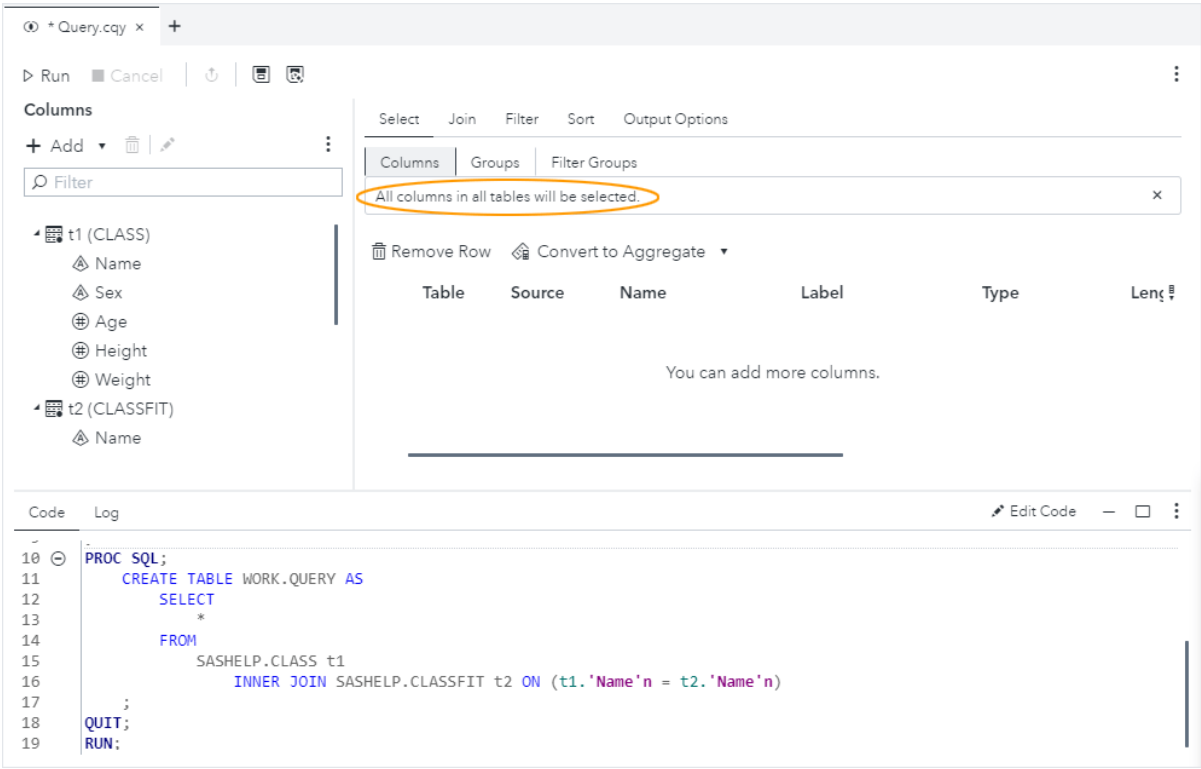
Note: This option is useful if the number of columns in your input table might change.

To select all columns from all tables by using the asterisk (*) notation:

- Click ⋮ in the Columns area of the Query tab and select **Select all columns in all tables**. This option creates a SELECT clause that uses the asterisk (*) notation to represent all columns and all tables:

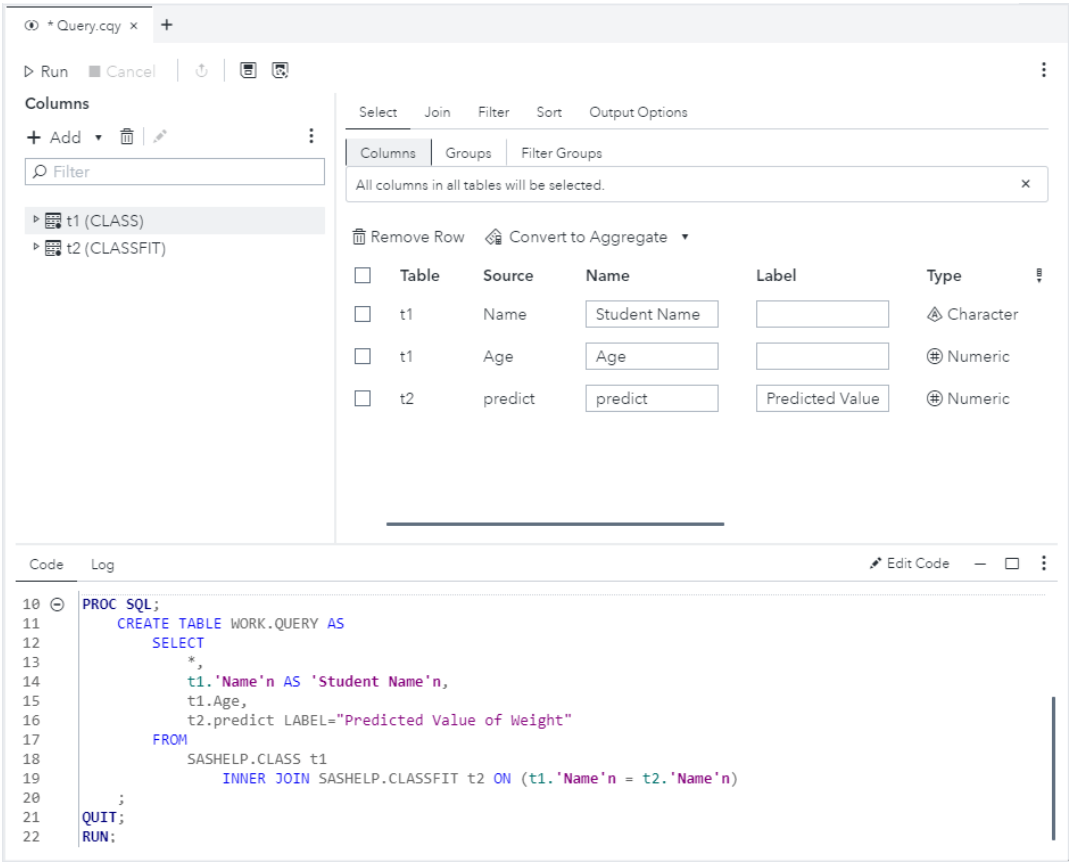
```
SELECT *
```

Note: This option is useful if the number of tables in the query might change or the number of columns in a table might change.



To specify an alias for a column:

- On the **Select** tab, use the Name column to specify any aliases that you want to use for the columns. The alias is used as the column heading for the output data.



To remove columns from the output:

- On the **Select** tab, select the check boxes for the columns that you want to remove and click **Remove Row**.

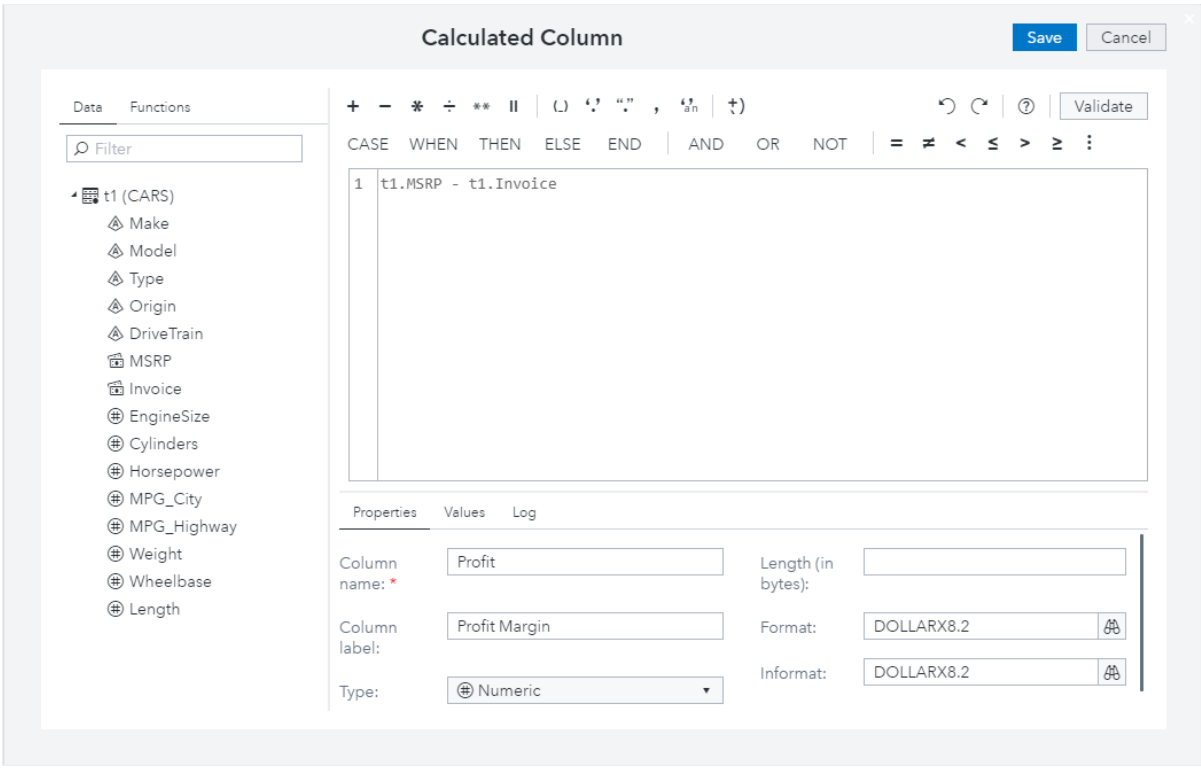
  To remove all columns from the query, select the check box for the table on the **Select** tab and click **Remove Row**.

Creating a Calculated Column

You can insert a new column into your query that is calculated from other columns or values. You can use a calculated column to summarize values, replace values based on a condition, and perform string and numeric calculations. After you create a calculated column, you can use the column to filter, sort, or group your output.

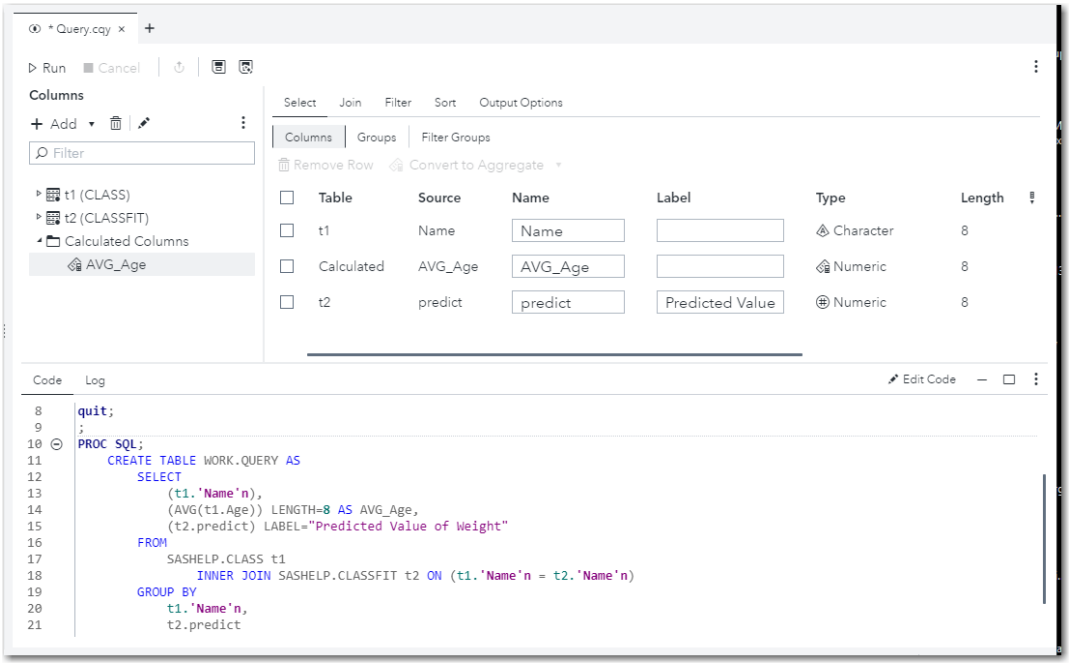
To create a calculated column:

1. In the Columns area of the Query tab, click **Add** ⇨ **Calculated column**.
2. Use the expression builder in the Calculated Column window to create your column. For more information, see “Building an Expression” on page 273.
3. After you create the expression for the column, use the **Properties** tab in the expression builder area to specify a name for the column. You can also specify a label, data type, length, format, and informat.
4. Click **OK** to create the calculated column.

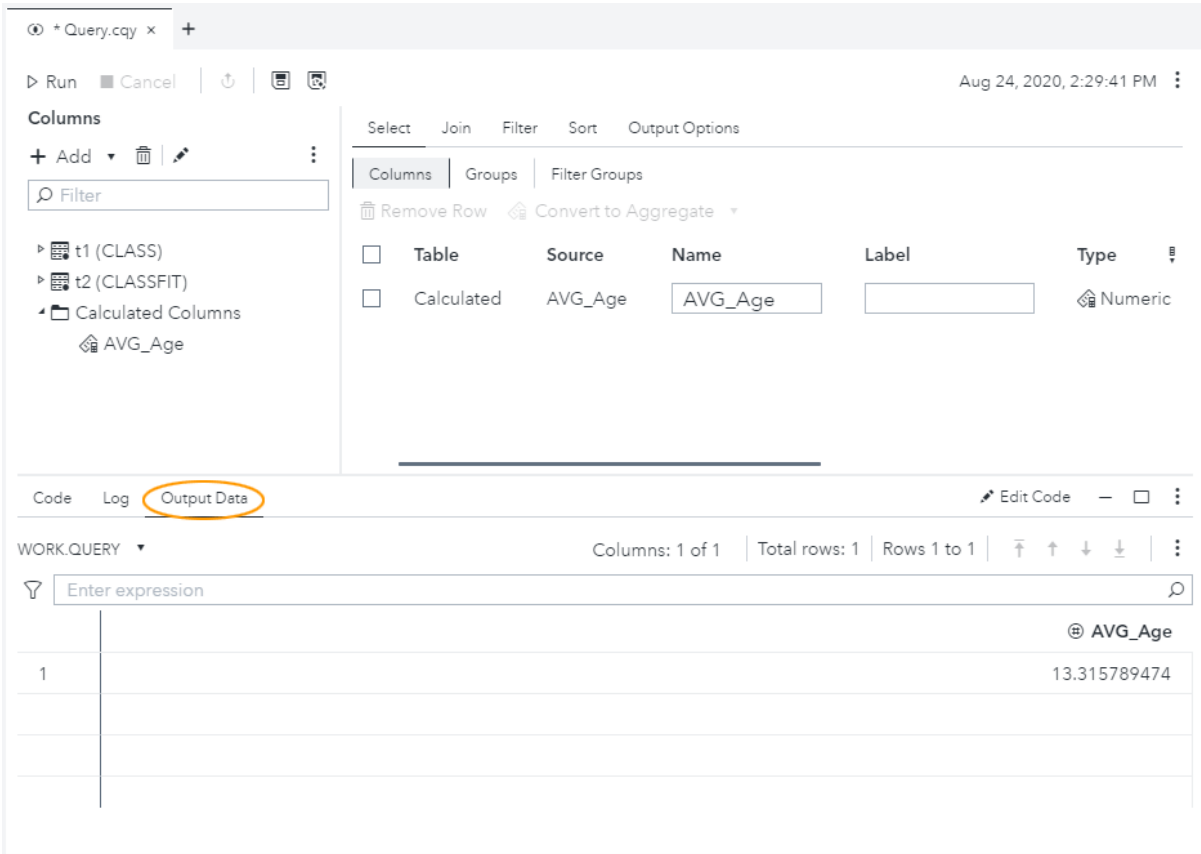


Using Summary Functions

You can perform aggregation, or summary, functions on any of the columns in your query. When you aggregate a column, a new calculated column is automatically created and added to your query. To perform an aggregation function, select the column on which you want to perform an aggregation function. Click **Convert to Aggregate** and select the function that you want to use. The following example shows the Age column after it has been converted to a calculated column that calculates the average age of all of the students:



By default, the query displays the results on the **Output Data** tab and generates an output table in the Work library.



By default, when you aggregate a column, your output is grouped by all of the columns without aggregation. For more information, see “Grouping Your Output” on page 90.

Working with Conditional Expressions

Creating a Conditional Expression

You can create a conditional expression in your query by creating a calculated column that uses the CASE function. A conditional expression enables you to use logic similar to "IF-THEN/ELSE" to return values that can be used to create a column.

Note: You cannot create conditional expressions if you are generating a FedSQL query. For more information, see “Generating a FedSQL Query” on page 98.

For example, you could create an expression to extract the first letter of a last name and replace last names starting with "A" through "F" with the value "A - F," replace last names starting with "G" through "L" with the value "G - L," and so on.

To create a conditional expression:

1. In the Columns area of the Query tab, click **Add** ⇨ **Calculated column**.
2. Use the following template for the CASE expression syntax to enter your own CASE expression in the expression box:

```
CASE <CaseOperand>
     WHEN <whenCondition> THEN <resultExpression>
     WHEN <whenCondition> THEN <resultExpression>
     ELSE <resultExpression>
END
```

   - **<CaseOperand>** – the name of the column that you are using in the expression.
   - **<whenCondition>** – the existing value of the column.
   - **<resultExpression>** – the value that you want to assign to the column.
3. After you create the expression for the column, use the **Properties** tab in the lower half of the expression builder window to specify a name for the column. You can also specify a label, data type, length, format, and informat.

Note: For more information, see “CASE Expression” in *SAS SQL Procedure User’s Guide*.

Sample Conditional Expressions

The following example uses the SUBSTR function to extract the first letter of the last name, and then assign a replacement value based on that letter.

```
CASE WHEN (SUBSTR(tablename.LastName,1,1) between "A" and "F") THEN "A - F"
     WHEN (SUBSTR(tablename.LastName,1,1) between "G" and "L") THEN "G - L"
     WHEN (SUBSTR(tablename.LastName,1,1) between "M" and "R") THEN "M - R"
     WHEN (SUBSTR(tablename.LastName,1,1) between "S" and "Z") THEN "S - Z"
END
```

The following example replaces age values with "Under 35," "35 - 50," and "Over 50."

```
CASE
     WHEN tablename.Age < 35 THEN "Under 35"
     WHEN tablename.Age between 35 and 50 THEN "35 - 50"
     ELSE "Over 50"
END
```

Filtering Data

Creating a Filter

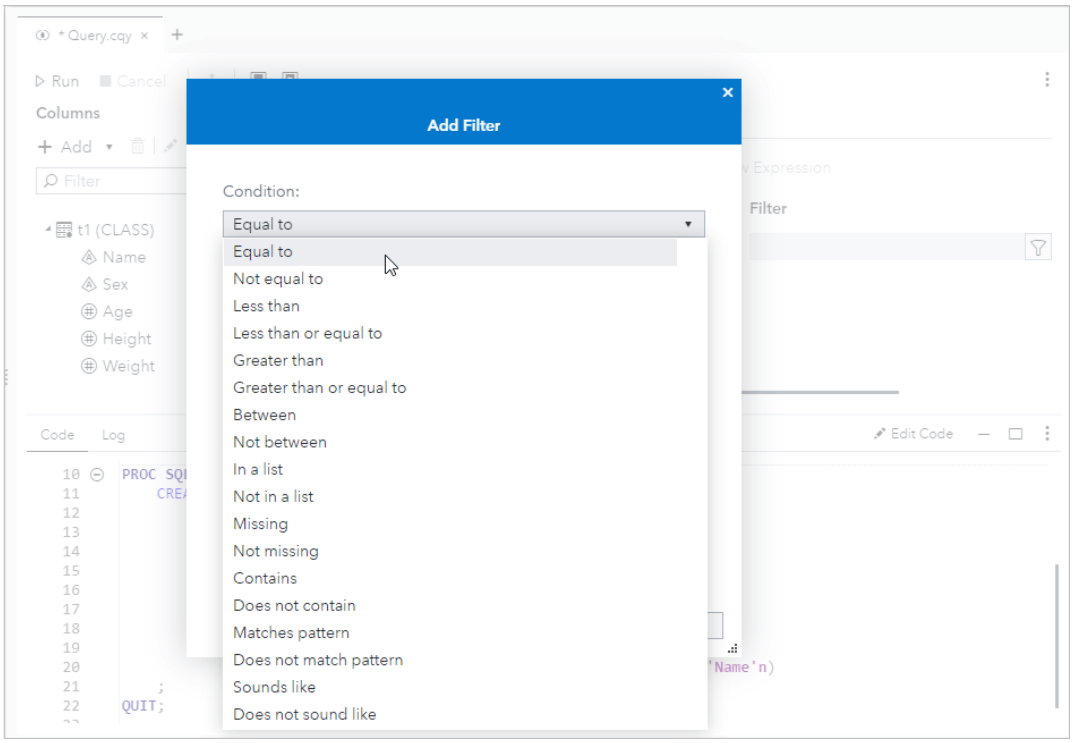
When you query data, you might want to retrieve only rows that meet certain criteria, based on values of columns in the data. The process of telling SAS Studio which rows to retrieve is called setting a filter and is done on the **Filter** tab. This process corresponds to using a WHERE clause in an SQL query.

To use the expression builder to create a filter expression, click **Expression Builder** on the toolbar. For more information, see “Building an Expression” on page 273.

1 Click the **Filter** tab, and then drag one or more columns from the Columns area.

Note: It is possible to filter the output table by columns that are not selected for the output.

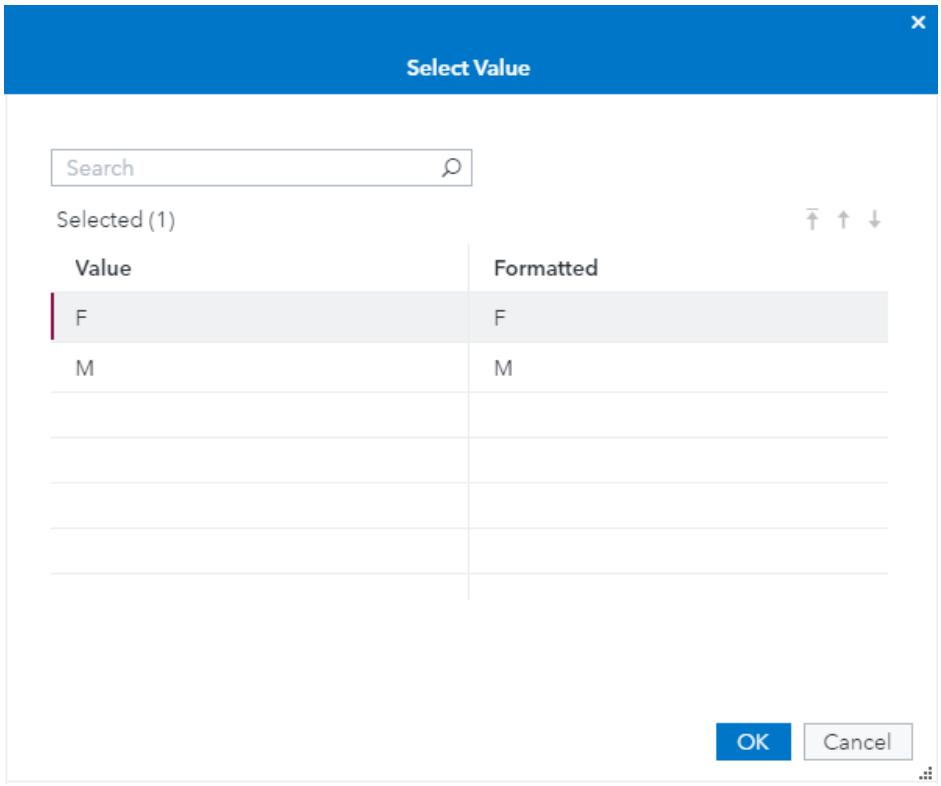
2 Click  beside the column for which you want to specify filter criteria. In the filter window, select a comparison operator from the operator drop-down list. The default value is **`Equal to`**.



3 If the operator that you have selected requires a value, enter or select a value in the **Value** box. To choose from a list of values, click and click **Get Values** in the Select Value window. Select the values that you want to use and click **OK**.

Note: If you want to search for a value in the value window, use the unformatted value.

Note: If you are using the `In a list` or `Not in a list` operators, click to enter values, or click + to choose from a list of values.



4 Depending on the data type of the column that you are using in the filter, you can choose from among the following options:

- **Match case** – retrieves only rows that match the capitalization of the value that you specify. If this option is not selected, the UPPER function is applied to the expression. This option is not selected by default.
- **Quote string** – encloses values in single quotation marks. This option is selected by default. If you are using a macro variable or other value that is evaluated when the filter is run, you should clear this option.
- **Allow macros** – enables you to enter character values in a filter on a numeric column.
- **Use raw values** – uses unformatted numeric, date, time, and datetime values instead of the formatted values. This option is selected by default and is available only for numeric, date, time, and datetime columns. If you want to create a filter based on formatted values, clear this option.

  Note: Some formats cannot be used in a filter that is based on formatted values if you are generating a FedSQL query.

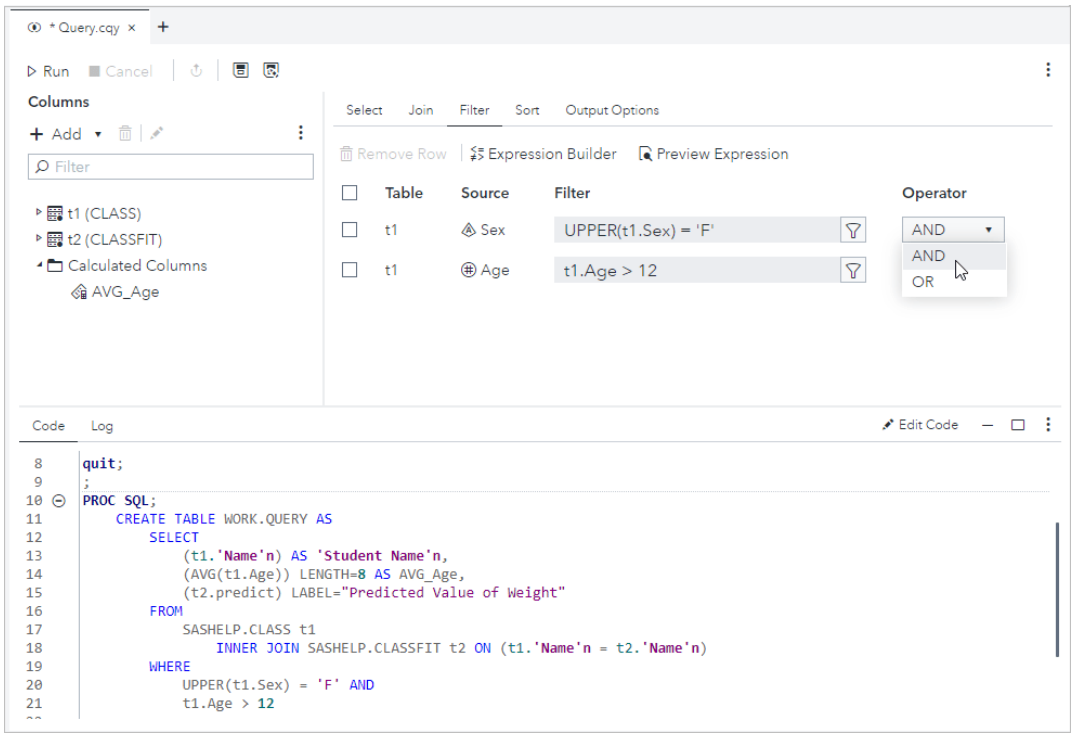
5 Click **Filter** to add the values to the filter.

Changing the Relationship between Filter Expressions

You can use only one column in a filter expression, but you can use multiple columns to create several expressions. If you create more than one comparison expression in your filter, the default relationship between these filter elements is AND. You can change the relationship between filter elements from AND to OR.

To change the relationship between filters:

- On the **Filter** tab, click the **Operator** value and select a new value.



If you are filtering the data by multiple columns, the table is filtered first by the column that is listed first. You can move columns up, down, and to the top or bottom of the list by right-clicking a column and selecting the appropriate option. You can move multiple adjacent columns in the list by selecting the columns that you want to move, right-clicking one of the columns, and then selecting the appropriate option.

Managing Output

Sorting Your Output

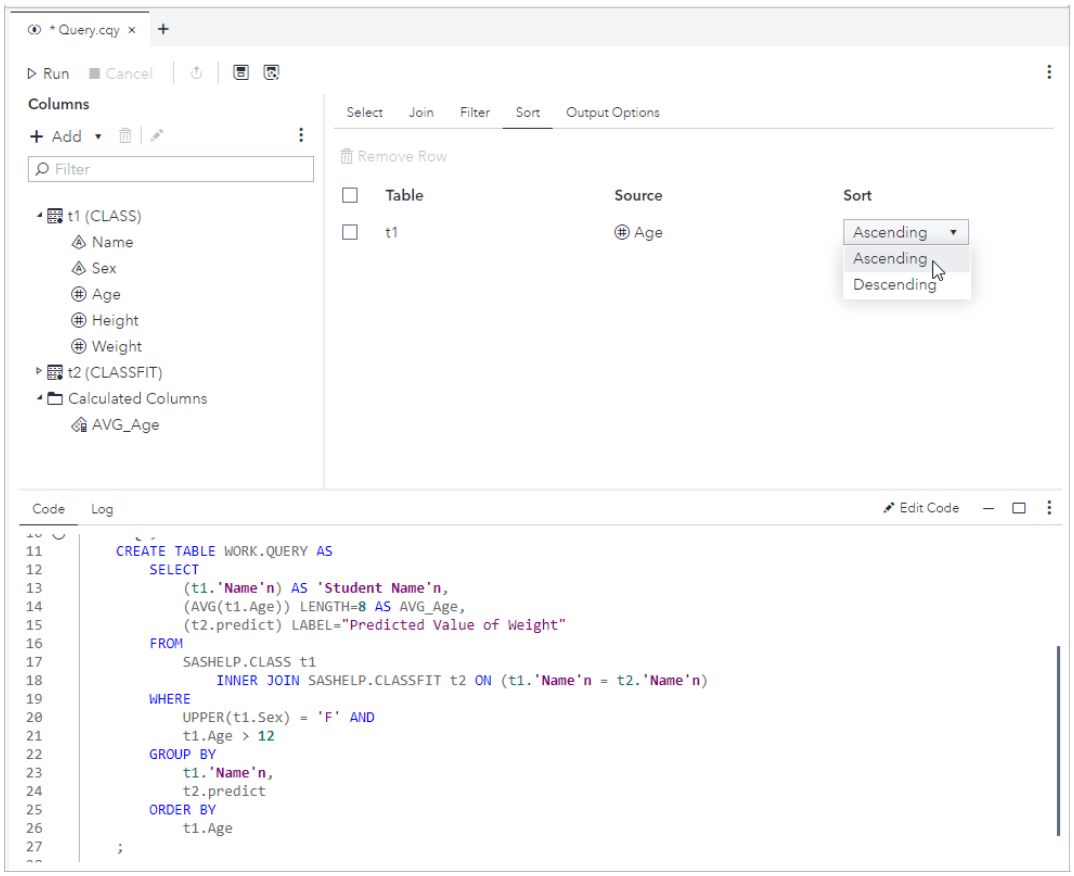
You can sort the output from your query by one or more columns from the tables that are used in the query.

Note: It is possible to sort the output table by columns that are not selected for the output.

Note: You cannot sort the output from your query if you are generating a FedSQL query using CAS data. For more information, see “Generating a FedSQL Query” on page 98.

To sort your output:

1. Click the **Sort** tab and drag one or more columns from the Columns area.
2. Click the **Sort** box for the column on which you want to sort the data. From the drop-down list, select **Ascending** or **Descending**. The default sort direction is **Ascending**.



3 If you are sorting by multiple columns, the output table is sorted first by the column that is listed first. Within each level of the first column, the rows are sorted by the second column in the list, and so on. You can move columns up, down, and to the top or bottom of the list by right-clicking a column and selecting the appropriate option. You can move multiple adjacent columns in the list by selecting the columns that you want to move, right-clicking one of the columns, and then selecting the appropriate option.

Eliminating Duplicate Rows in Output

Some types of queries generate multiple, identical rows. Because these duplicate rows are generally not useful, SAS Studio enables you to keep only one of the identical rows and eliminate the duplicates.

To eliminate duplicate rows, click ⋮ in the Columns area and select the **Select distinct rows only** check box.

Grouping Your Output

If you have created a summarized column, you can choose to classify your data into groups based on the values in a column. This action is equivalent to using the GROUP BY clause in an SQL query. For example, if you are calculating the average

height of a group of students, you might want to group the results by age so that you can see the average height for each age group.

By default, the **Automatically group data** option is selected on the **Groups** subtab of the **Select** tab. When this option is selected and you have performed a summary function on a column, your query is automatically grouped by all columns without summary functions. You can choose to edit the list of columns that the query is grouped by. You can add or remove columns from the list, and you can clear the **Automatically group data** option to remove all columns from the list. To return to the default grouping selection, select the **Automatically group data** option.

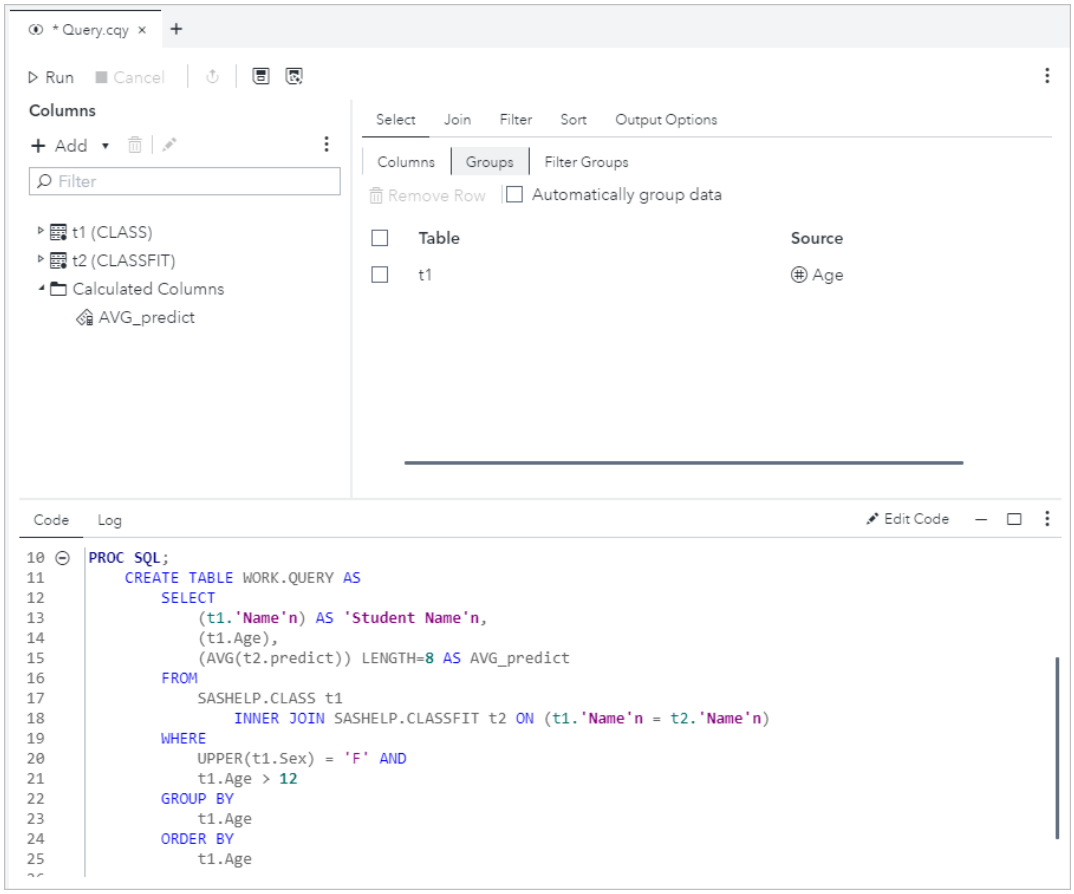
Note: The **Automatically group data** option includes only columns that are explicitly added to the query using column names and does not include any columns that are added to the query by using the asterisk (*) notation. For more information, see "Specifying Columns in the Output" on page 78.

To group your output:

1 Click the **Groups** subtab of the **Select** tab and drag one or more columns from the Columns area.

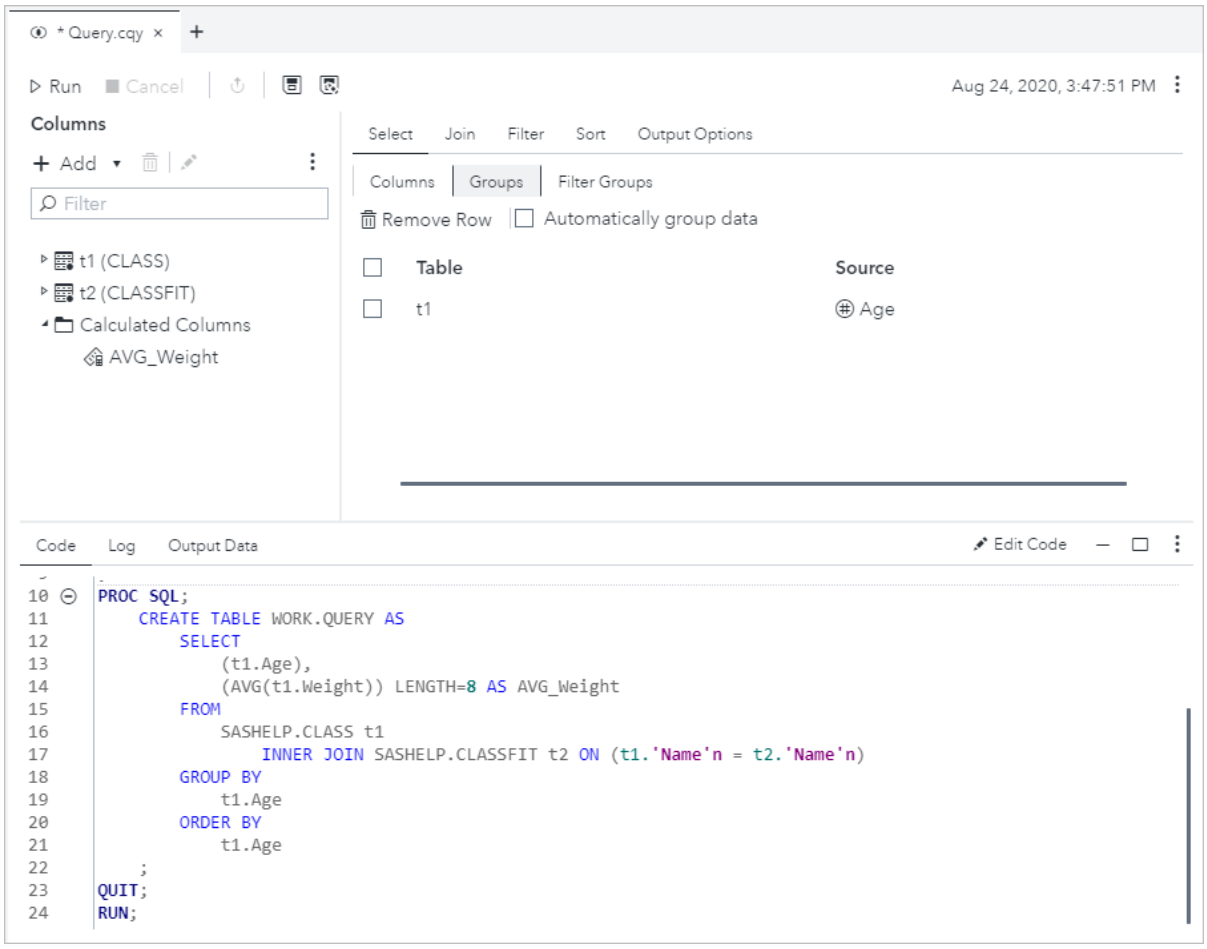
Note: It is possible to group your output by columns that are not selected for the output.

Note: When you add columns to or remove columns from the **Groups** subtab, the **Automatically group data** option is automatically cleared. To remove the columns that you have added to the **Groups** subtab, select the **Automatically group data** option again. You can also remove individual columns by selecting the column that you want to remove and clicking **Remove Row**.

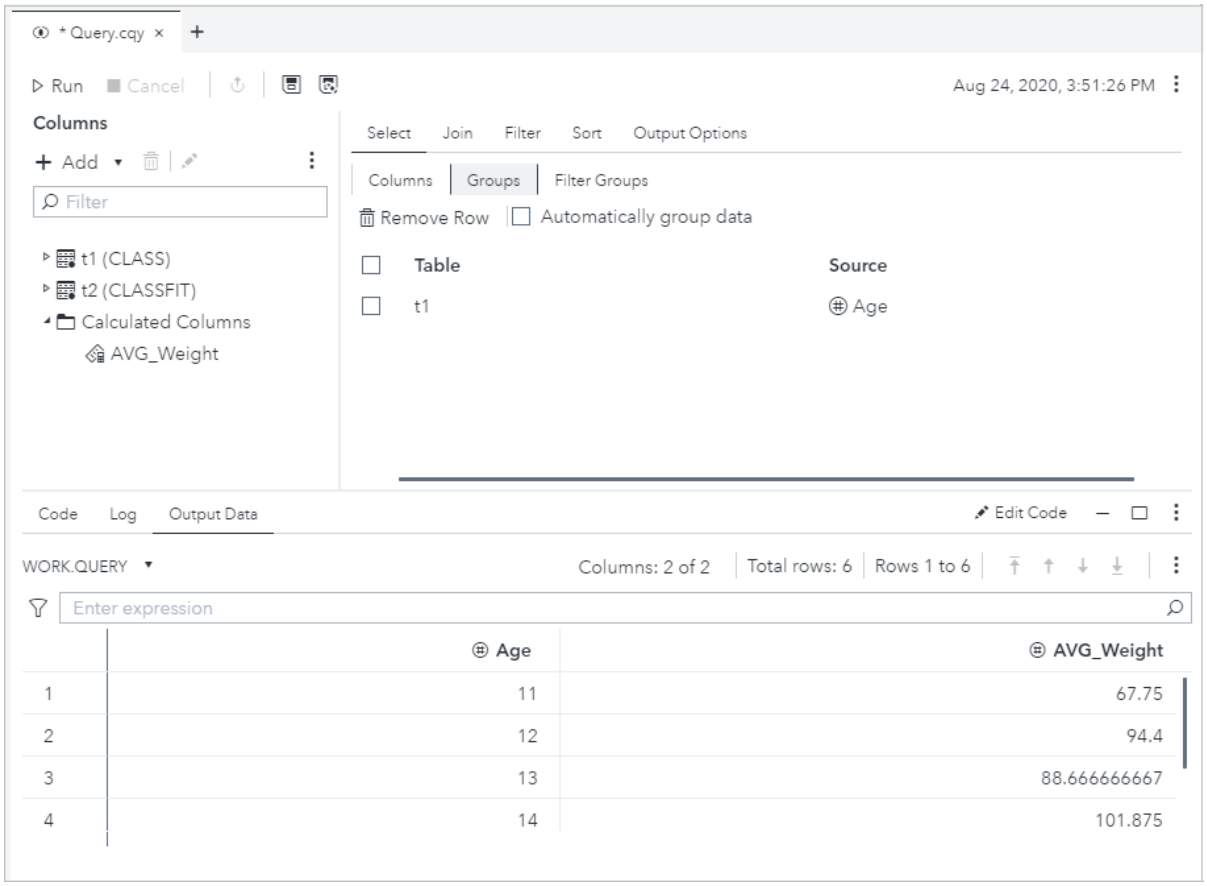


2 If you are grouping the data by multiple columns, the table is grouped first by the column that is listed first. You can move columns up, down, and to the top or bottom of the list by right-clicking a column and selecting the appropriate option. You can move multiple adjacent columns in the list by selecting the columns that you want to move, right-clicking one of the columns, and then selecting the appropriate option.

The following example shows you how to find the average weight of students in each age group. First, add the Age and Weight columns to the **Select** tab, and then convert the Weight column to a calculated column using the AVG summary function.



To see the average weight of students by age, the query is grouped by the Age column. The results show the average weight for each age group.



Note: By default, the query generates a table of the results. To generate a report of the results (which is displayed on the **Results** tab), you must specify `Report` as the output type for the query. For more information, see "Saving Your Results" on page 95.

Subsetting Grouped Data

If you have created a summarized column or a calculated column that contains a summary function, you can choose to classify your data into groups based on the values in a column. When you group your output by one or more columns, you can create a filter to subset the grouped data based on specified conditions. This action is equivalent to using the HAVING expression in an SQL query.

Note: You can also choose to filter the raw, unsummarized data. For more information, see "Creating a Filter" on page 85.

You can use one column in a filter, or you can use multiple columns to create several comparison expressions. If you create more than one comparison expression in your filter, the default relationship between these filter elements is AND. You can change the relationship between filter elements from AND to OR, and you can group elements together.

You can choose to create either a basic filter or you can create an advanced filter by using the expression builder:

1 Click the **Filter Groups** subtab of the **Select** tab and drag one or more columns from the Columns area.

Note: It is possible to subset your grouped data by columns that are not selected for the output.

2 Click  beside the column for which you want to specify filter criteria. In the filter window, select a comparison operator from the **Condition** drop-down list. The default value is `Equal`.

3 If the operator that you have selected requires a value, enter or select a value in the **Value** box. To choose from a list of values, click and click **Get Values** in the Select Value window. Select the values that you want to use and click **OK**.

Note: If you want to search for a value in the value window, use the unformatted value.

4 Depending on the data type of the column that you are using in the filter, you can choose from among the following options:

- **Match Case** – retrieves only rows that match the capitalization of the value that you specify.
- **Quote Strings** – encloses values in single quotation marks. This option is selected by default. If you are using a macro variable or other value that is evaluated when the filter is run, you should clear this option.
- **Allow macros** – enables you to enter character values in a filter on a numeric column.

5 Click **Filter** to add the filter expression to the query.

Note: If you create more than one filter expression, you can change the relationship between the expressions. For more information, see "Changing the Relationship between Filter Expressions" on page 88.

Saving Your Results

You can choose to generate your results in any one of three formats: report, data table, or data view.

If you save your results as a data table or data view, you can specify the library and table name that you want to use. If you do not specify the library and table name, the results are saved in the Work library.

To specify the results format:

1 In the query window, click the **Output Options** tab.

2 Select the format that you want to use from the **Output type** drop-down list.

Report
: saves the query results as a report that you can download as an HTML, PDF, or RTF file. Query results in this format are not updated until you rerun the query. You cannot run tasks against query results in this format.

Table
: saves the query results as a static data table against which you can run tasks. Query results in this format are not updated until you rerun the query. By default, the data table is stored in the Work library.

View
: saves the query results as a dynamic data view against which you can run tasks. Each time you open query results in the data view format, the results are updated with any changes to the data that is used in the query. By default, the data view is stored in the Work library.

Note: This option is not available if you are generating a FedSQL query. For more information, see "Generating a FedSQL Query" on page 98.

To save your output data to a specific location:

1. On the Query tab, click the **Output Options** tab.
2. If you are generating a FedSQL query using CAS tables, you must use the **Session reference** option to specify the name of the default CAS session to use. For more information, see "Generating a FedSQL Query" on page 98.
3. Enter the name of the library in which you want to save your results in the **Library** box.
4. To specify a name for the output data, enter the name that you want to use in the **Table name** box.

Specifying Limits for Input Processing

When you create a query, you can restrict the number of rows that your query processes from each source table. This option corresponds to the INOBS= option in PROC SQL and is useful when you want to reduce processing time in order to test a query.

Note: This option is not available if you are generating a FedSQL query. For more information, see "Generating a FedSQL Query" on page 98.

When you set a limit *n* on the number of rows to process, SAS Studio uses the first *n* rows from each source table in the query. If you have added a filter to your query, SAS Studio uses the first *n* rows from each table that match your filter criteria.

This option does not apply when the output format is a data view. For more information, see "Saving Your Results" on page 95.

Note: If you have a computed column in your query data that performs a function on the whole table, such as sum or average, then limiting the number of input rows

affects the value that is calculated for that column. The function is performed only on the rows that are processed.

To limit the number of rows to process:

- Click the **Output Options** tab. Select **Limit number of matching rows to process (INOBS)** and specify the number of rows that you want the query to process.

Note: When you limit the number of rows to process and you have joined tables, the maximum number of rows that the query can return is the Cartesian product of the number of rows from each table. For example, if you set the limit to five rows and you join two tables, the maximum number of rows that the query can return is 25 (5 * 5). If you join three tables, the maximum number of rows that the query can return is 125 (5 * 5 * 5).

Specifying Limits for Output Processing

When you create a query, you can restrict the number of rows that your query saves in the output. This option corresponds to the OUTOBS= option in PROC SQL and is useful when you want to restrict the size of your output for testing purposes.

Note: This option is not available if you are generating a FedSQL query. For more information, see "Generating a FedSQL Query" on page 98.

This option does not apply when the output format is a data view. For more information, see "Saving Your Results" on page 95.

Note: When you limit the size of your output, you are not reducing the processing time of the query. The query still processes all of the data in each source table. You are reducing only the size of the output.

- Click the **Output Options** tab. Select **Limit number of matching rows to save in output (OUTOBS)** and specify the number of rows that you want in your output.

Running a Query

After you specify all the criteria for your query, you can generate your results by clicking **Run** on the Query tab toolbar. The output data or results are added to the results area of the Query tab.

Note: You can reset the query to its initial state in which only the tables are specified and any selected columns, filters, sorting criteria, and output options are removed. To reset the query, click ⋮ > **Reset**.

Generating a FedSQL Query

By default, queries in SAS Studio use PROC SQL to generate results. However, you can choose to create your query using PROC FEDSQL instead of PROC SQL. PROC SQL is designed for queries that use data on your SAS server or data from a single database. PROC FEDSQL is designed for federated queries, which can access tables from multiple database connections and return a single result set.

PROC FEDSQL provides many benefits if you are working in an environment in which you need more features than are provided in the SQL procedure:

- FedSQL conforms to the ANSI SQL:1999 core standard.
- FedSQL supports many more data types than are available in SAS SQL implementations, including BIGINT, DATE, INTEGER, and VARCHAR.
- FEDSQL queries can run on a CAS server.
- FedSQL handles federated queries and can join tables from multiple database connections.
- FedSQL language can create data in any of the supported data sources, even if the target data source is not represented in a query.

For more information, see Overview: PROC FEDSQL.

The data types that are supported by PROC SQL and PROC FEDSQL are listed in the following table:

PROC SQL	PROC FEDSQL
CHARACTER	CHARACTER
DATE*	DATE
DECIMAL*	DECIMAL*
DOUBLE	DOUBLE
FLOAT*	FLOAT*
INTEGER*	INTEGER
NUMERIC	NUMERIC
REAL*	REAL*
SMALLINT*	SMALLINT*
	BIGINT

PROC SQL	PROC FEDSQL
	VARCHAR

Note: * When you specify these data types, they are converted to either the CHARACTER or DOUBLE (NUMERIC) data type.

By default, queries are generated using PROC SQL. You can change the default procedure for all queries to PROC FEDSQL, and you can also change the procedure that is used for an individual query.

- To automatically use PROC FEDSQL when you create a new query, select **Options** ⇨ **Preferences** and click **Query**. Select **PROC FEDSQL**.

  Note: This option is applied only to queries that you create after you have selected the option.

- To specify the procedure that is used in an individual query, click the **Output Options** tab in your query. Use the **Generate code using** option to specify either **PROC SQL** or **PROC FEDSQL** before you begin creating your query. If you change this option after you have started creating your query, your query might contain errors.

Note: If you are generating a FedSQL query using CAS tables, you must use the **Session reference** option on the **Output Options** tab to specify the name of the default CAS session to use.

The query functionality for creating SQL and FedSQL queries is very similar. The following table lists some of the differences between SQL and FedSQL queries:

Query Feature	Available in SQL?	Available in FedSQL?
Output type – View	Yes.	No.
Output data – CAS table*	Yes, if the input data source is one or more SAS tables and DBMS data sets.	Yes, if the input data source is one or more CAS tables.
Select tab: Length, Label, and Informat columns	Yes.	No. In addition, the Format column is available only for stand-alone queries.
Sorting output	Yes.	Yes, if the query does not use CAS data. No, if the query uses CAS data.

Query Feature	Available in SQL?	Available in FedSQL?
Output Options – **Session reference**	No.	Yes.
Output Options: **Limit number of matching rows to process**, **Limit number of matching rows to save in output**	Yes.	No.
Operators: Contains, Does not contain, Sounds like, Does not sound like, Any, All	Yes.	No.
Functions: CASE, CAT, CATS, CATX, COMPBL, COMPRESS, FIND, INPUT, INT, INTCK, INTNX, LEFT, LENGTH, MDY, N, NLDATE, NLDATM, PROPCASE, SCAN, SUBSTR, TRANWRD, WEEK, YRDIF	Yes.	No.
Aggregations: CV, FREQ, MEAN, N, PRT, STD, SUMWGT, T, VAR	Yes.	No. The following aggregations are available only in FedSQL: ■ PROBT (replaces PRT) ■ STDDEV (replaces STD) ■ STUDENT_T (replaces T) ■ VARIANCE (replaces VAR)

* If the output from a FedSQL query is a CAS table, then the input data source must also be CAS tables. If the output from a FedSQL query is a SAS table or a DBMS table, then the input must be SAS tables and DBMS tables.

4

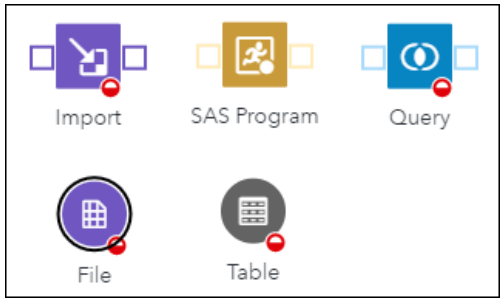
Working with Flows

What Is a Flow?

A *flow* is a sequence of operations on data. Data and operations are represented by nodes. A flow orchestrates nodes in a series of steps in which the output of one node is the input to another node. As you build a flow, SAS Studio automatically generates SAS code for each node. You can use flows to prepare data for reporting and analysis.

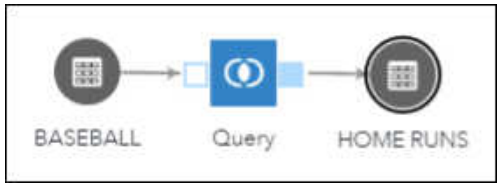
Data and operations in a flow are represented by nodes.

Figure 4.1 *Example of Flow Nodes*



For example, this flow shows the number of home runs for each player in a table of baseball data.

Figure 4.2 *Flow That Shows Home Run Data*

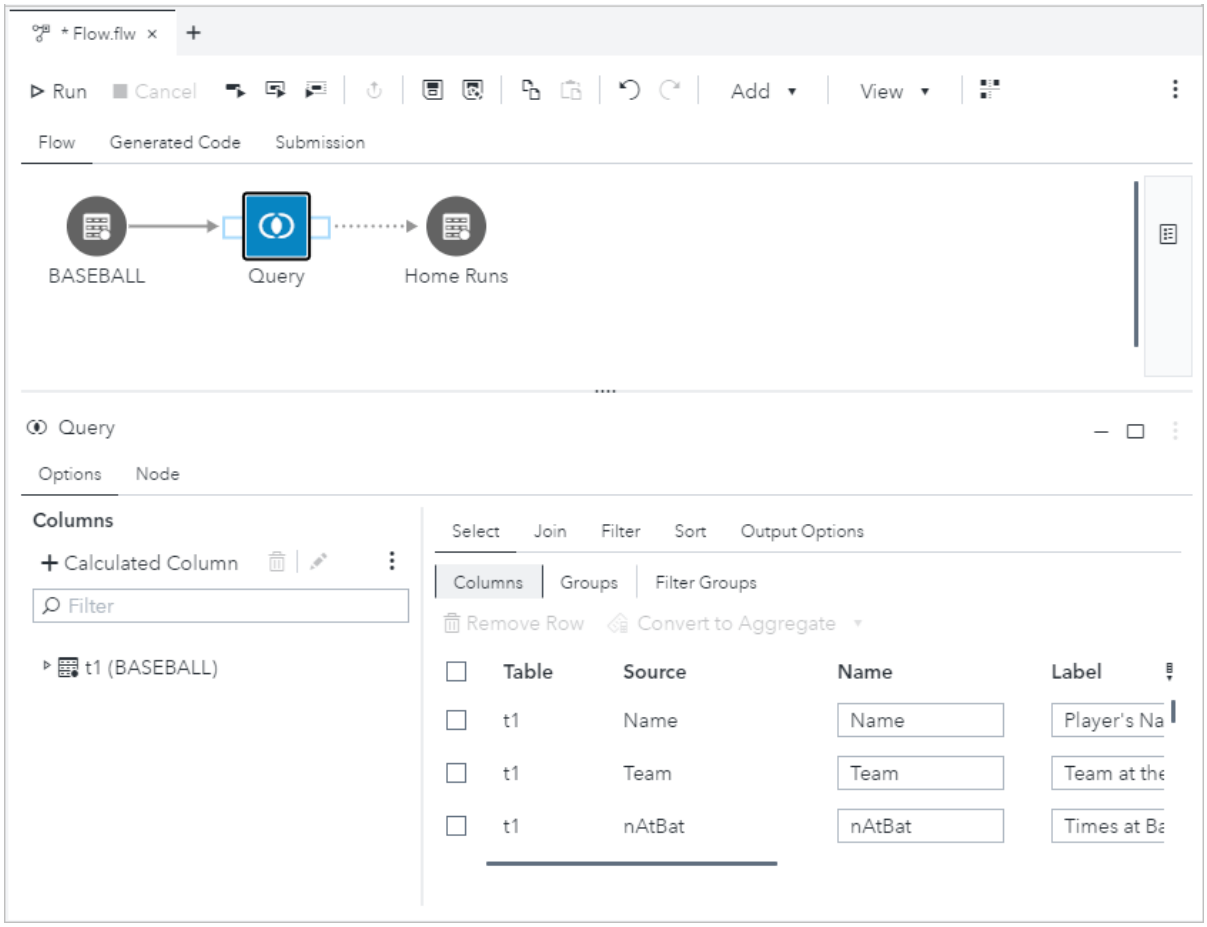


- BASEBALL is a Table node that specifies an input table of data about baseball teams.
- Query is a Query node that selects the number of home runs for each player in the BASEBALL table.
- HOME RUNS is a Table node that specifies the output table for the query.

Flow nodes can be accessed from the **Steps** section of the navigation pane. Only nodes in that section can be used in a flow.

Understanding the Flow Tab

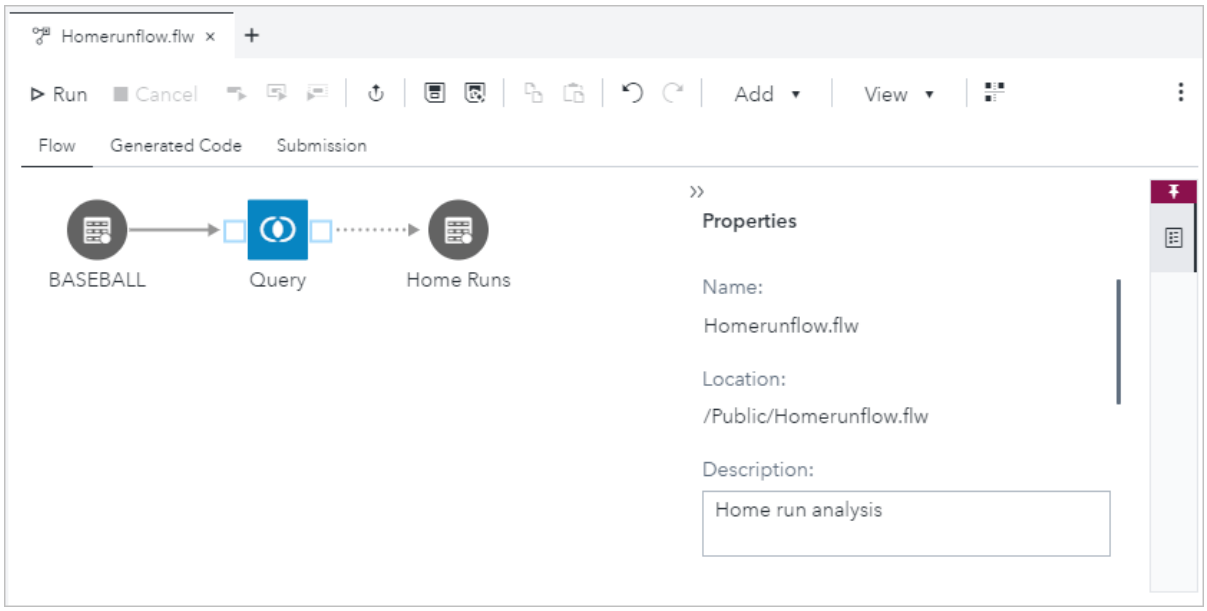
The flow functionality is available from the **Flow** tab.



The Flow tab has two areas:

- The flow canvas enables you to build the sequence of nodes and manage the flow as a whole. You can also use the **Generated Code** and **Submission** tabs to view the code and log that SAS Studio automatically generates as you build the flow. The **Submission** tab also displays any results and output data that are generated when you run a node.
- The node details under the canvas enable you to manage the attributes of a selected node in the flow, including node options and node properties. If you select a node on the flow canvas, the details for that node appear under the flow. You can specify various options that define the behavior of the selected node. You must specify a minimum set of options for each node in the flow in order to run the flow successfully.

By default, the flow properties are collapsed on the right side of the canvas. When you expand the Properties pane, you can edit the Description property to specify information about the flow.



You can perform the following actions on the Properties pane:

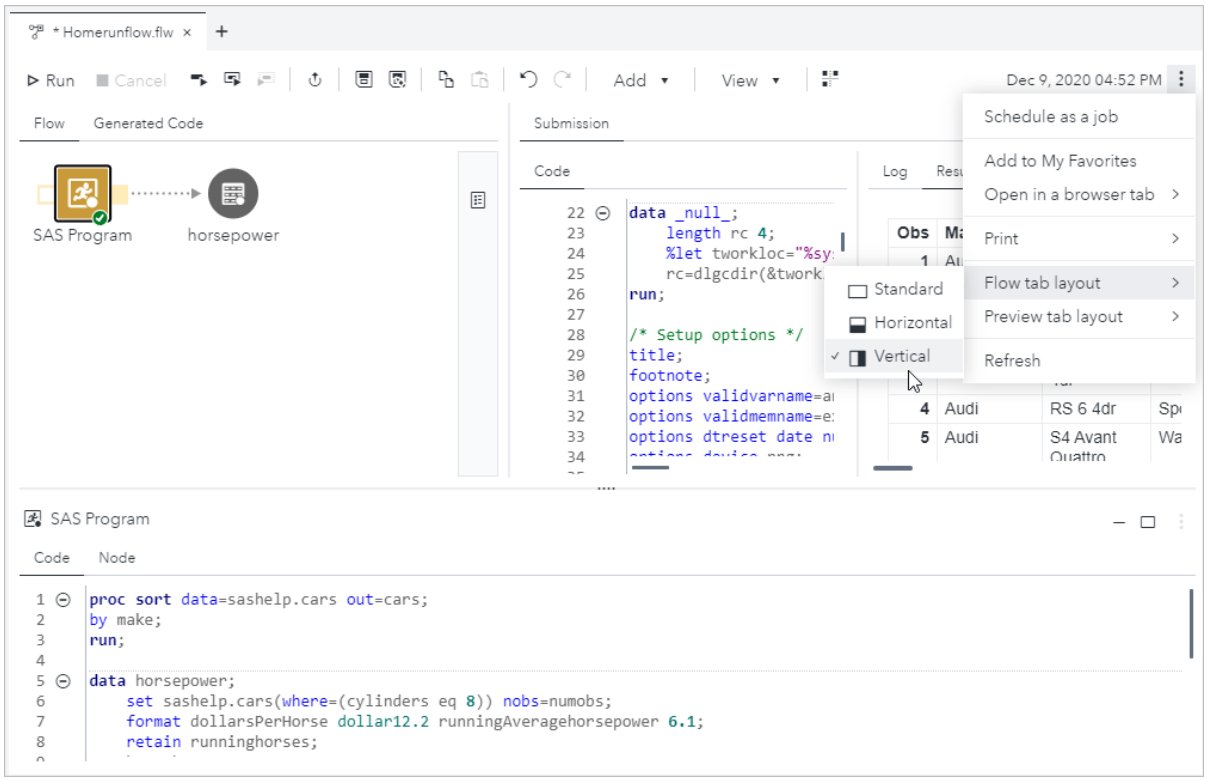
- To display the Properties pane when it is collapsed, click .
- To collapse the Properties pane, click .
- To pin the Properties pane so that it remains visible when you scroll the flow horizontally, click . The Properties pane is pinned by default when it is displayed. If you do not pin the Properties pane, some nodes on the right side of the flow canvas might not be visible.

Customizing the Flow Tab

You can change the layout of the default flow tabs so that the **Submission** tab is displayed separately from the **Flow** and **Generated Code** tabs. You can also choose to display the preview tabs horizontally or vertically.

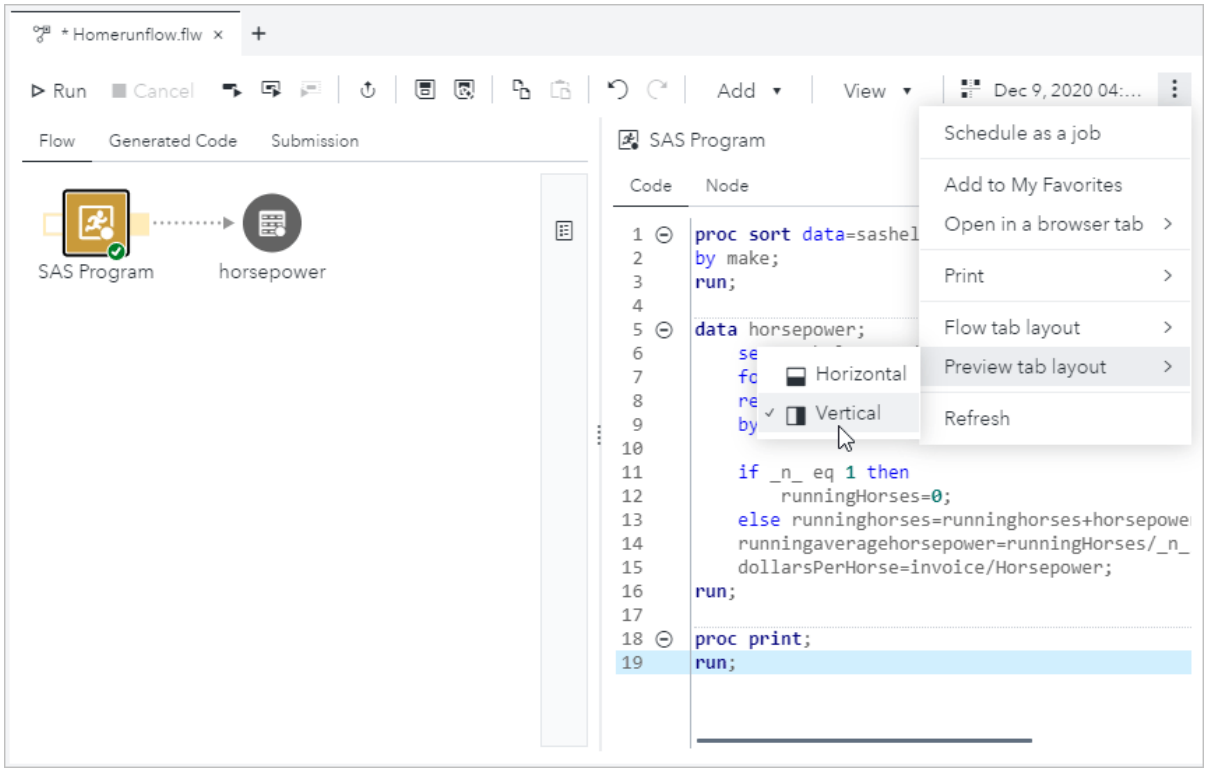
To change the layout of the default flow tabs:

- On the flow toolbar, click and select **Flow tab layout**. Select the layout that you want to use. The default layout is `Standard`.



To change the layout of the preview tabs:

- On the flow toolbar, click ⋮ and select **Preview tab layout**. Select the layout that you want to use. The default layout is `Horizontal`.



Creating a Flow

You can create a flow in these ways:

- On the **Start Page** tab, click **Build a flow**.
- From the SAS Studio menu, select **New** ⇨ **Flow**.

Opening a Flow

Flows can be saved to SAS Content or to a SAS Compute Server folder in the **Explorer** section of the navigation pane. Flows are saved as *.flw files. To open a saved flow, navigate to the appropriate folder and perform one of these actions:

- Double-click a flow to open it.
- Right-click a flow and select **Open**.
- Select a flow and drag it to an open space next to any open tab in the work area. The tip changes from `Invalid` to `Open`. Release the flow to open it.

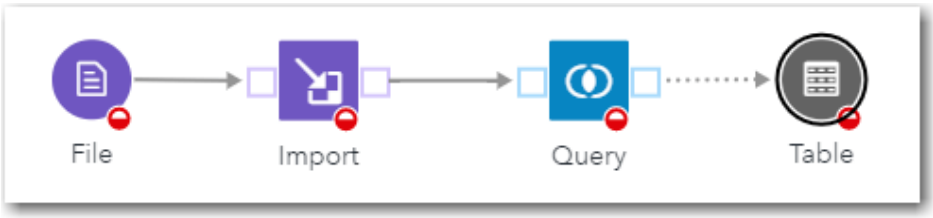
Understanding Nodes

About Flow Nodes

Flows are built from a sequence of data and operational nodes, which you can connect in order to specify the order of execution. Data nodes, such as the File and Table nodes, represent data in a flow. Operational nodes, such as Import, Query, and SAS Program nodes, represent operations that can be performed on data. File nodes and Table nodes are not considered operational nodes.

You can add nodes to a flow as placeholders in order to create the structure of your flow, and then specify the attributes and content of the nodes later. For example, you could create a flow with placeholders for an external file, an Import node, a Query node, and a SAS node. When you are ready, you can specify the external file and output table that you want to use as well as the options for the Import and Query nodes.

Note: You cannot undo changes to nodes by clicking ↶ on the toolbar.



You can also add saved files to a flow from different sections of the navigation pane. Each node can have input and output ports, depending on the node type.

> **TIP** You can move individual nodes by dragging them around the flow canvas. To move multiple nodes as a group, use your mouse pointer to draw a box around the nodes that you want to move. The nodes in the box are selected. Click one of the selected nodes to drag the entire group to another location on the flow canvas.

About Flow Node Types

The types of nodes that you can add to a flow are listed in the **Steps** section of the navigation pane. Here are the basic steps that are available in SAS Studio:

- Export - exports data to an external file.
- File - references an external file.
- Import - converts an external file to a SAS data set.
- Table - references a SAS data set from a SAS library.
- SAS Program - enables you to write a new SAS program or open a saved SAS program or snippet.
- Query - extracts data from one or more tables according to criteria that you specify.
- Sort - enables you to order your data by the values of one or more columns.

You can use the node properties under the flow canvas to specify the attributes and content of the node. Every node has a Node tab, which you can use to specify a name and description of the node.

Adding Nodes to a Flow

When you have a flow open in the work area, you can use the **Steps** section of the navigation pane or click **Add** on the flow toolbar to add nodes to a flow.

> **TIP** You can copy and paste one or more nodes within a flow. When you copy one or more nodes in a flow, the properties and values that are associated with the nodes are retained.

To add one or more saved files to a flow, expand the **Explorer** section of the navigation pane and drag the appropriate files to the flow canvas. You can add the following types of saved files to a flow:

- SAS program files (*.sas)
- Comma-separated values files (*.csv)
- Delimited files (*.tsv, *.tab, or *.dlm)
- Text files (*.txt)

You cannot add *.sas7bdat, *.ctm, *.ctk, *.ctl, *.cqy, or *.flw files to a flow.

Connecting Nodes

Nodes are joined by connections to either the node itself or to a port on the node.

Depending on the node type, you can drag a node to the flow canvas, drop it on another node, and automatically create a connection between the nodes. The tip on your mouse pointer changes to indicate valid locations in which you can insert the node in the flow, connect the node to the input port of another node, or connect the node to the output port of another node.

For example, you can drag a Table node and connect it to either the input or output port of a Query node, or you can drag an external .csv file from the **Explorer** section of the navigation pane and connect it to the input port of an Import node.

> **TIP** SAS Studio can auto-arrange the nodes in your flow by repositioning nodes, ports, and connections into a logical arrangement. You can modify any changes SAS Studio makes by manually updating the flow. To auto-arrange the nodes in the flow, click on the toolbar.

Expanding and Collapsing Node Ports

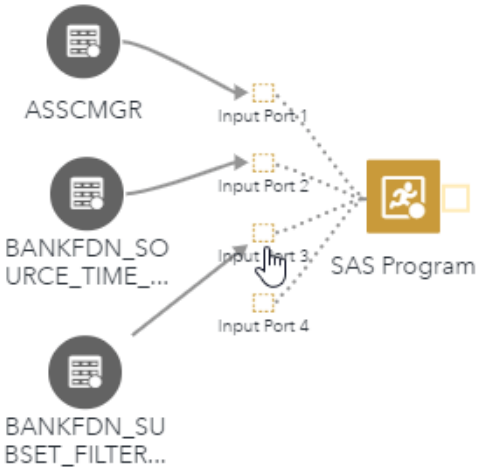
Some node types, such as SAS Program, Query, and Import, can contain one or more input and output ports. Ports represent either an input source or an output target for connecting data with nodes.

By default, ports are collapsed. To expand or collapse a port, right-click the port and select **Expand** or **Collapse**. You can expand or collapse all ports by clicking **View** on the toolbar and selecting **Expand all ports** or **Collapse all ports**.

You can assign source data to an input port by connecting another node to the input port. Depending on the node type, input can be either data or the output from another node.

SAS Studio automatically adds ports to a node as they are needed when you connect nodes. You can also manually add input and output ports to some node types by right-clicking the node and selecting **Add input port** or **Add output port**. When a node has more than two ports, they are displayed as a single port with the number of ports noted.

When a node has multiple input ports, the ports automatically expand when you create a connection.



Note: When a node has multiple output ports, you must first expand the ports before creating a connection to another node.

By default, output ports represent tables in the Work library. To specify a location for the data, add a Table node to the flow and connect the node to the output port. An empty output port indicates that there is no data available in the port.

When a node has run successfully and data is available from the output node, the node is displayed as filled in.

Working with Data in a Flow

Exporting Data to an External File

You can use the Export node to save data to a delimited file (*.csv, *.dlm, *,tab, *.tsv), a text file, or a Microsoft Excel file (*.xls, *.xlsx).

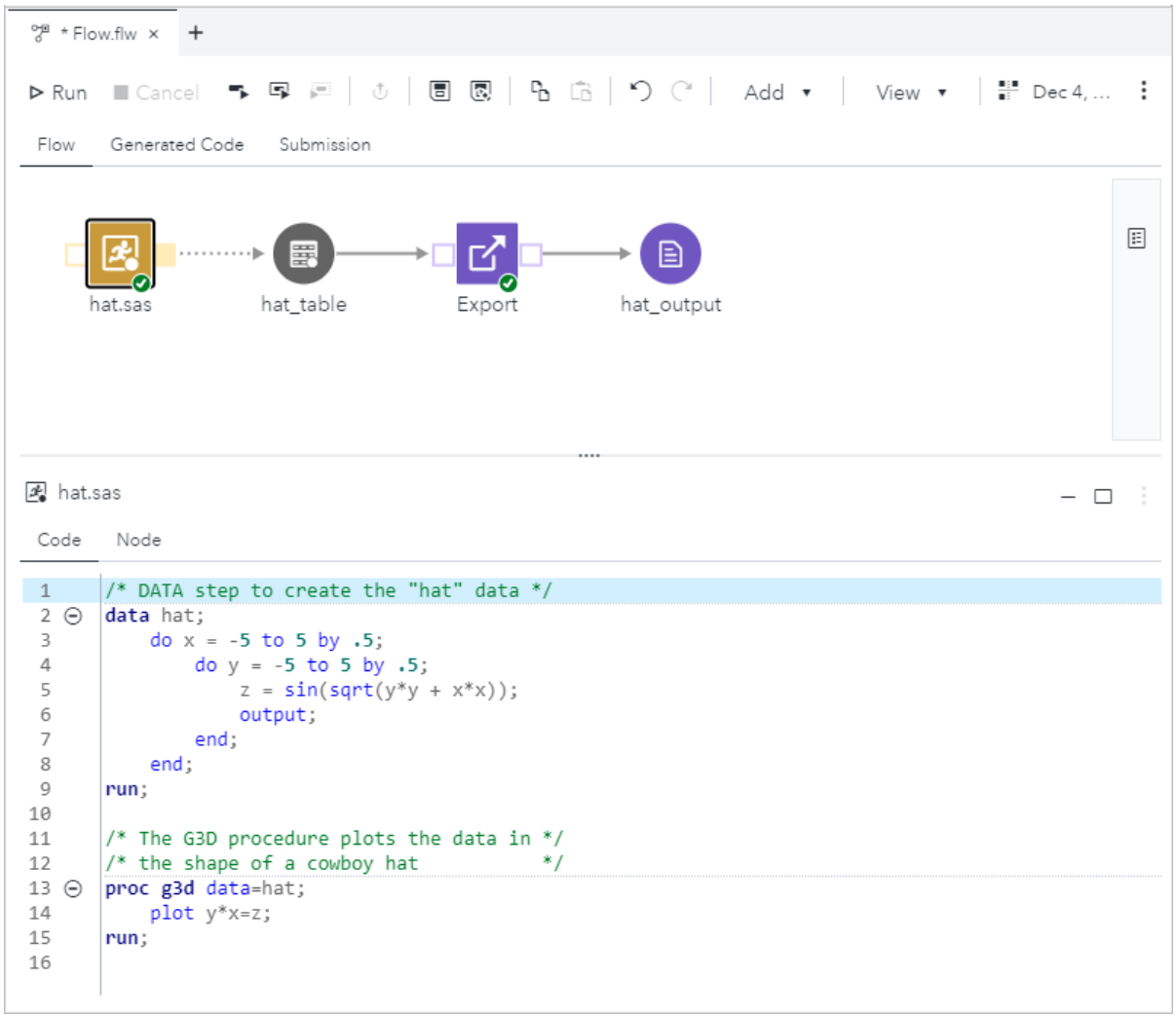
To export data in a flow:

1. Click the **Steps** section of the navigation pane.
2. Expand the **Data** folder and double-click the **Export** step to add an Export node to the flow.
3. Connect the input port of the Export node to a data source, such as a Table node or another node that creates an output table, such as a Query node or an Import node. For more information, see “Connecting Nodes” on page 108.
4. Connect the output port of the Export node to a File node. Use the File node to specify options for the output file, including the name and location of the file. The file extension determines the format of the exported file. For more information, see “Adding an External File to a Flow” on page 111.

   TIP You can export data to an unconnected File node in your flow by right-clicking the **File** node and selecting **Add an export**. You must also connect a data source to the input port of the Export node.

5. Click **Run**.

The following example shows the `hat.sas` SAS Program node that creates an output table in the `hat_table` Table node. The `hat_table` node is connected to the input port of the Export node and is exported as a text file to the `hat_output` File node.



Adding an External File to a Flow

File nodes are used to point to external files. You can use a File node as input to a step such as the Import step in order to convert a file to a table in a SAS library. File nodes can also be used as output to a step such as the Export step.

File nodes can point to these file types:

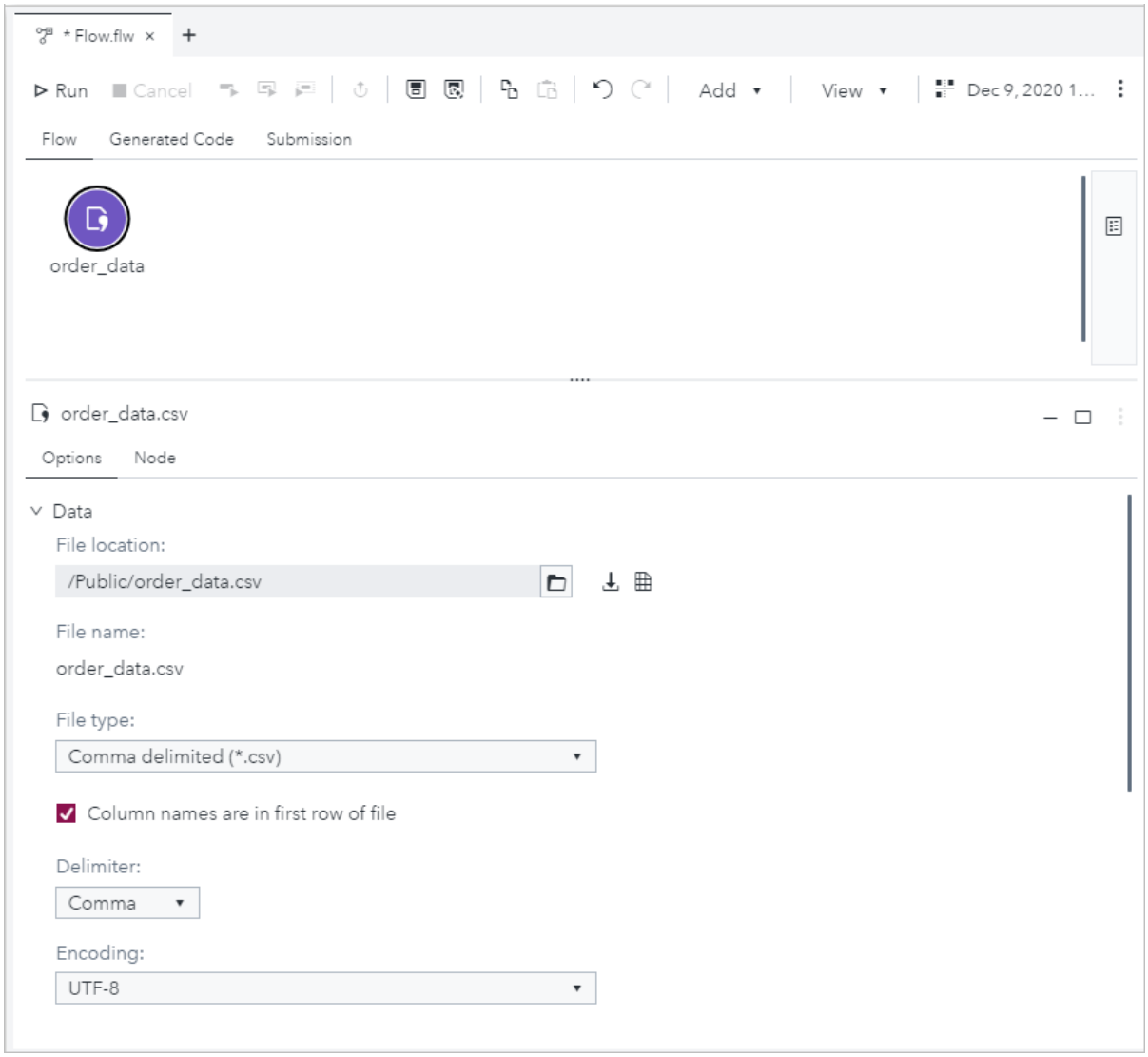
- delimited (*.dlm).
- comma-delimited (*.csv) .
- tab-delimited (*.tab, *.tsv).
- text (*.txt).
- Microsoft Excel (*.xlsx). To import XLSX files, you must license and install SAS/ACCESS to PC Files.
- Microsoft Excel 97–2003 (*.xls).

The file that is referenced by a File node must be located in SAS Content or on the file system for the current SAS Compute Server.

To add an external file to the flow:

1. Click the **Explorer** section of the navigation pane and navigate to the appropriate folder.
2. Right-click the file that you want to add, and then select **Add to flow**. You can also drag the file into the flow.

The File node options include information about the properties of the file. Depending on the file type, the options also enable you to download and view the file and update some of the data that is associated with the node.



- To download the file, click . Depending on your browser settings, the file might open in a viewer.
- To view the raw data, click .

  Note: This option is not available for Microsoft Excel files.

- You can update the following fields:
  - **File location**.

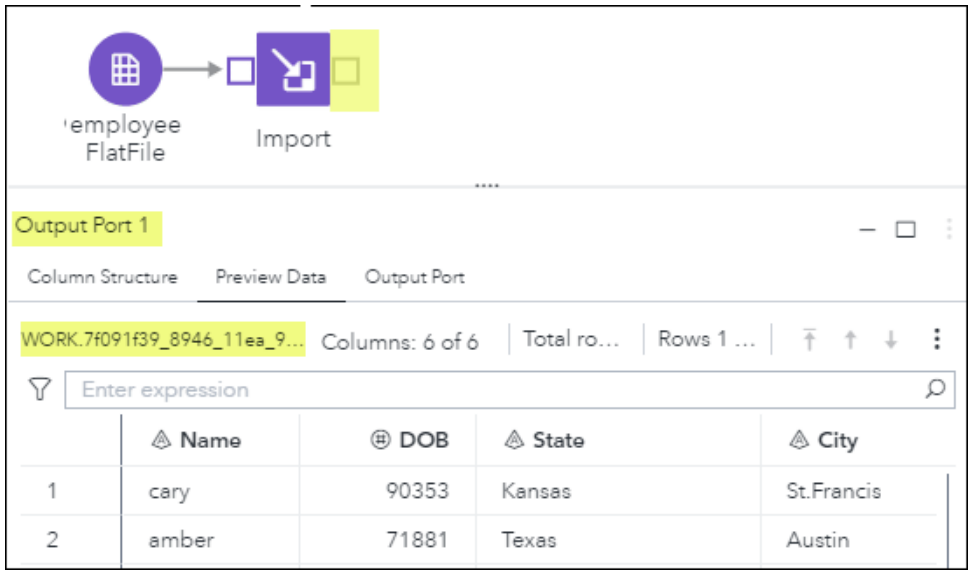
- **File type**.
- **Column names are in first row of file** - indicates whether the data includes a header row.
- **Delimiter**.
- **Encoding**.

Importing Data from an External File

The information in external files, such as delimited files and Microsoft Excel spreadsheets, must be converted to SAS format in order to be queried, filtered, or analyzed by SAS. Use the File node and the Import node to convert the information in an external file to the SAS format that is required for a flow.

The next figure shows a flow that converts the information in a delimited file (employee FlatFile) and writes the result to the output port of the Import node. By default, output for the Import node is written to a table in the Work library. Import node output can be connected to any flow node that accepts SAS data as an input, such as a Table node, a Query node, or a SAS Program node.

Figure 4.3 *Flow That Imports Delimited Data to the Work Table for the Import Node*



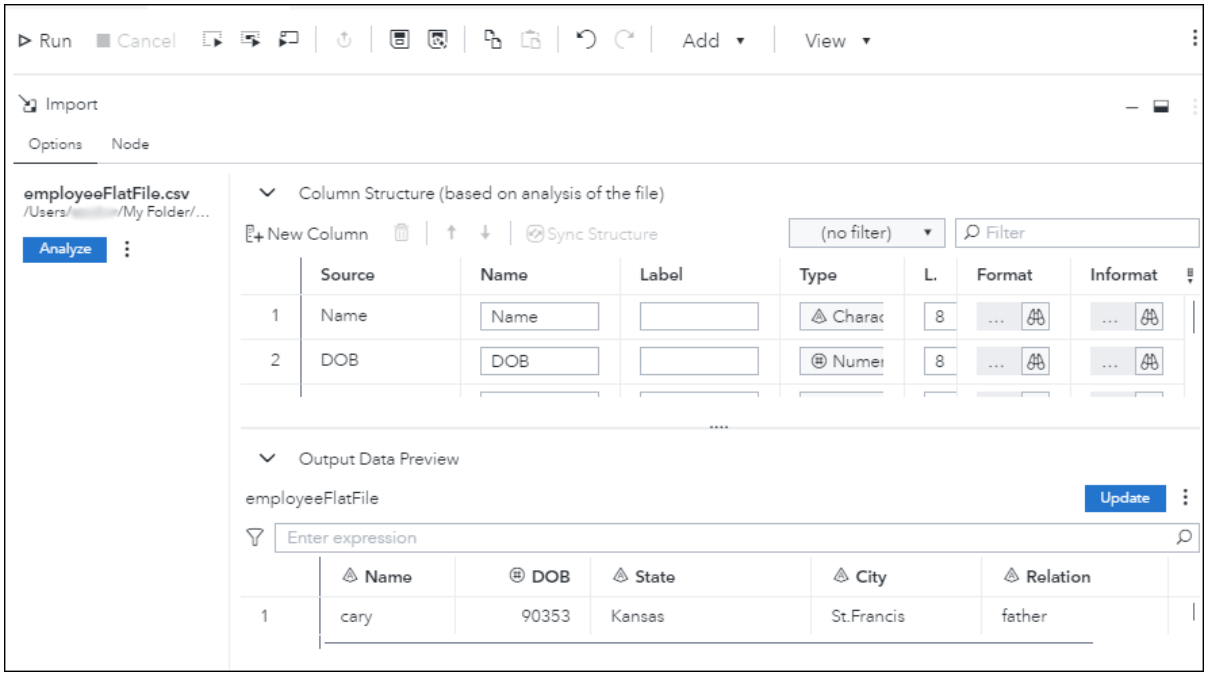
1 Right-click the **File** node in the flow and select **Add an import**.

   An Import node is added to the flow, and the File node is an input to the Import node.

2 Click the **Import** node, click the **Options** tab, and then click **Analyze** to identify the structure of the external file.

   The next figure shows the results from analyzing an external file.

Figure 4.4 *Analysis of an External File*



3 If you are importing a Microsoft Excel spreadsheet with multiple worksheets, select the worksheets that you want to import. By default, all of the worksheets are selected.

4 Review the column structure for the external file.

The same column structure is used for the output table for the Import node. If you change the column information, click **Update** to see the impact on the output data for the Import node.

Note: You cannot change the column information for Microsoft Excel files.

5 When you are finished reviewing the column information, click **Run**.

6 To view the output, click the output port for the Import node, and then click the **Preview Data** tab.

Adding a Table from a SAS Library to a Flow

You can use Table nodes to add SAS tables to your flow. Table nodes enable you to specify the library and table name of a table in the flow and can be used to connect to the input and output ports of operational nodes.

If you have Read access to a table in a SAS library, you can specify the table in a Table node. You can use the Table node as the input for an operational node in the flow, such as a Query node or a SAS Program node.

If you have Write access to a SAS library, you can use a Table node to create or update a table in that library. The table can be an existing table or a table that is created when a flow runs. You can then use the Table node as the output for an

operational node in the flow, such as an Import node, a Query node, or a SAS Program node. By default, output ports represent tables in the Work library. To specify a name and location for the data, add a Table node to the flow and connect the node to the output port.

To add a Table node to a flow:

1 Click the **Steps** section of the navigation pane.

2 Expand the **Data** folder and double-click the **Table** step.

> **TIP** You can quickly add a Table node to the flow and connect the node to the output data source of an operational node by right-clicking the output port of the operational node and selecting **Add a table**. The operational nodes include nodes such as Import, Query, and SAS Program. File nodes and Table nodes are not considered operational nodes.

3 In the flow, click the **Table** node and use the **Table Properties** tab to specify a library and table.

4 Use the **Published Columns** tab to edit and add columns. To copy the column definitions from an existing input table, click **Sync Structure**.

> **Note:** When you click **Sync Structure**, you are prompted to replace your existing column structure with the column structure on the **Preview Data** tab. Any changes you have made to the column structure are overwritten.

To add an existing table to a flow:

- From the **Libraries** section of the navigation pane, drag one or more tables to the flow canvas. A Table node is automatically created for each table.

Working with Code in a Flow

About the SAS Program Node

Use a SAS Program node to perform any SAS operation that is required in a flow. You can add a program that is an external file or a code snippet. The next figure shows a flow in which a SAS Program node (Means) runs the MEANS procedure against the BASEBALL table. The results are written to the default output for the SAS Program node: a table in the Work library.

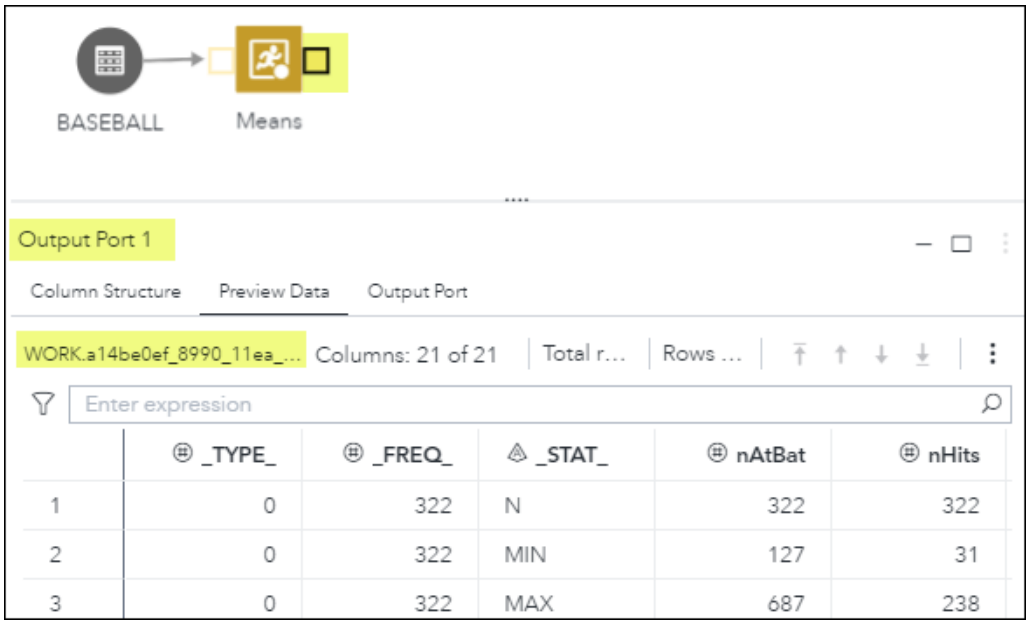
> **Note:** If your SAS program creates output data, the columns in the output data are automatically assigned to the column structure of the data in the output port after the

program has run. The output data is available to be used as input for subsequent nodes in the flow.

The input for a SAS Program node is a table output from a Table node, an Import node, a Query node, or a SAS Program node. SAS Program node output can be connected to a Table node, a Query node, or a SAS Program node.

The SAS Program node interface is the same as the interface for stand-alone SAS programs. For more information, see Chapter 2, “Working with Programs,” on page 31.

Figure 4.5 *SAS Program Node after a Successful Run of the Flow*



Adding a SAS Program

To add a SAS Program node by selecting a SAS program file:

1 From the **Explorer** section of the navigation pane, drag the SAS program file to the flow canvas. You can also right-click a program and select **Add to flow**.

> **TIP** You can also add snippets to the flow from the **Snippets** section of the navigation pane.

2 Use macro variables in your code to reference the input and output ports for the SAS Program node. For more information, see “Using Macro Variables to Reference Input and Output Ports” on page 117.

3 Connect another node to the output port of the SAS Program node, as appropriate.

Copying Code to a Flow

You can copy the code that is automatically generated by a task, or you can copy the code from a SAS program to a SAS Program node in a flow from which you can make and save changes.

Note: The code in the SAS Program node is no longer associated with the original program or task. Editing this code does not affect the original program or task.

To copy code to a flow:

- Open the SAS program or task from which you want to copy code. On the toolbar, click **Code to Flow** and select the flow that you want to add a SAS Program node to.

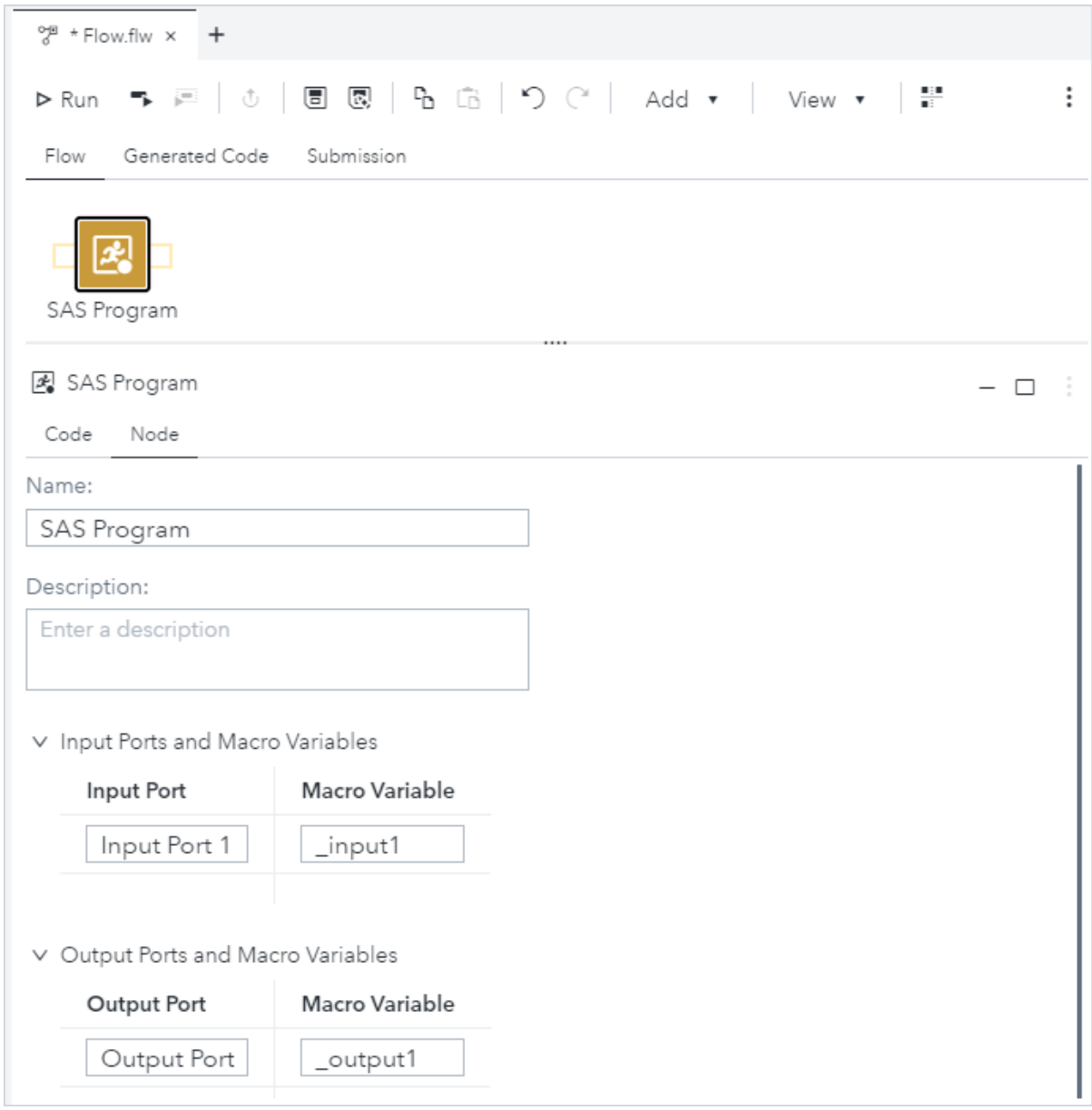
  Note: The **Code to Flow** option is available only if these conditions are met:

- A flow must be open in the work area.
- The input data source must be specified for the task.

Using Macro Variables to Reference Input and Output Ports

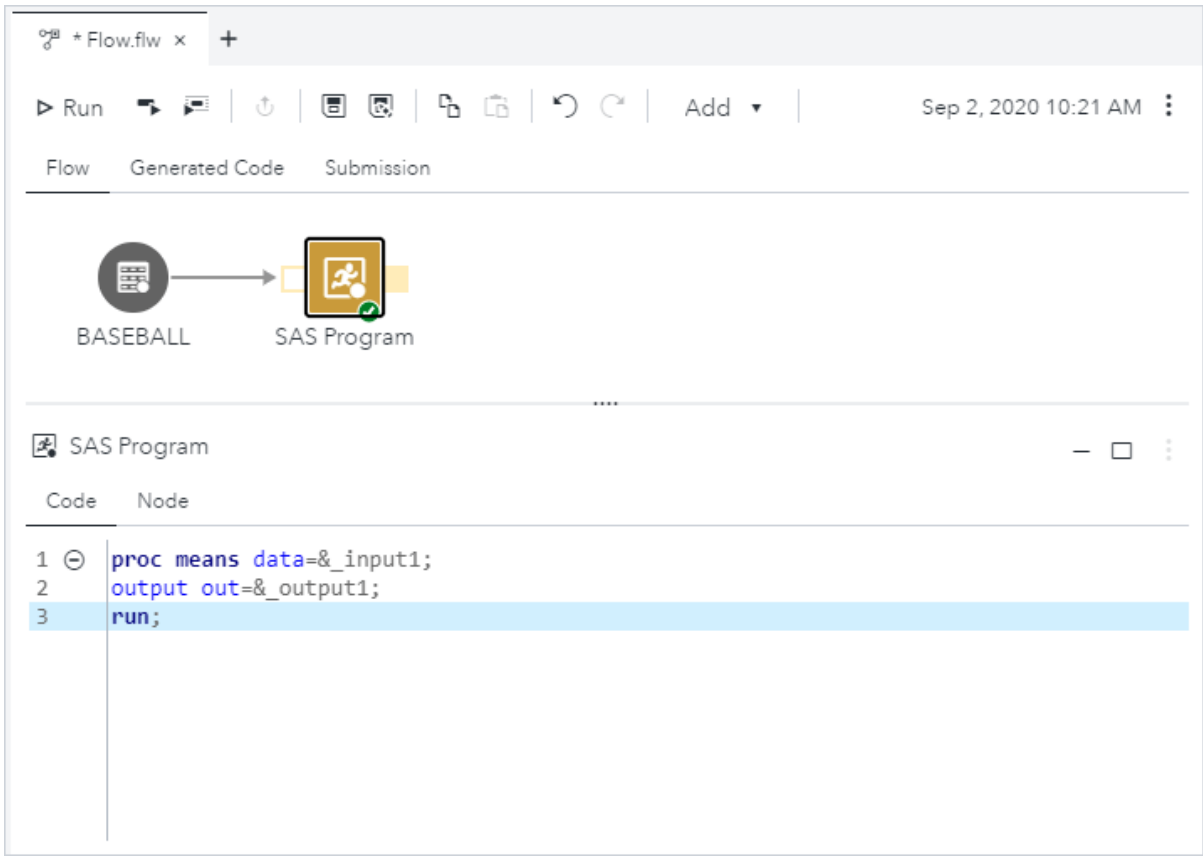
You must use macro variables in your code to reference the input and output ports of the SAS Program node before data can be read from or written to a Table node.

You can view the default input and output ports of a SAS Program node by clicking the node in the flow, and then clicking the **Node** tab. The input port, the output port, and the associated macro variables are displayed.

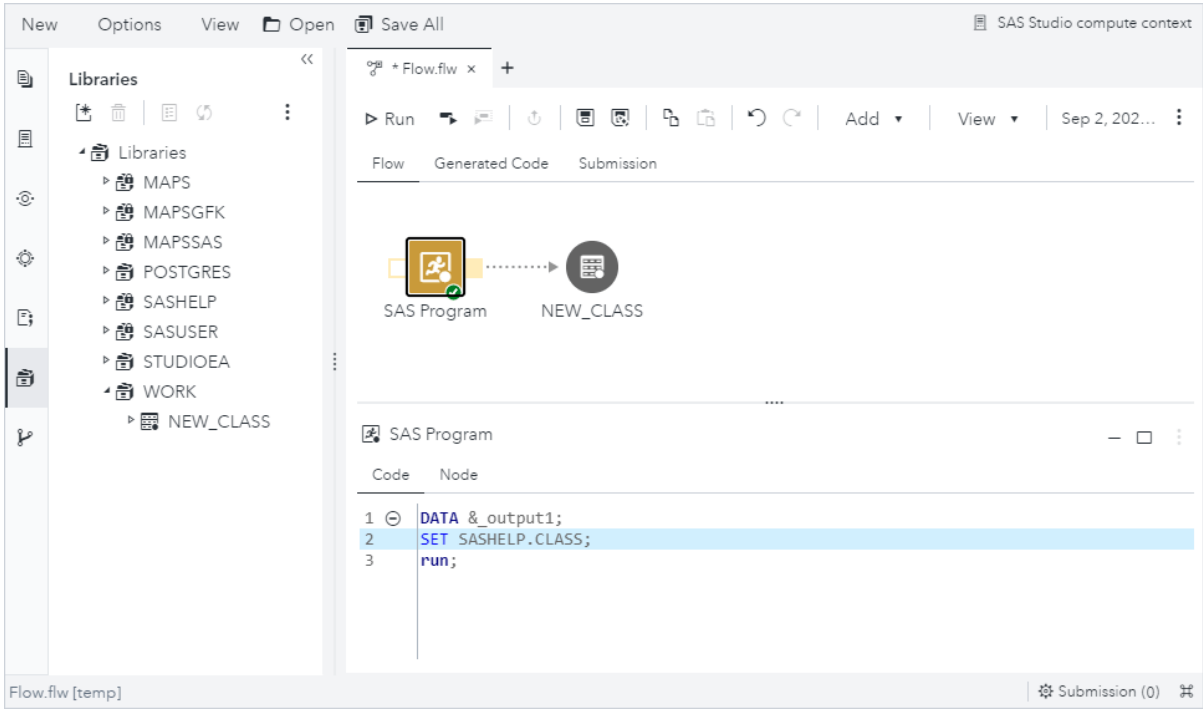


Note: You can change the default names of the macro variables. You can then reference the custom macro variable names in your code. For more information, see "Macro Variables Defined by Users" in *SAS Macro Language: Reference*.

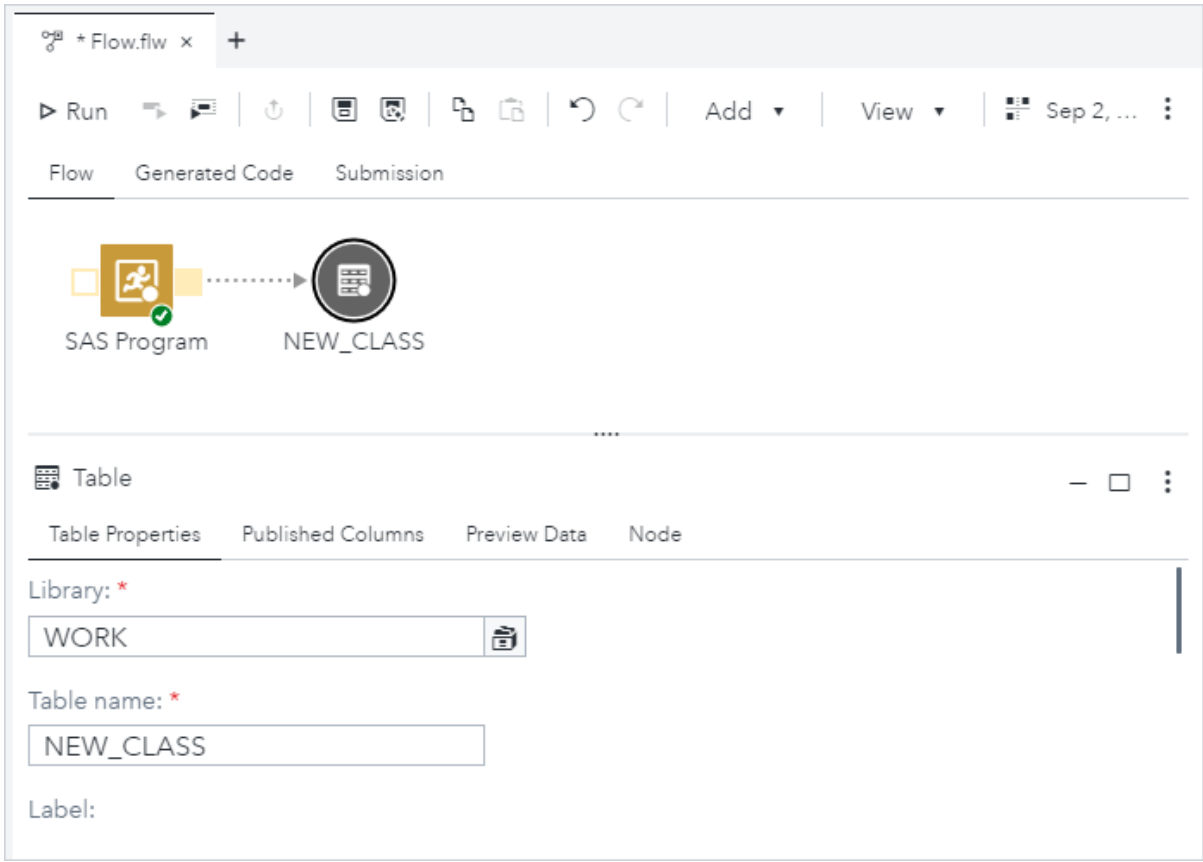
In the following example, the BASEBALL table is connected to a SAS Program node. The SAS Program node runs the MEANS procedure against the BASEBALL table. The code that is associated with the SAS Program node specifies the `&_input1` macro as the data source for the MEANS procedure and the `&_output1` macro as the destination for the output data. By default, the results are written to a table in the Work library.



The following example shows a SAS Program node that is connected to the NEW_CLASS table. The SAS Program node runs a DATA step to create a table that is based on the CLASS table. The code that is associated with the SAS Program node specifies the `&_output1` macro variable as the destination for the output data.



The table properties for the Table node specify the Work library and NEW_CLASS table name.



Creating a Flow from a SAS Program

You can convert a SAS program file to a flow. The input tables, procedures, and output tables in the program are used to create nodes in the flow.

Note: You might need to manually edit the flow that is created by the **Create a Flow from a Program** option in these cases:

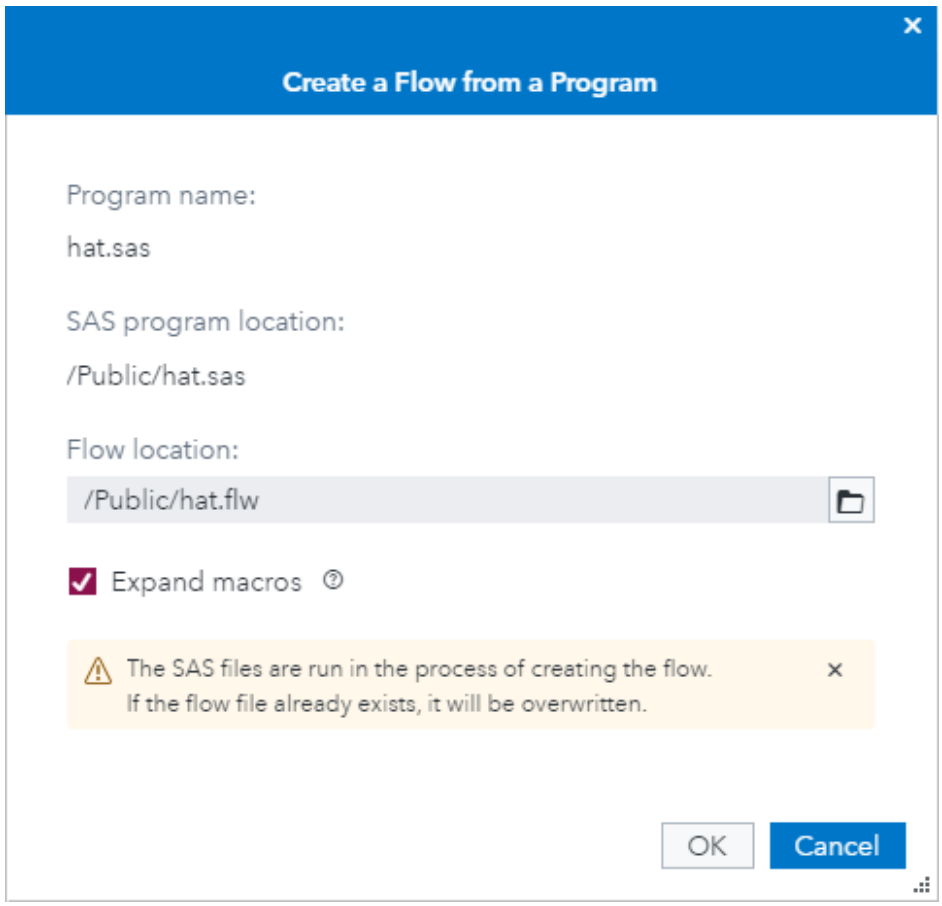
- The **Create a Flow from a Program** option cannot parse all of the possible LIBNAME options for DBMS engines. If you are creating a flow from a program that contains LIBNAME options for DBMS engines, you need to verify that the LIBNAME statement in the SAS Program node in the flow is correct and that you can access the appropriate library. You might need to manually edit the SAS Program node to add any missing LIBNAME options.
- If you are creating a flow from a program that uses CAS procedures, you might need to manually add input and output ports to the SAS Program nodes and create connections between nodes to ensure that the nodes are run in the correct order.

To create a flow from a SAS program:

1 Right-click the SAS program in the **Explorer** section of the navigation pane and select **Create flow from program**.

> **TIP** You can also create a flow from an open program by clicking ⋮ on the program toolbar and selecting **More options** ⇨ **Create flow from program**.

Note: In order to create the flow, SAS Studio runs the program.



2 By default, the new flow is created in the same location as the SAS program file. If a flow file with the same name already exists, the file is overwritten. To create the flow in a different location, click 🗀 and browse to find the location that you want to use.

3 If your program includes macros and you want to create a separate SAS Program node for each macro, select **Expand macros**. This option is selected by default.

4 Click **OK** to create the flow. The flow opens on a new tab in the work area.

Transforming Data in a Flow

Understanding the Steps That Subset Data

You can choose from among three different steps to select a subset of data from an input table. Each step can be useful in different situations, depending on your needs:

- **Branch Rows** - splits a table into multiple output tables based on conditions that you specify by using column values. The Branch Rows step performs only a single function in a flow, which makes it easy to understand the function of a Branch Rows node in a flow that is complicated and includes many nodes. For more information, see “About the Branch Rows Step” on page 123.
- **Filter Rows** - selects a subset of rows from an input table and writes the rows to a single output table. The Filter Rows step performs only a single function in a flow, which makes it easy to understand the function of a Filter Rows node in a flow that is complicated and includes many nodes. For more information, see “About the Filter Rows Step” on page 128.
- **Query** - selects, filters, and sorts columns from one or more input tables that can be joined together and writes the rows to a single output table. The Query step can perform multiple functions, which can make understanding a flow more complicated. To understand the functions that a Query node performs in a flow, you must examine the options that are specified for the node. For more information, see “Creating a Query in a Flow” on page 141.

Note: The Branch Rows and Filter Rows steps are available only if your site licenses SAS Studio (Analyst).

Table 4.1 *Comparison of Steps That Subset Data*

	Branch Rows	Filter Rows	Query
Selects a subset of rows based on one or more filter conditions	x	x	x
Includes option to use the Expression Builder for creating expressions	x	x	x
Includes a condition for rows that do not match any other condition	x		
Writes rows to multiple output tables	x		

	Branch Rows	Filter Rows	Query
Selects rows from multiple input tables			x
Filters, sorts, and groups columns			x
Creates calculated columns			x

Branch Rows Step: Split an Input Table into Output Tables

About the Branch Rows Step

By default, the Branch Rows step splits a table into two output tables based on conditions that you specify by using column values. The conditions that you create do not need to be mutually exclusive: a row can be written to more than one output table.

Note: The Branch Rows step is available only if your site licenses SAS Studio (Analyst).

TIP You can also use the Filter Rows step and the Query step to select a subset of rows from an input table. For more information, see "Understanding the Steps That Subset Data" on page 122.

Node Connection Requirements

In order to run the Branch Rows step, you must create connections to the input and output ports of the node as indicated here:

Input Port	Output Port
■ Table node or ■ Operational node that creates an output table, such as a Query node, a SAS	No connection is required. **Note:** By default, the output data is written to temporary tables in the Work library. You can specify the library and name of the output tables by connecting the output ports to Table nodes. For

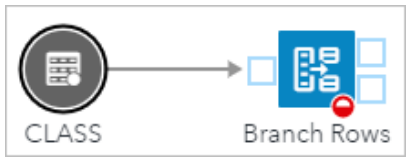
Input Port	Output Port
Program node, or an Import node	more information, see “Adding a Table from a SAS Library to a Flow” on page 114.

Example: Splitting the Sashelp.Class Data Set

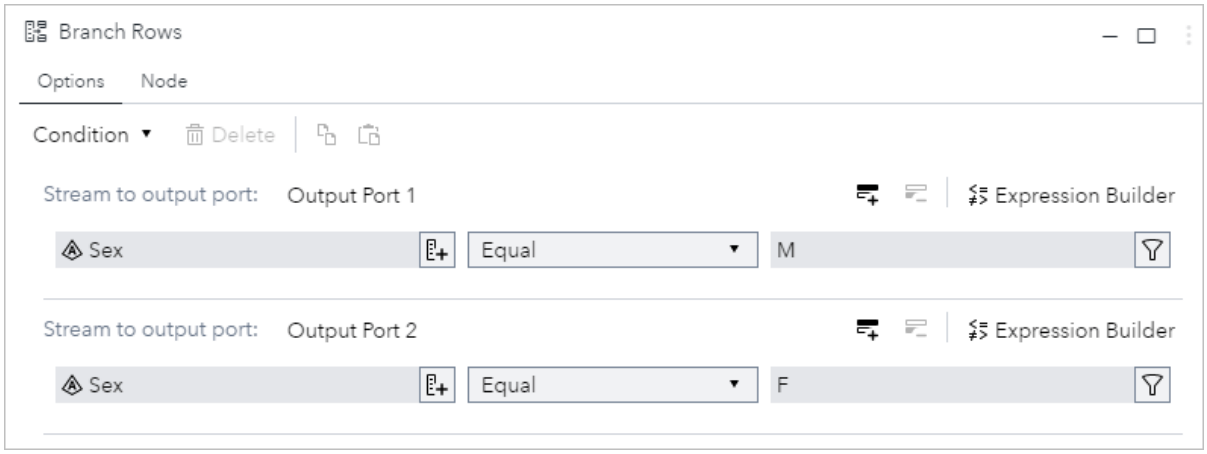
In this example, you split the Sashelp.Class data set into two tables: one for male students and another for female students.

To create this example:

1. From the main SAS Studio menu, select **New** ⇨ **Flow**.
2. In the **Steps** section of the navigation pane, expand the **Transform Data** folder, and double-click **Branch Rows**.
3. In the **Libraries** section of the navigation pane, expand the Sashelp library. Drag the Class data set onto the input port of the Branch Rows node. Drop the data set on the flow canvas when the tooltip on your mouse pointer changes to `Connect to input port`.



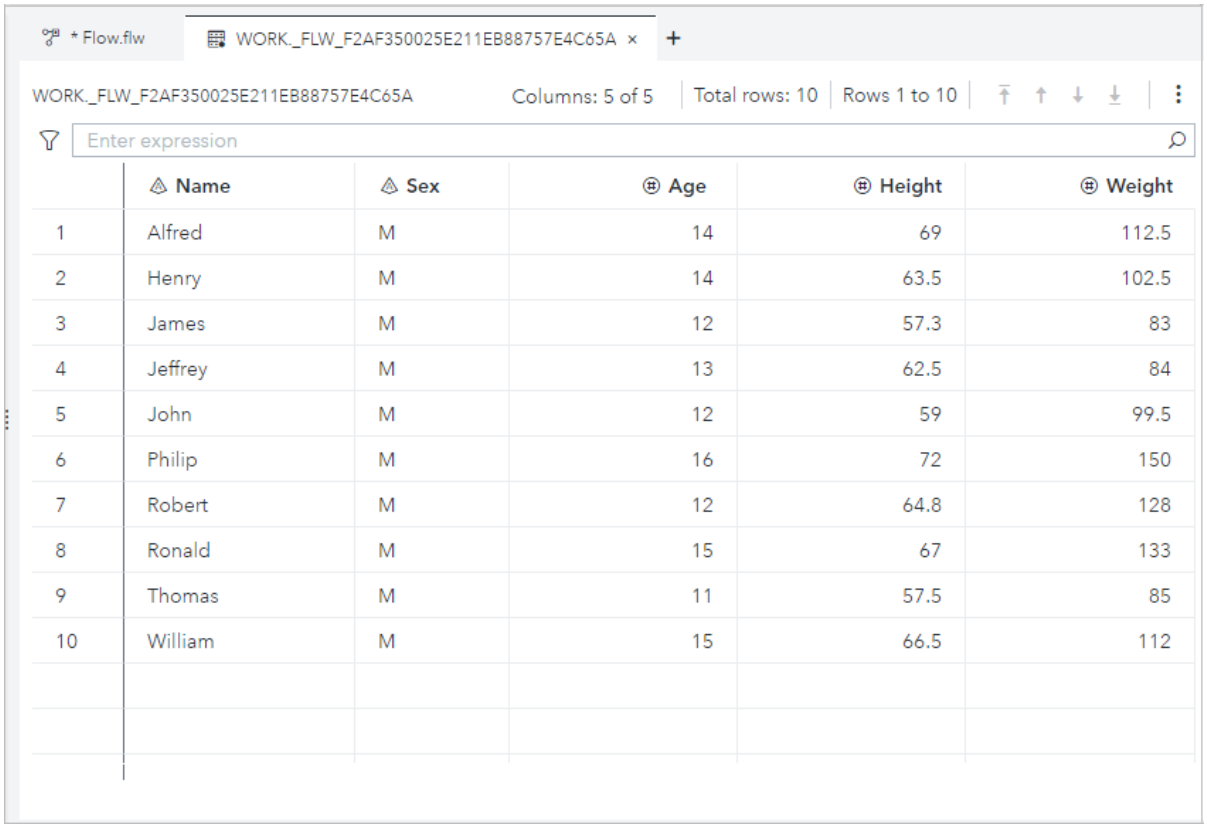
4. Click the **Branch Rows** node to specify the conditions for the output port. On the Options tab, click  for Output Port 1 and select the *Sex* column. Accept the default condition of `Equal`, and click  .
5. In the Add Filter window, click  . Click **Get Values** in the Select Value window. Select `M` and click **OK**. Click **Filter** to create the condition.
6. Repeat steps 4 and 5 to create a similar condition for Output Port 2 for female students.



7 To run the flow, click ▷ Run.

Here is the default output table of male students.

> **TIP** You can specify the library and name of the output tables by connecting the output ports to Table nodes. For more information, see "Adding a Table from a SAS Library to a Flow" on page 114.



* Flow.flw | WORK._FLW_F2AF350025E211EB88757E4C65A

WORK._FLW_F2AF350025E211EB88757E4C65A Columns: 5 of 5 | Total rows: 10 | Rows 1 to 10

Enter expression

	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Henry	M	14	63.5	102.5
3	James	M	12	57.3	83
4	Jeffrey	M	13	62.5	84
5	John	M	12	59	99.5
6	Philip	M	16	72	150
7	Robert	M	12	64.8	128
8	Ronald	M	15	67	133
9	Thomas	M	11	57.5	85
10	William	M	15	66.5	112

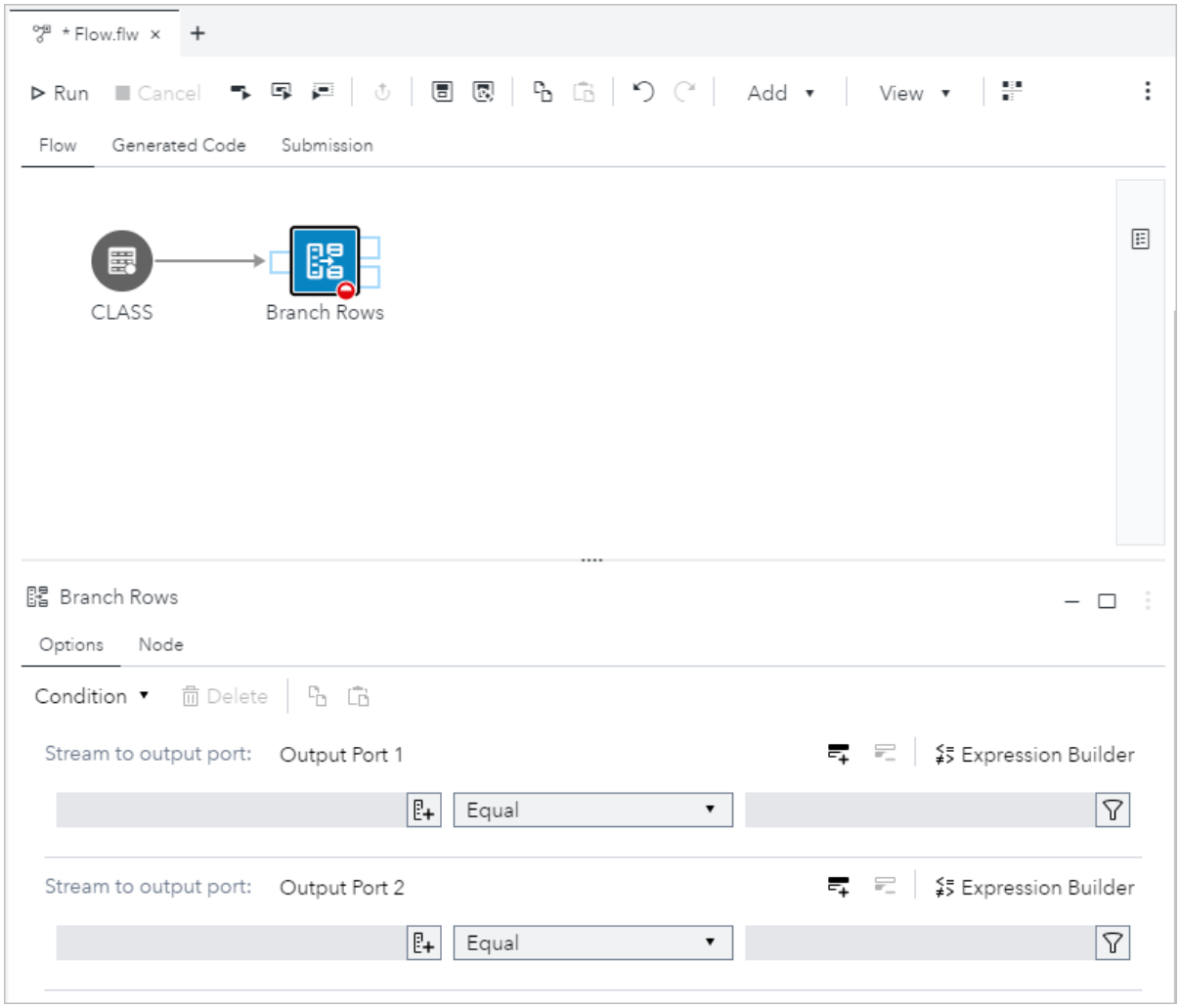
Branch Rows: Step-by-Step Instructions

By default, the Branch Rows step includes two output ports and the options for two conditions. You can choose to specify only one condition and output port, or you can add additional conditions and output ports. You can also add a nonmatching condition that writes the rows that do not meet any condition to a separate output port.

1 In the **Steps** section of the navigation pane, expand the **Transform Data** folder, and double-click **Branch Rows**.

2 Connect the input port of the Branch Rows node to a data source by using a Table node or another node that creates an output table, such as a Query node, a SAS Program node, or an Import node. For more information, see "Connecting Nodes" on page 108.

3 Click the **Branch Rows** node, and then click the **Options** tab.



4 Click  for Output Port 1, and select the column that you want to use. Click **OK**.

> **TIP** To use the expression builder to create a condition, click **Expression Builder** on the toolbar for the condition. For more information, see "Building an Expression" on page 273.

5 Select a comparison operator from the operator drop-down list. The default value is `Equal to`.

6 If the operator that you have selected requires a value, click . In the Add Filter window, enter or select a value in the **Value** box. To choose from a list of values, click and click **Get Values** in the Select Value window. Select the values that you want to use and click **OK**.

> **TIP** If you want to search for a value in the Select Value window, use the unformatted value.

7 Depending on the data type of the column that you are using in the condition, you can choose from the following options:

- **Match case** – retrieves only rows that match the capitalization of the value that you specify. If this option is not selected, the UPPER function is applied to the expression. This option is not selected by default.
- **Quote string** – encloses values in single quotation marks. This option is selected by default. If you are using a macro variable or other value that is evaluated when the filter is run, you should clear this option.
- **Allow macros** – enables you to enter character values in a filter on a numeric column.

8 Click **Filter** to add values to the condition.

9 To add another row to the condition, click and repeat steps 4–8. If you create more than one comparison expression in your condition, the default relationship between these filter elements is AND. You can change the relationship between filter elements from AND to OR by clicking the relationship drop-down list.

Note: To delete a row from a condition, select the row and click

10 Repeat steps 4–9 to create a condition for Output Port 2.

11 To create additional conditions, select **Condition** ⇨ **Add condition** and repeat steps 4–9.

To add a condition for the rows that do not match any other condition, select **Condition** ⇨ **Add nonmatching condition**.

12 To run the flow, click ▷ Run.

Filter Rows Step: Subsetting Rows from an Input Table into an Output Table

About the Filter Rows Step

You can use the Filter Rows step to select a subset of rows from an input table and write the rows to an output table. The Filter Rows step can be combined with other steps in which you want to work with only a subset of the data in a table. For example, you can create a flow that uses the Filter Rows step to create a table that contains only baseball players who have had more than 10 home runs, and then use the Branch Rows step to split the filtered data into separate tables for each team.

> **TIP** You can also use the Branch Rows step and the Query step to select a subset of rows from an input table. For more information, see "Understanding the Steps That Subset Data" on page 122.

You can choose to create filter expressions by using the user interface in the step or you can create the filter by using the Expression Builder. For more information, see "Building an Expression" on page 273. SAS Studio automatically generates the appropriate DATA step WHERE statement for the filter when the step is run.

Note: The Filter Rows step is available only if your site licenses SAS Studio (Analyst).

Node Connection Requirements

In order to run the Filter Rows step, you must create connections to the input and output ports of the node as indicated here:

Input Port	Output Port
■ Table node or ■ Operational node that creates an output table, such as a Query node, a SAS Program node, or an Import node	■ Table node or ■ Operational node that requires an input table, such as a Query node, a SAS Program node, or an Export node

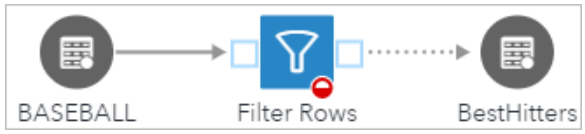
Example: Subsetting Data from the Sashelp.Baseball Data Set

In this example, you select a subset of rows from the Sashelp.Baseball data set and write the rows to a new output table named BestHitters. The BestHitters table contains a subset of the rows and all of the columns that are in the Sashelp.Baseball data set.

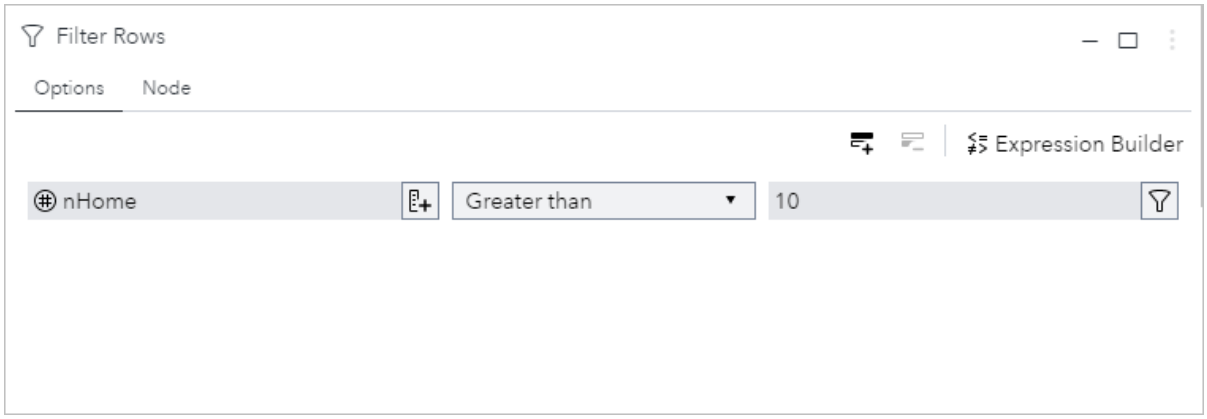
To create this example:

1. From the main SAS Studio menu, select **New** ⇨ **Flow**.
2. In the **Steps** section of the navigation pane, expand the **Transform Data** folder and double-click **Filter Rows**.
3. In the **Libraries** section of the navigation pane, expand the Sashelp library. Drag the Baseball data set onto the input port of the Filter Rows node. Drop the data set on the flow canvas when the tooltip on your mouse pointer changes to `Connect to input port`.

4. On the flow toolbar, click **Add** ⇨ **Table**. Connect the Filter Rows node to the Table node by clicking the Filter Rows output port and dragging the mouse pointer to the Table node.
5. Click the **Table** node. On the **Table Properties** tab, click  beside the **Library** box, and select the Work library. In the **Table** box, enter *BestHitters*.

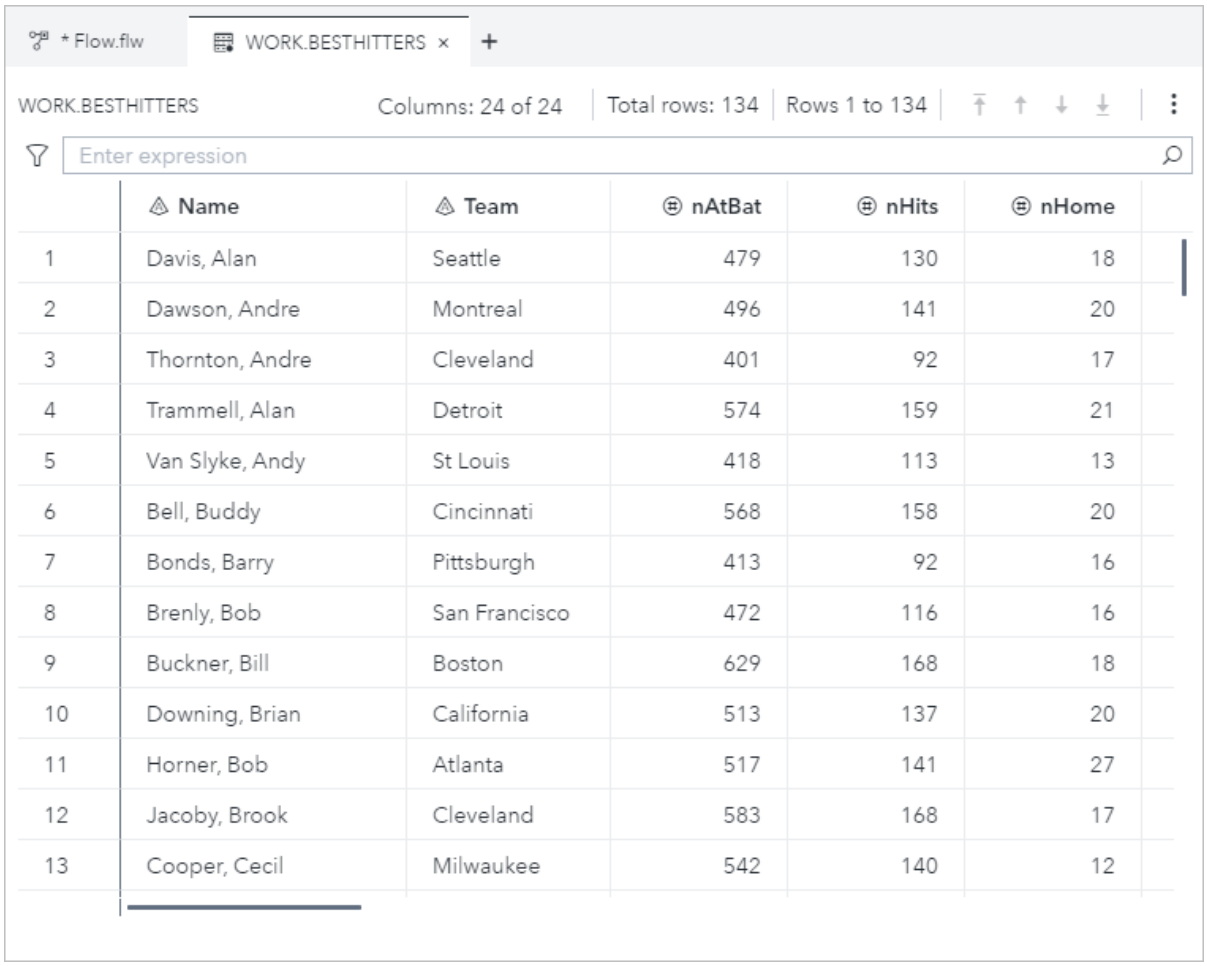


6. Click the **Filter Rows** node to specify the filter condition for the output table. On the Options tab, click  and select the `nHome` column. Click **OK**.
7. From the operator drop-down list, select `Greater than`, and then click  .
8. In the Add Filter window, click  . Click **Get Values** in the Select Value window. Select `10` and click **OK**. Click **Filter** to create the condition.



9 To run the flow, click ▷ Run.

Here is the output table of players with more than 10 home runs.



	Name	Team	nAtBat	nHits	nHome
1	Davis, Alan	Seattle	479	130	18
2	Dawson, Andre	Montreal	496	141	20
3	Thornton, Andre	Cleveland	401	92	17
4	Trammell, Alan	Detroit	574	159	21
5	Van Slyke, Andy	St Louis	418	113	13
6	Bell, Buddy	Cincinnati	568	158	20
7	Bonds, Barry	Pittsburgh	413	92	16
8	Brenly, Bob	San Francisco	472	116	16
9	Buckner, Bill	Boston	629	168	18
10	Downing, Brian	California	513	137	20
11	Horner, Bob	Atlanta	517	141	27
12	Jacoby, Brook	Cleveland	583	168	17
13	Cooper, Cecil	Milwaukee	542	140	12

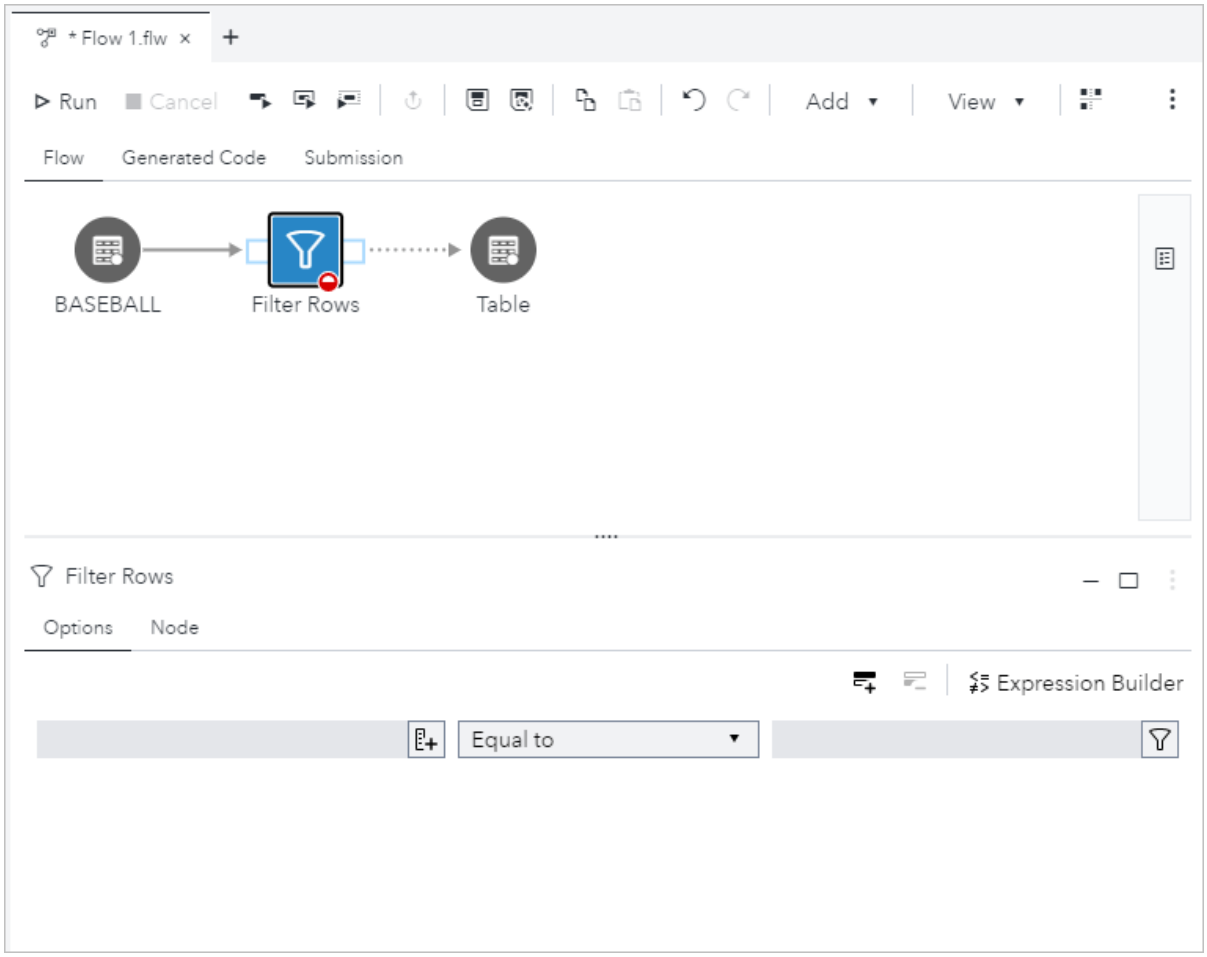
Filter Rows: Step-by-step Instructions

The Filter Rows step requires one input port and one output port. You can create one or more filter conditions to select a subset of the data that is written to the output table.

1. In the **Steps** section of the navigation pane, expand the **Transform Data** folder, and double-click **Filter Rows**.
2. Connect the input port of the Filter Rows node to a data source by using a Table node or another node that creates an output table, such as a Query node, a SAS Program node, or an Import node. For more information, see “Connecting Nodes” on page 108.
3. Connect the output port of the Filter Rows node to a data source by using a Table node or another node that requires an input table, such as a Query Node, a SAS Program node, or an Export node.

   Note: If you are using a Table node, click the node, and use the Table Properties tab to make sure that the node specifies a library and name for the table. Any existing data in the output table node is overwritten when you run the Filter Rows step.

4. Click the **Filter Rows** node, and then click the **Options** tab.



5 Click , and select the column that you want to use. Click **OK**.

> **TIP** To use the expression builder to create a condition, click **Expression Builder** on the toolbar for the condition. For more information, see "Building an Expression" on page 273.

6 Select a comparison operator from the operator drop-down list. The default value is `Equal to`.

7 If the operator that you have selected requires a value, click . In the Add Filter window, enter or select a value in the **Value** box. To choose from a list of values, click and click **Get Values** in the Select Value window. Select the values that you want to use and click **OK**.

> **TIP** If you want to search for a value in the Select Value window, use the unformatted value.

8 Depending on the data type of the column that you are using in the condition, you can choose from the following options:

- **Match case** – retrieves only rows that match the capitalization of the value that you specify. If this option is not selected, the UPPER function is applied to the expression. This option is not selected by default.
- **Quote string** – encloses values in single quotation marks. This option is selected by default. If you are using a macro variable or other value that is evaluated when the filter is run, you should clear this option.
- **Allow macros** – enables you to enter character values in a filter on a numeric column.

9 Click **Filter** to add values to the condition.

10 To add another row to the condition, click and repeat steps 5–9. If you create more than one comparison expression in your condition, the default relationship between these filter elements is AND. You can change the relationship between filter elements from AND to OR by clicking the relationship drop-down list.

Note: To delete a row from a condition, select the row and click

11 To run the flow, click Run.

Insert Rows Step: Insert Rows from an Input Table into an Output Table

About the Insert Rows Step

You can use the Insert Rows step to insert the rows from an input table into an output table. For example, you can create a flow that uses the Insert Rows step to create and add data to an output table. You can regularly update the data in the output table by scheduling the flow as a job.

The output table must contain one or more columns that have the same name and data type as columns in the input table. SAS Studio automatically generates PROC SQL or PROC FEDSQL code when the step is run.

The Insert Rows step enables you to add rows to an output table in any of the following ways:

- append the rows in the input table to existing rows in the output table
- replace the existing rows in the output table with the rows from the input table
- add the rows from the input table to a new output table

Note: The Insert Rows step is available only if your site licenses SAS Studio (Analyst).

Node Connection Requirements

In order to run the Insert Rows step, you must create connections to the input and output ports of the node as indicated here:

Input Port	Output Port
■ Table node or ■ Operational node that creates an output table, such as a Query node, a SAS Program node, or an Import node	Table node that contains one or more columns that have the same name and data type as columns in the input table

Example: Inserting Rows into a New Output Table from the Sashelp.Baseball Data Set

In this example, you insert rows from the Sashelp.Baseball data set into a new output table named KeyStats. The KeyStats table contains a subset of the columns that are in the Baseball data set.

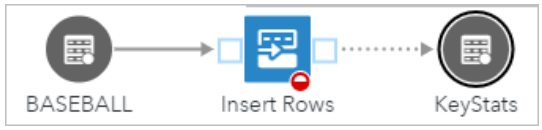
To create this example:

1 From the main SAS Studio menu, select **New** ⇨ **Flow**.

2 In the **Steps** section of the navigation pane, expand the **Transform Data** folder and double-click **Insert Rows**.

3 In the **Libraries** section of the navigation pane, expand the Sashelp library. Drag the Baseball data set onto the input port of the Insert Rows node. Drop the data set on the flow canvas when the tooltip on your mouse pointer changes to `Connect to input port`.

4 On the flow toolbar, click **Add** ⇨ **Table**. Connect the Insert Rows node to the Table node by clicking the Insert Rows output port and dragging the mouse pointer to the Table node.

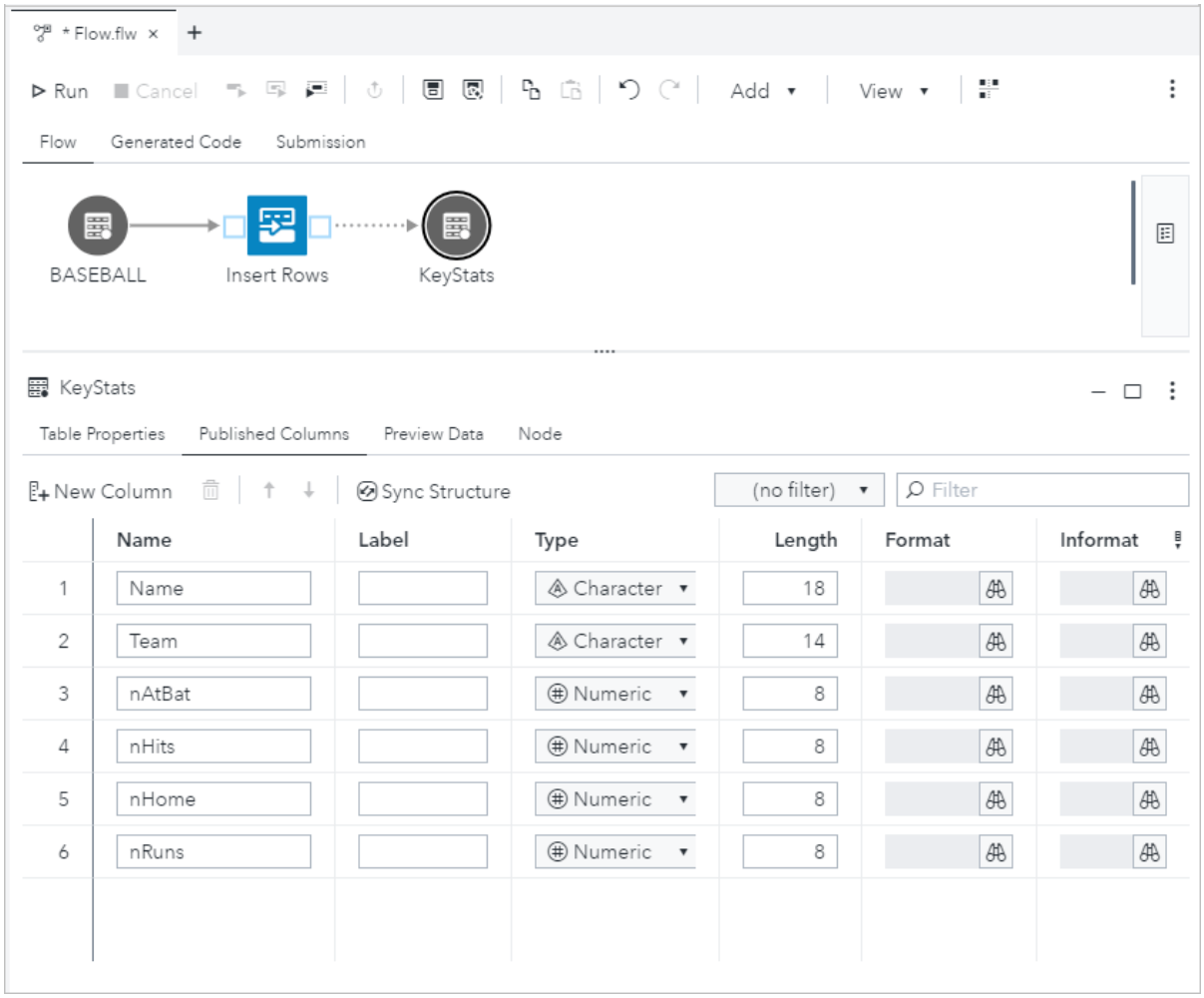


5 Click the **Table** node. On the **Table Properties** tab, click  beside the **Library** box, and select the Work library. In the Table box, enter *KeyStats*.

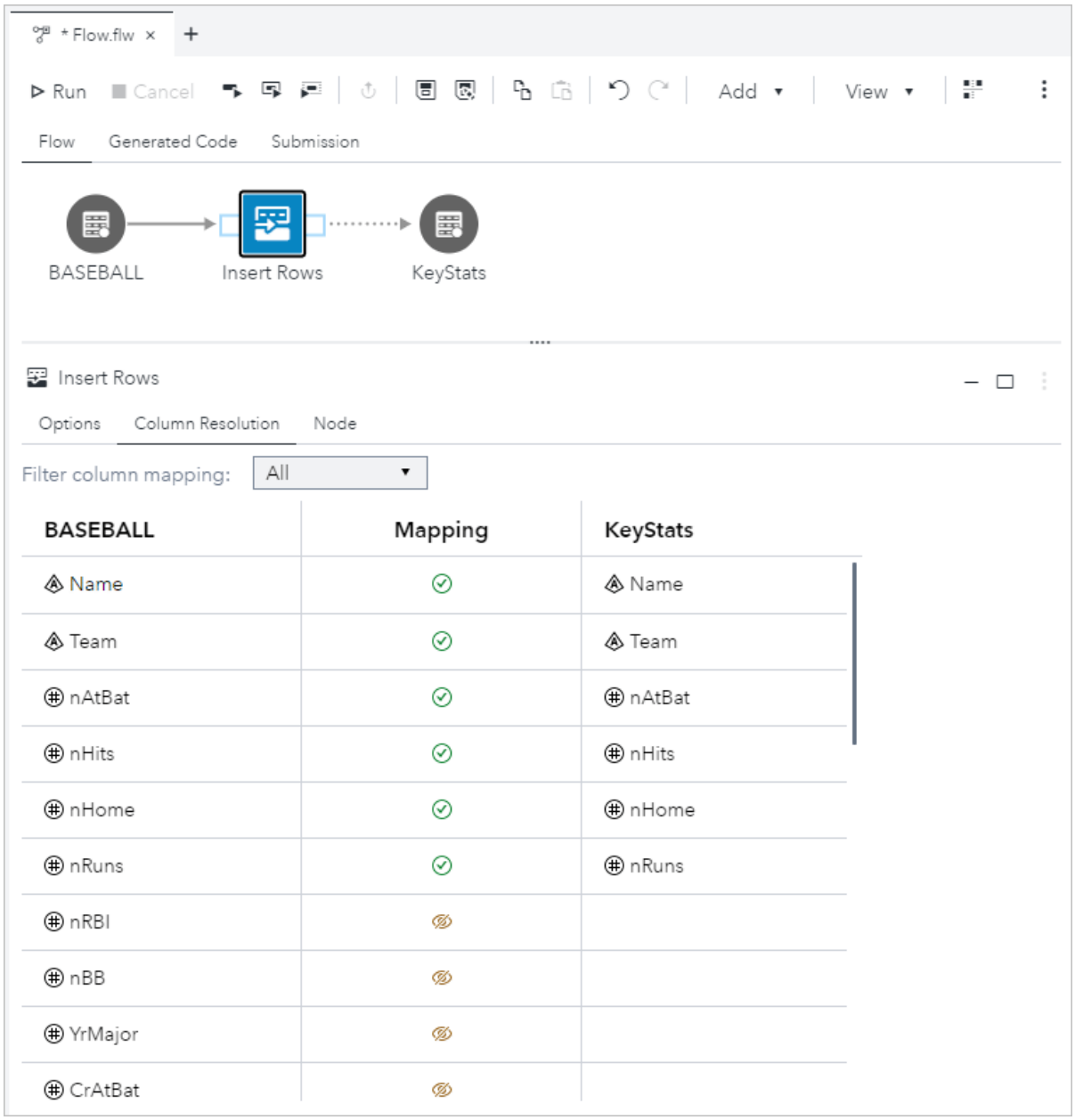
6 Click the **Published Columns** tab to add some columns from the Baseball data set. Click **New Column** and enter *Name* in the **Name** column. Accept the default data type of `Character` and enter *18* for the Length. Continue adding five more columns using these column names, data types, and lengths:

Column Name	Data Type	Length
Team	Character	14
nAtBat	Numeric	8
nHits	Numeric	8
nHome	Numeric	8
nRuns	Numeric	8

Note: For information about how to copy the entire column structure of the input table to the output table, see the Tip that is associated with Step 4 on page 139.

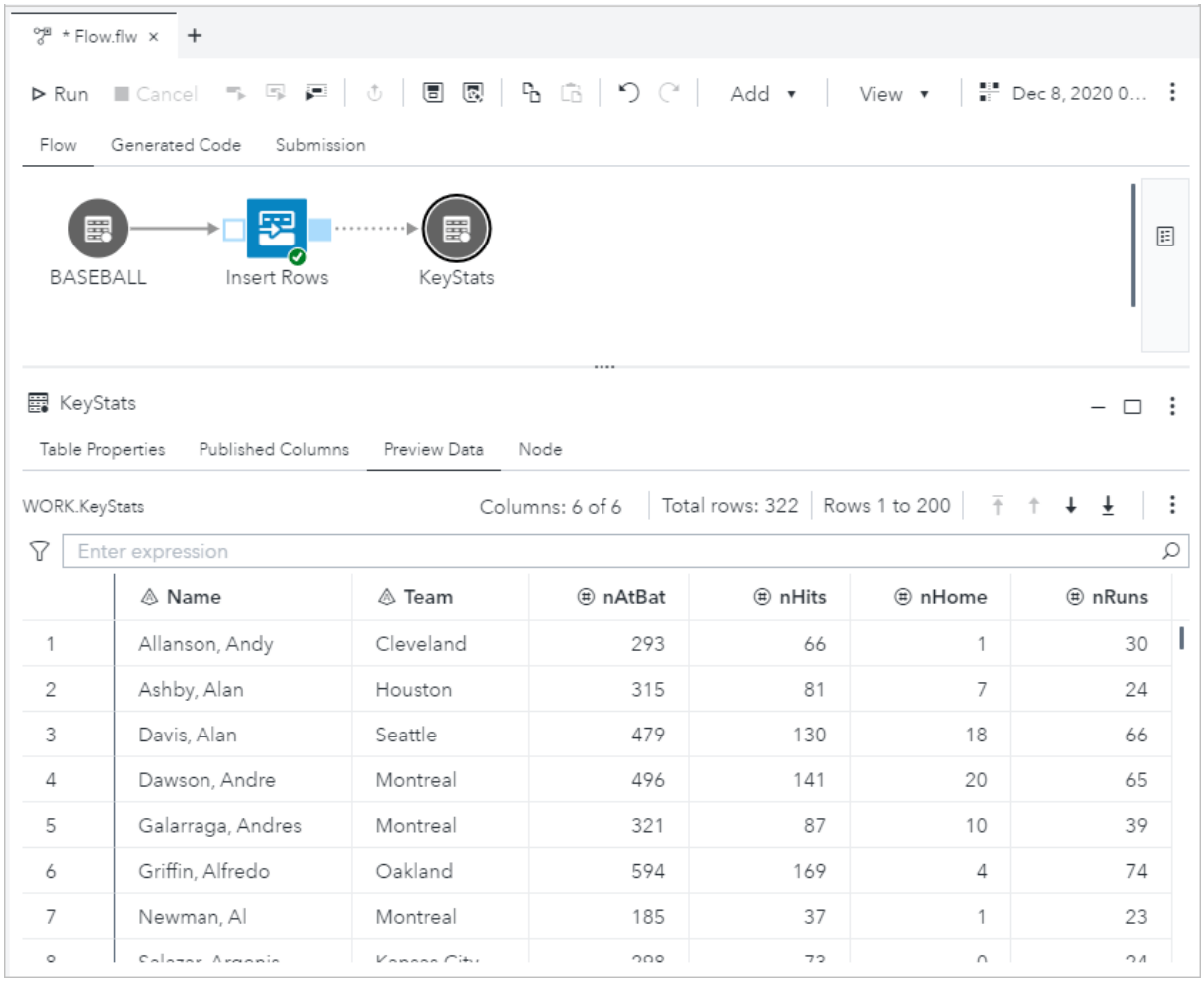


7 Click the **Insert Rows** node, and then click the **Column Resolution** tab. Notice that the columns in the KeyStats table are all mapped to columns in the Baseball data set. There are additional columns in the Baseball data set that do not map to columns in the KeyStats table.



8 To run the flow, click ▷ Run.

Here is the KeyStats output table:

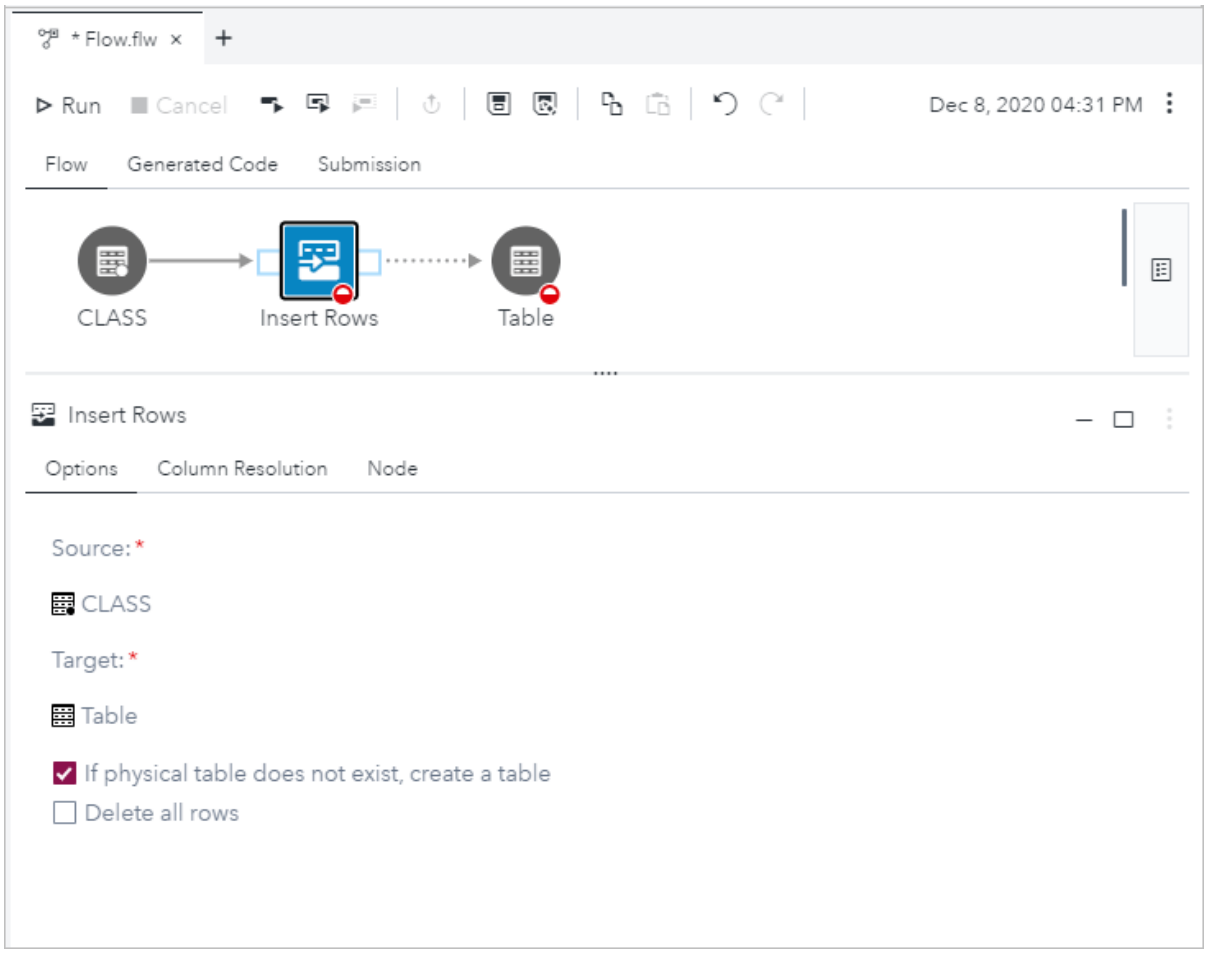


	Name	Team	nAtBat	nHits	nHome	nRuns
1	Allanson, Andy	Cleveland	293	66	1	30
2	Ashby, Alan	Houston	315	81	7	24
3	Davis, Alan	Seattle	479	130	18	66
4	Dawson, Andre	Montreal	496	141	20	65
5	Galarraga, Andres	Montreal	321	87	10	39
6	Griffin, Alfredo	Oakland	594	169	4	74
7	Newman, Al	Montreal	185	37	1	23

Insert Rows: Step-by-Step Instructions

By default, the Insert Rows step appends rows from an input table to the end of the output table. You can also add rows to a new table or choose to delete any existing rows in the output table before you insert the new rows.

1. In the **Steps** section of the navigation pane, expand the **Transform Data** folder and double-click **Insert Rows**.
2. Connect the input port of the Insert Rows node to a data source by using a Table node or an operational node that creates an output table, such as a Query node, a SAS Program node, or an Import node. For more information, see “Connecting Nodes” on page 108.
3. Connect the output port of the Insert Rows node to a Table node. The Table node must include at least one column with the same name and data type as a column in the input table. The Table node can reference an existing table, or you can create a new table when the step runs.



4 Click the output table node, and use the Table Properties and Published Columns tabs to make sure that the node specifies a library and name for the table and that the node includes one or more columns from the input table. For more information, see "Adding a Table from a SAS Library to a Flow" on page 114.

Note: If the length of a character column in the output table does not match the length of the corresponding column in the input table, the data is truncated when it is inserted in the output table.

TIP When you are inserting columns into a new output table, you can define the columns in the output table manually on the Published Columns tab of the table node, or you can use the following steps to copy the column structure of the input table:

1. Click the output table node in the flow, and then click the **Table Properties** tab.
2. Change the values of the **Library** and **Table name** fields to specify the library and table name of the input table.
3. Click the **Published Columns** tab, and then click **Sync Structure** to synchronize the column structure of the output table with the input table. Click **Yes** in the confirmation window. You can delete any columns that you do not need.
4. Click the **Table Properties** tab again, and change the values of the **Library** and **Table name** fields back to the values of the new output table.

5. Click the **Insert Rows** node.
   - On the Options tab, use the **If physical table does not exist, create a table** option to create an output table if the table does not already exist. This option is selected by default. If you clear this option, and the table does not already exist when you run the flow, the flow fails with an error.
   - If you want to delete all of the existing rows in the output table before the rows from the input table are inserted, select **Delete all rows**. This option is not selected by default.
   - On the Column Resolution tab, you can view the column mappings between the input and output tables. Use the **Filter column mapping** drop-down list to filter the mappings that are displayed. You can filter the mappings by the following categories:
     - Successful (✓) - the input column is successfully mapped to a column in the output table. Data for this column is inserted in the corresponding column in the output table.
     - Ignored (⊘) - the input column does not have a corresponding column in the output table. Data for this column is ignored in the output table.
     - Unresolved (⚑) - the input column cannot be mapped to a column in the output table. For example, an unresolved column can occur when columns have the same name and different data types or when a column exists in the output table and not the input table. The step cannot run with an unresolved column.
6. To run the flow, click ▷ Run.

Creating a Query in a Flow

Use the Query node to select, join, filter, and sort columns from a table in a flow. By default, output for the Query node is written to a table in the Work library. Query node output ports can be connected to a Table node, a Query node, or a SAS Program node.

> **TIP** You can also use the Branch Rows step and the Filter Rows step to select a subset of rows from an input table. For more information, see "Understanding the Steps That Subset Data" on page 122.

The Query node interface is mostly the same as the interface for stand-alone queries. For more information, see Chapter 3, "Working with Queries," on page 69.

To add a Query node from a flow toolbar:

1. Select **Add** ⇨ **Query**.
2. Connect the Query node to a data source by using a Table node, an Import node, a Query node, or a SAS Program node.

   > **TIP** You can quickly add a Query node to the flow and connect the node to an input data source by right-clicking the input data source and selecting **Add a query**. Valid input data sources include a Table node or the output ports of the operational nodes, including Import, SAS Program, and Query.

3. Click the **Query** node, click the **Options** tab, and then select output columns.

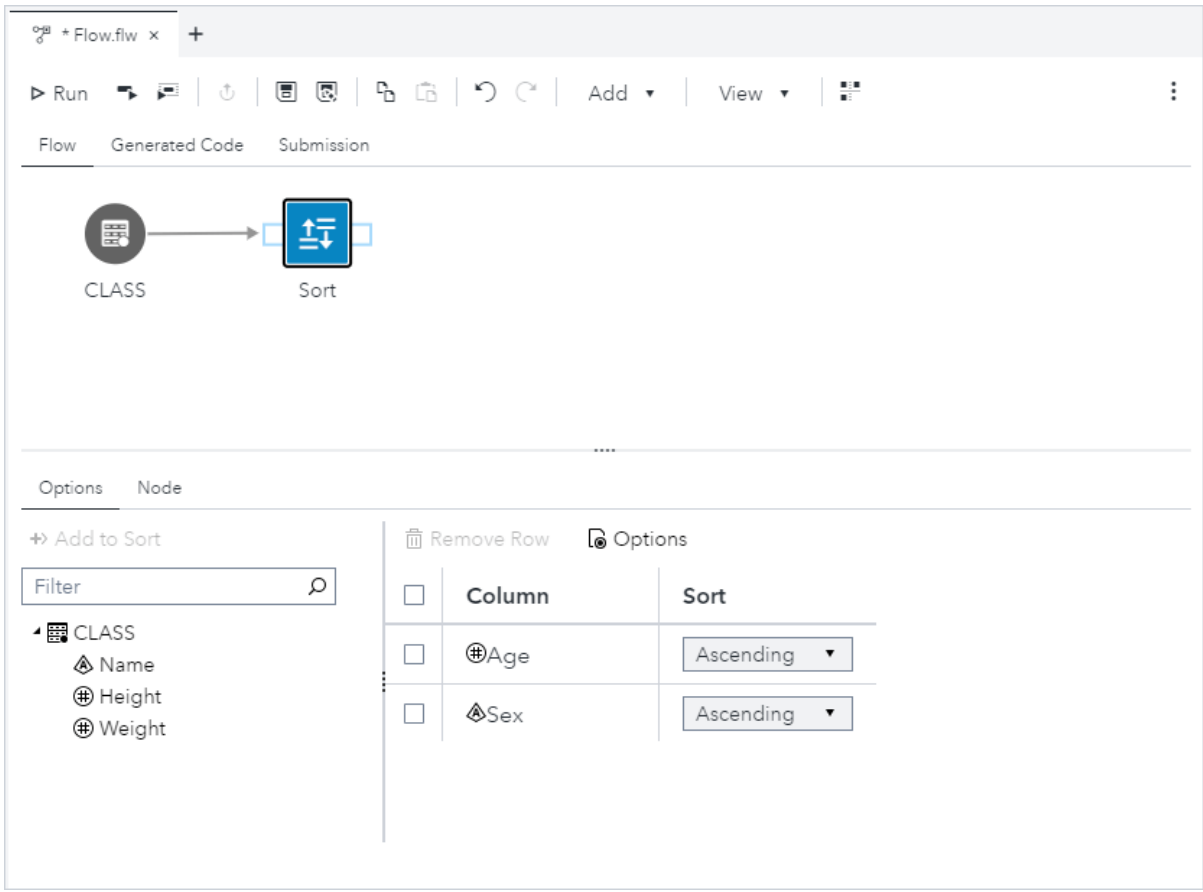
Sorting Data

The Sort node enables you to order your data by the values of one or more columns. By default, output for the Sort node is written to a table in the Work library. Sort node output ports can be connected to Table, Query, and SAS Program nodes.

To sort a table:

1. Click the **Steps** section of the navigation pane.
2. Expand the **Transform Data** folder and double-click the **Sort** node.
3. From the **Libraries** section of the navigation pane, drag the table that you want to sort onto the **Sort** node. When the tooltip changes to `Connect to input port`, drop the file.
4. Click the **Sort** node, and use the **Options** tab to add one or more columns. To add a column, click the column that you want to add to the sort, and then click **Add to Sort**. The order in which the columns are listed on the **Options** tab

determines which variable is the primary sort key, which variable is the secondary sort key, and so on. The primary sort key is always the first variable in the list.



5 To specify the following optional arguments to the sort criteria, click **Options** on the toolbar.

- **Specify the output order** - specifies the order of the rows in the output data set.
- **Duplicate rows** - specifies whether to keep all rows that are in the output table, including duplicate rows.
- **Force redundant sorting** - sorts and replaces the data set and destroys all user-created indexes for the data set.
- **Use tags to sort large data** - reduces temporary disk usage by storing only the BY variables and observation numbers in temporary files.

Click **OK** to save your changes.

Optimizing Steps in a Flow

When you use the output of an operational node as the input to another node, it might not be necessary to create and store the output table from the first node. You

can optimize the performance of your flow by combining the code generation of adjacent nodes so that the table that is associated with the output port of the first node is not created.

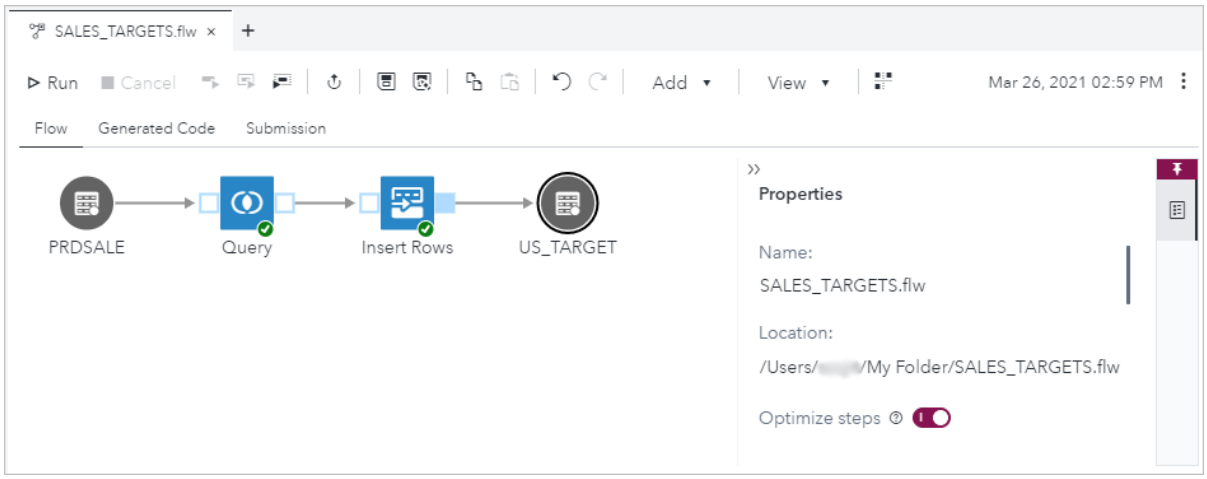
Note: The optimization option works only when the nodes that can be optimized are run at the same time.

For example, you could use the Query step to calculate local sales data that you want to append regularly to a regional sales table by using the Insert Rows step. You can optimize the performance of your flow by combining the code that is generated by the Query and Insert Rows nodes. When the code for the Query and Insert Rows nodes is combined, the output table for the Query node is not explicitly created.

To optimize the performance of your flow:

1 On the flow canvas, expand the Properties pane by clicking .

2 Click to turn on the **Optimize steps** option.



Running a Flow

To run all the nodes in a flow, click Run.

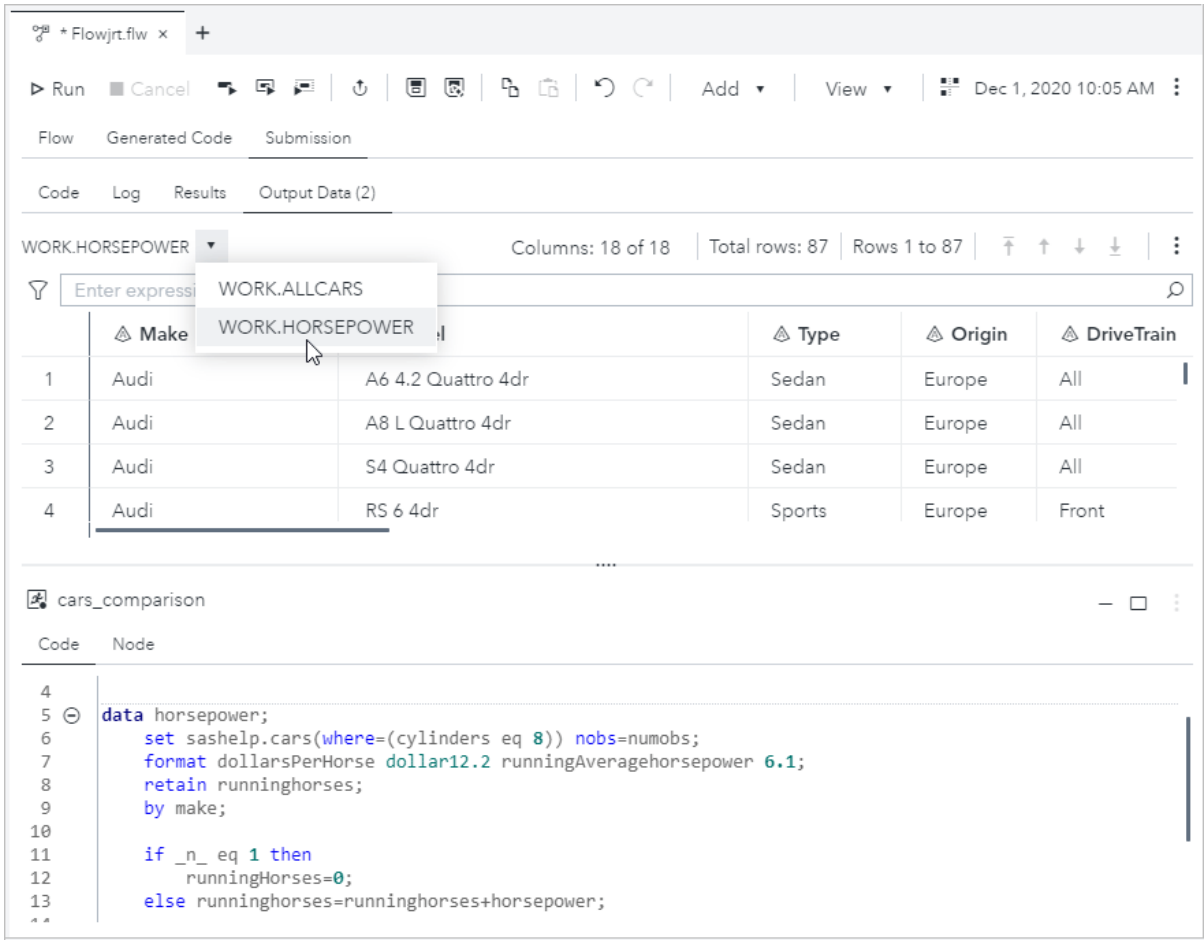
You can also run a subset of the flow by using the following buttons on the flow toolbar:

- - runs only the currently selected node in the flow.
- - runs the currently selected node and all nodes that occur downstream from the currently selected node.
- - runs all nodes that occur upstream from the currently selected node. The currently selected node is not run.

Note: To cancel your flow, click **Cancel** on the toolbar. Currently processing nodes might take a few minutes to be canceled.

You can view the automatically generated code and log for a flow by clicking the **Generated Code** and **Submission** tabs on the flow canvas. The **Submission** tab also displays any results and output data that are generated when you run a node. If more than one output table is generated, you can use the output table drop-down list to select which table to display.

TIP You can interact with a flow while it is running by scrolling the flow canvas and by clicking nodes and tabs on the flow canvas. You cannot modify the flow or any node details until the flow is finished running.



To view the code or log that is associated with a particular node, right-click the node and select **Go to code** or **Go to log**.

To run a saved flow as a background job, open the flow and click on the toolbar. For more information, see “Using the Background Submit Feature” on page 19.

or right-click the flow in the **Explorer** section of the navigation pane and select **Background submit**.

5

Working with Custom Steps

What Is a Custom Step?

A custom step enables you to create a user interface for users at your site to complete a specific task. Custom steps are saved to SAS Content, so they can be shared with others at your site.

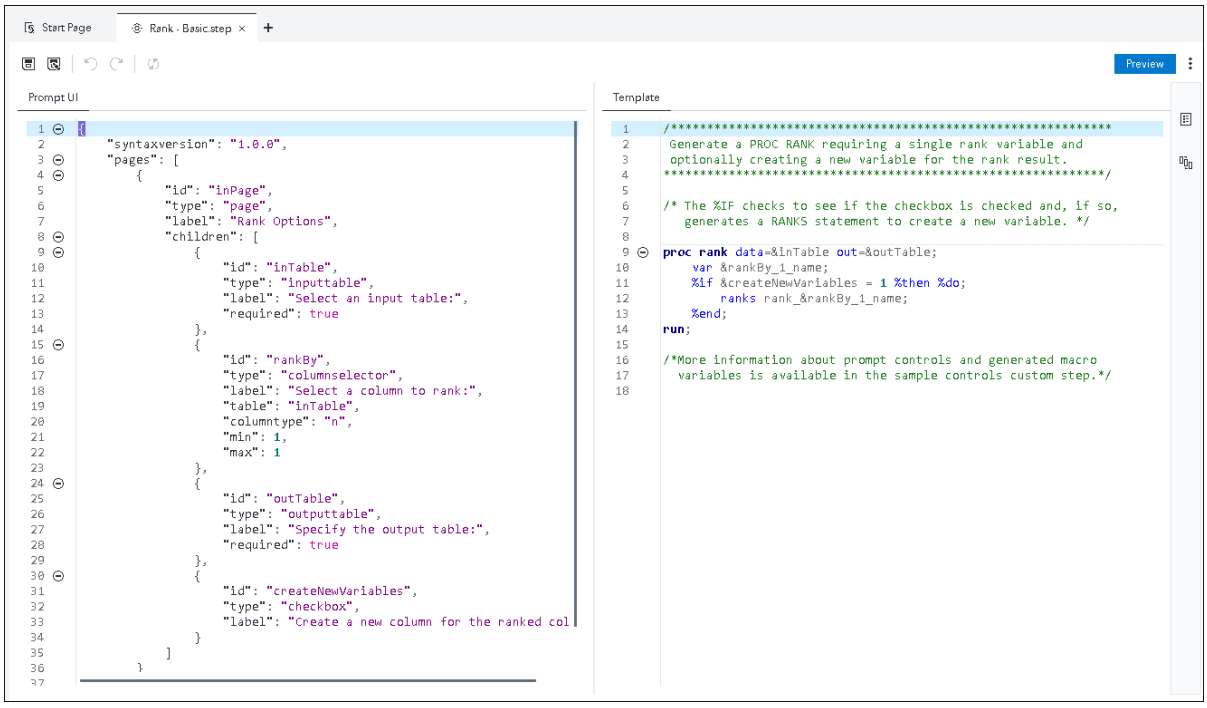
Custom steps are available in the Steps pane. To access the latest version of a custom step or new custom steps, click  .

Note: The Custom Step functionality is available only if your site licenses SAS Studio Analyst. Custom steps are not available in the Interactive perspective.

To view a sample custom step:

1. Open the Steps pane.
2. Click  and select **Starter templates** ⇨ **Basic - Rank**.

The Basic - Rank step opens in the **Rank - Basic.step** tab.



Here is an explanation of the code:

```
{
   "syntaxversion": "1.0.0",
       "pages": [
     {
         "id": "inPage",
```

```
            "type": "page",
            "label": "Rank Options",
                "children": [
               {
                  "id": "inTable",
                  "type": "inputtable",
                  "label": "Select input table:",
                  "required": true
               },
               {
                  "id": "rankBy",
                  "type": "columnselector",
                  "label": "Select a column to rank:",
                  "table": "inTable",
                         "columntype": "n",
                         "min": 1,
                         "max": 1
                       },
                       {
                          "id": "outTable",
                          "type": "outputtable",
                          "label": "Specify the output table:",
                          "required": true
                       },
                       {
                          "id": "createNewVariables",
                          "type": "checkbox",
                          "label": "Create a new column for the ranked
column"
                       }
            ]
         }
      ],
      5 "values": {}
```

1 The `syntaxversion` property specifies the semantic version of the code. To run this task, the end user must have this version of the code or later.

2 Pages can be used to organize controls into different categories. Pages appear as tabs in the user interface. You must define at least one page.

3 The first (and only) page in this sample is the **Rank Options** tab. The `label` property specifies the name of the tab as it appears in the user interface.

4 The **Rank Options** tab contains four controls:

- `inputtable`
- `columnselector`
- `outputtable`
- `checkbox`

For more information about each control, see “Syntax for Custom Steps” on page 167.

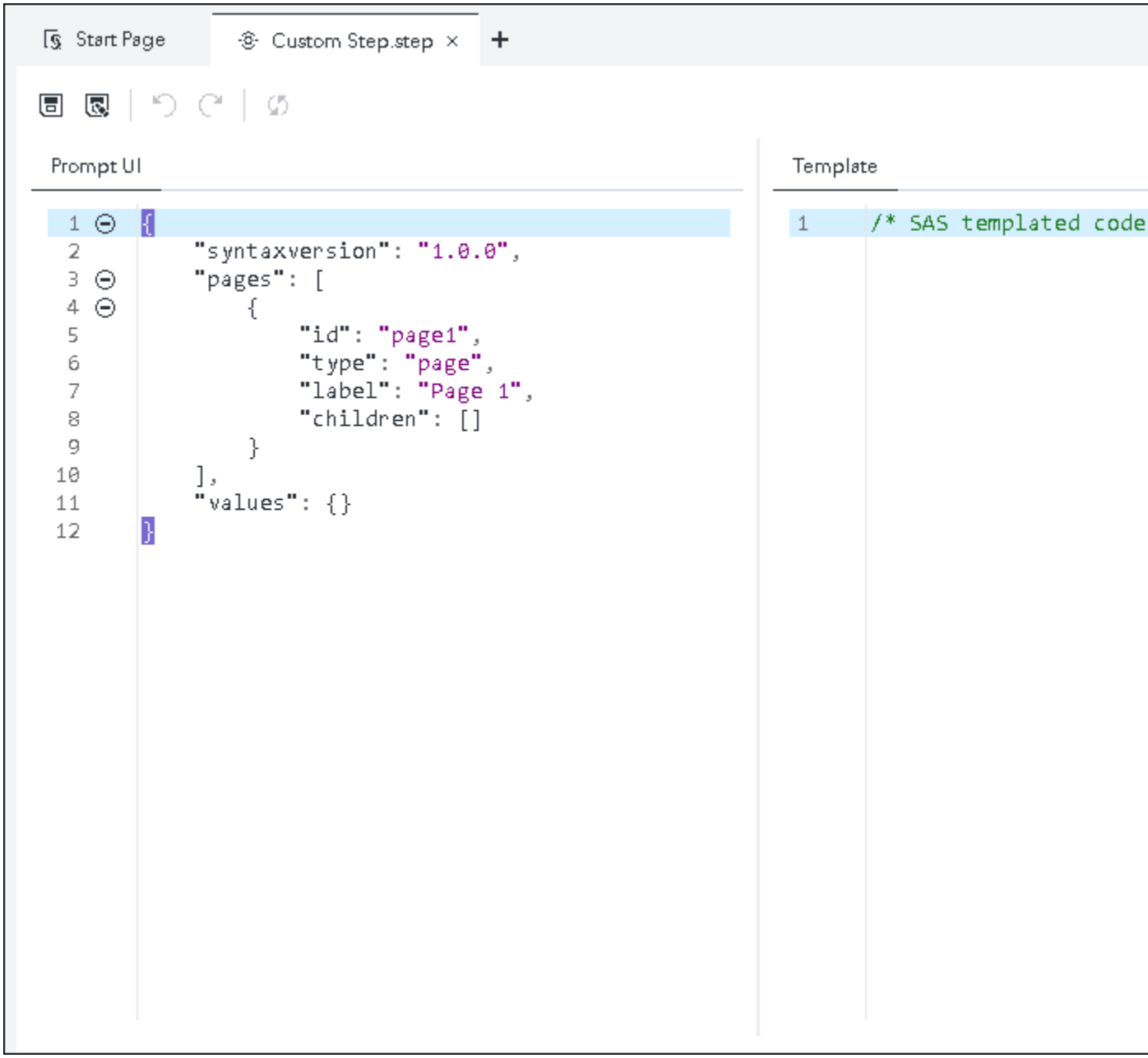
5 For some controls, you can specify the values that appear in the user interface or a default value. Those values would appear in this section. This simple example does not specify any values.

Create a Custom Step

To create a custom step:

1 In the Steps pane, click  and select **Custom step quick start**.

In the workspace, a **Custom Step.step** tab opens.



2 On the **Prompt UI** tab, enter the JSON syntax to create the prompting interface for your custom step. For the JSON syntax, see "Syntax for Custom Steps" on page 167.

This example shows the code from the Basic - Rank step.

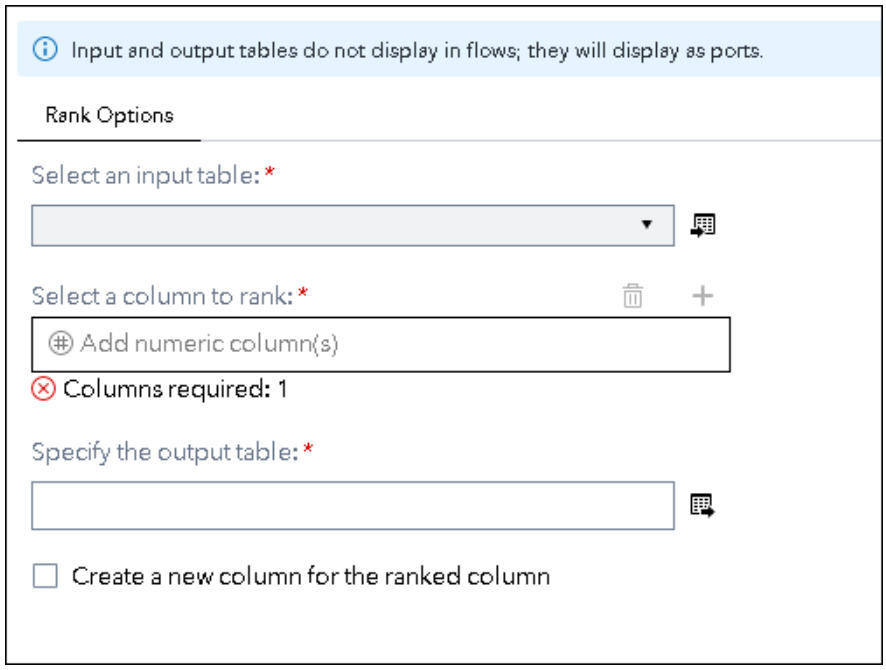
Prompt UI

```
{
    "syntaxversion": "1.0.0",
    "pages": [
        {
            "id": "inPage",
            "type": "page",
            "label": "Rank Options",
            "children": [
                {
                    "id": "inTable",
                    "type": "inputtable",
                    "label": "Select an input table:",
                    "required": true
                },
                {
                    "id": "rankBy",
                    "type": "columnselector",
                    "label": "Select a column to rank:",
                    "table": "inTable",
                    "columntype": "n",
                    "min": 1,
                    "max": 1
                },
                {
                    "id": "outTable",
                    "type": "outputtable",
                    "label": "Specify the output table:",
                    "required": true
                },
                {
                    "id": "createNewVariables",
                    "type": "checkbox",
                    "label": "Create a new column for the ranked col
                }
            ]
        }
```

3 (Optional) To define the metadata for input and output ports on table nodes in the flow, click  .

4 To validate that your code is correct and to view the controls in the user interface, click **Preview**. The user interface opens in a new tab in the workspace.

Here is the user interface Basic - Rank step.



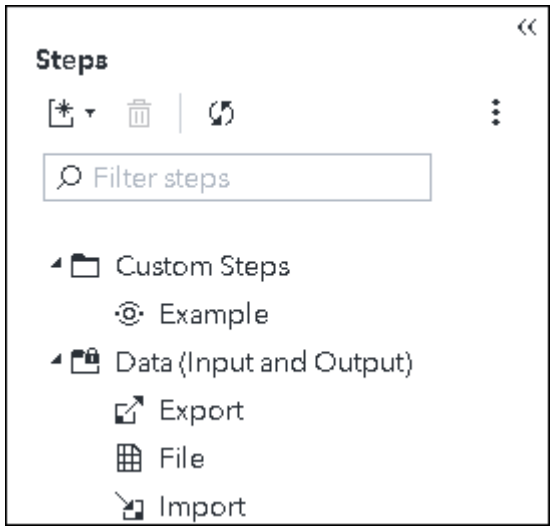
5 On the **Template** tab, add the SAS code that needs to run after the user enters the information in the prompt interface. Macro variables are generated for each control ID that you specify in the JSON code on the **Prompt UI** tab. Use the macro variables in the SAS code to reference the corresponding prompt.

Template

```
1  /*************************************************************
2   Generate a PROC RANK requiring a single rank variable and
3   optionally creating a new variable for the rank result.
4  *************************************************************/
5
6  /* The %IF checks to see if the checkbox is checked and, if so,
7     generates a RANKS statement to create a new variable. */
8
9  proc rank data=&inTable out=&outTable;
10     var &rankBy_1_name;
11     %if &createNewVariables = 1 %then %do;
12         ranks rank_&rankBy_1_name;
13     %end;
14 run;
15
16 /*More information about prompt controls and generated macro
17   variables is available in the sample controls custom step.*/
18
```

6 To save the custom step to SAS Content, click  or  .

The new custom step (called Example in the following screenshot) appears in the Custom Steps folder in the Steps pane.



Specify Port Details for Input and Output Tables in a Flow

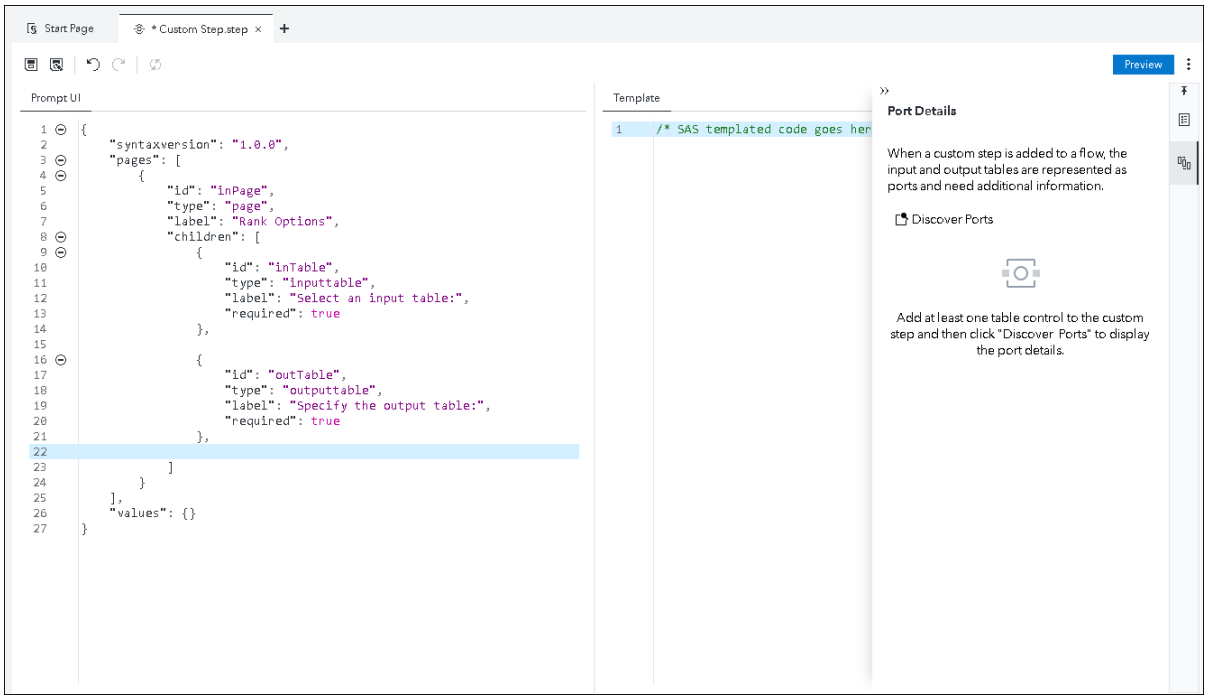
When you add a custom step to a flow, the controls for the inputtable and outputtable are represented as ports. By default, the port name for input and output tables is the ID that you specified in the JSON code.

Follow these steps to understand port details for input and output tables:

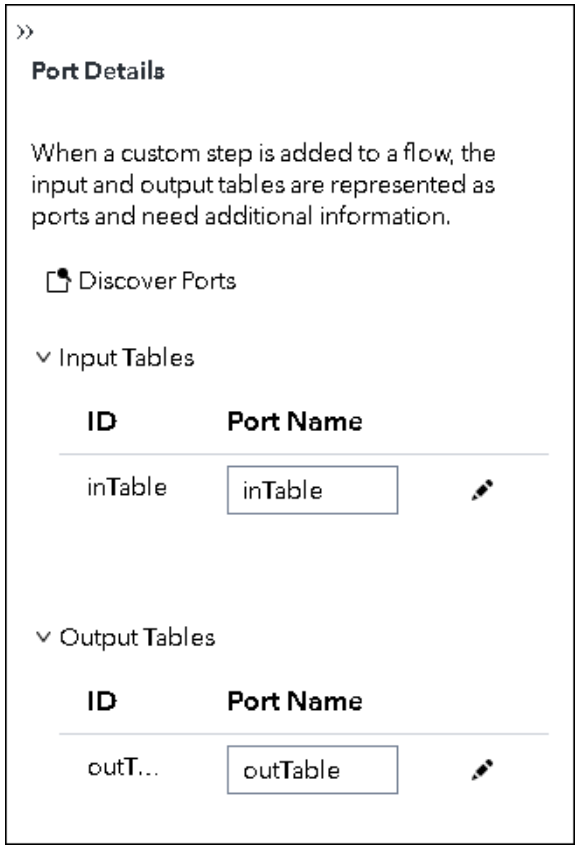
1 On a new **Custom Step.step** tab, add this code:

```
"syntaxversion": "1.0.0",
    "pages": [
        {
            "id": "inPage",
            "type": "page",
            "label": "Rank Options",
            "children": [
                {
                    "id": "inTable",
                    "type": "inputtable",
                    "label": "Select an input table:",
                    "required": true
                },
                {
                    "id": "outTable",
                    "type": "outputtable",
                    "label": "Specify the output table:",
                    "required": true
                }
```

2 Click  to open the Port Details pane.



3 To view the input and output controls that are defined in the JSON code, click **Discover Ports**.

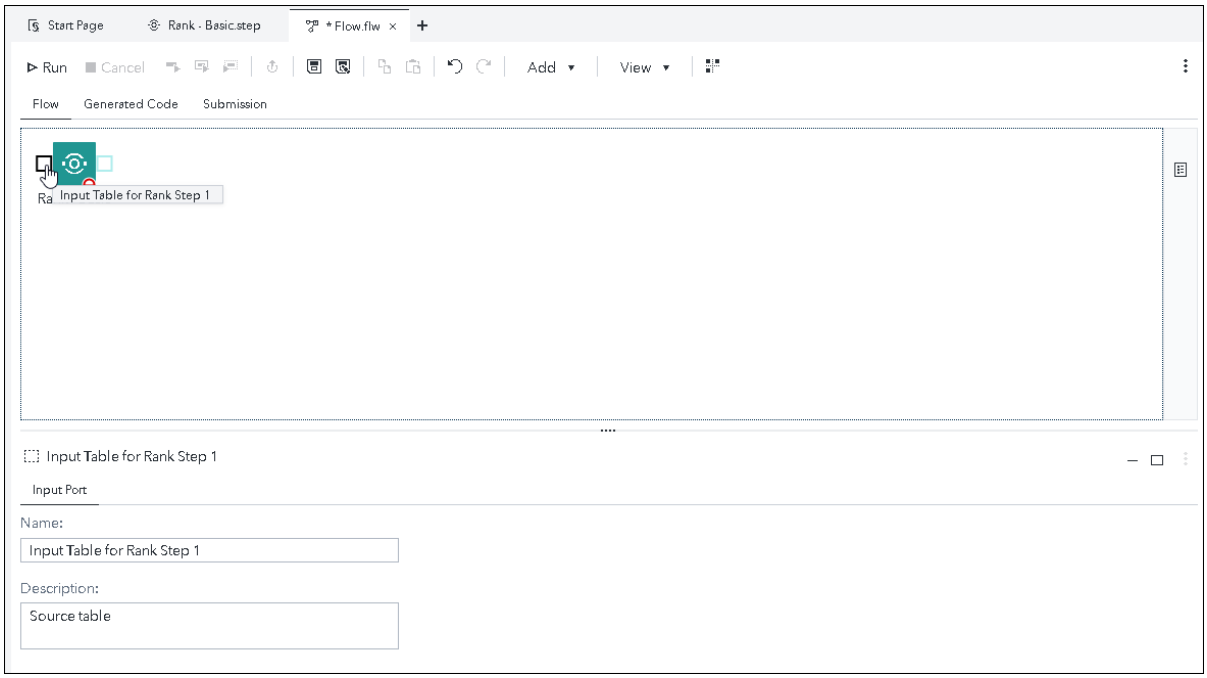


The port name is the same as the ID that you defined for the control in the JSON code. This is the name that appears for the port in the flow. You might want to change the name to make it more user friendly.

4 To change the port name for the input or output table, enter the new name in the **Port Name** field.

To change the name and add a description, click .

After the custom step has been added to a flow, the port name appears when you hover your mouse pointer over the input port for the custom step.



Example: Create a Simple Rank Data Task

This example creates the simple rank data task. To access the code for this task, click and select ⇨ .

Step 1: Create the Rank Options Tab with Four Controls

In this part of the example, you create the **Rank Options** tab that contains the four controls for this task.

1 In the Steps pane, click and select **Starter templates** ⇨ **Basic - Rank**. The JSON code for the Basic - Rank step opens on the **Rank - Basic.step** tab.

By default, the custom step includes JSON code for one page. In this code, page ID is inPage. In the user interface, this page is labeled Rank Options.

```
{
    "syntaxversion": "1.0.0",
    "pages": [
        {
            "id": "inPage",
            "type": "page",
            "label": "Rank Options",
            "children": []
        }
    ],
    "values": {}
}
```

2 To add a field for specifying the input data source, add the following code:

```
{
    "syntaxversion": "1.0.0",
    "pages": [
        {
            "id": "inPage",
            "type": "page",
            "label": "Rank Options",
            "children": [
              1{
                2"id": "inTable",
                3"type": "inputtable",
                4"label": "Select an input table:"
              5  "required": true
                6}
      ]
        }
    ],
    "values": {}
}
```

1 The new control is added to the children property. Add an opening curly bracket for the new child.

2 Specifies the id for the new control. IDs must be unique. For this example, the id is inTable.

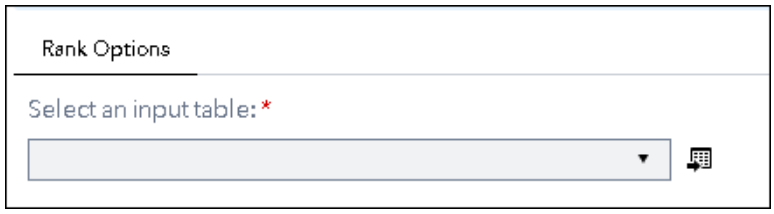
3 For the type, enter **`inputtable`**.

4 For the label property, enter the text that helps the user know what to do with this field. In the label, enter **`Select an input table:`**.

5 To specify that a value is required for the inputtable control, enter **`"required": true`**. If you do not specify the required parameter, a blank value is considered a valid value and defaults to `false`.

6 Specify a closing curly bracket to end the code for the input table control.

Click **Preview** to view the **Rank Options** tab with the input table control.



3 To add a field where the user can select the columns to rank, add the following code:

```
{
    "syntaxversion": "1.0.0",
    "pages": [
        {
            "id": "inPage",
            "type": "page",
            "label": "Rank Options",
            "children": [
         {
            "id": "inTable",
            "type": "inputtable",
            "label": "Select an input table:"
                    "required": true
          1 },
          2 {
              3 "id": "rankBy",
              4 "type": "columnselector",
              5 "label": "Select a column to rank:",
              6 "table": "inTable",
                      7 "columntype": "n",
                     8 "min": 1,
                      "max": 1
          }
      ]
        }
    ],
    "values": {}
}
```

1 Because you are creating a second child for the page, add a comma after the closing curly bracket for the input table control.

2 Add an opening curly bracket for the new child.

3 Specifies the id for the new control. IDs must be unique. For this example, the id is rankBy.

4 For the type, enter **`columnselector`**.

5 For the label property, enter the text that helps the user know what to do with this field. In the label, enter **`Select a column to rank:`**.

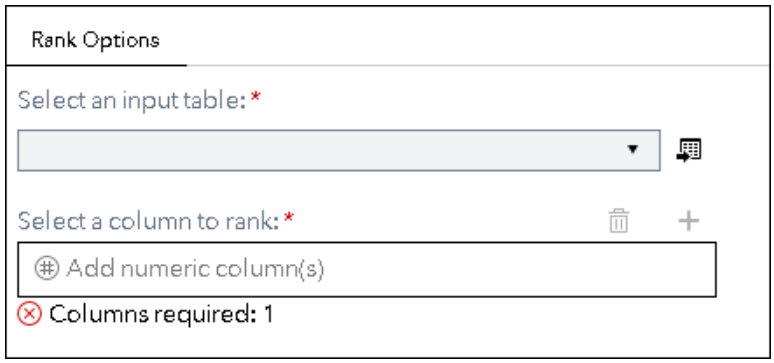
6 The columnselector control requires the table property so that the control knows what table contains the columns for this field. For this example, specify the id for the inputtable control (**`inTable`**).

7 The `columntype` property specifies the column types that are accepted by this field. In this example, the `n` value specifies that only numeric columns are allowed.

8 With the `columnselector` control, you specify whether any columns are required and the maximum number of columns that the user can select. In this example, the `min` value specifies that one column is required. The `max` value specifies that only one column can be selected.

9 Specify a closing curly bracket to end the code for the columnselector control.

Click **Preview** to view the **Rank Options** tab with the new column selector field.



4 To specify a field where the user can specify the name of the output table, add the following code:

```
{
    "syntaxversion": "1.0.0",
    "pages": [

        ...

                {
                    "id": "rankBy",
                    "type": "columnselector",
                    "label": "Columns to rank:",
                    "table": "inTable",
                    "columntype": "n",
                    "min": 1,
                    "max": 1
                1},
                2{
                    3"id": "outTable",
                    4"type": "outputtable",
                    5"label": "Specify the output table:",
                    6"required": true
                }
```

1 Because you are adding another child to the page, add a comma after the closing curly bracket for the columnselector control.

2 Opening the curly bracket for new control.

3 Specifies the id (outTable) for the new control.

4 For the type, enter **`outputtable`**.

5 For the label property, enter the text that helps the user know what to do with this field. In the label, enter **`Specify the output table:`**.

6 To specify that a value is required for the outputtable control, enter **`"required": true`**.

5 To create a checkbox to specify whether a new variable is created for the ranked variable, add the following code:

```
...
        {
      "id": "outTable",
            "type": "outputtable",
            "label": "Specify the output table:",
            "required": true
        1 },
       2 {
       3 "id": "createNewVariables",
            4 "type": "checkbox",
            5 "label": "Create a new column for the ranked column"
        6 }
    ...
```

1 Because you are adding values for a different control, add a comma after the closing curly bracket for the outputtable control.

2 Add an opening curly bracket for the new checkbox control.

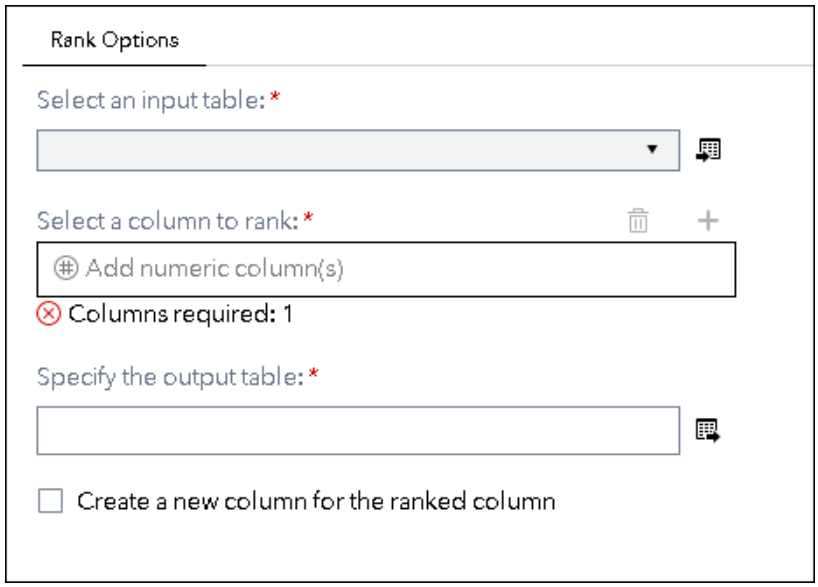
3 Specifies the id (createNewVariables) for the new control.

4 For the type, enter **`checkbox`**.

5 For the label property, enter the text that helps the user know what to do with this field. In the label, enter **`Create a new column for the ranked column`**.

6 Enter a closing curly bracket.

Click **Preview** to view all the controls onn the **Rank Options** tab.



Step 2: Adding Your SAS Code

Now that you have designed the user interface, you need to add the SAS code that will rank the data. In the SAS code, use the control IDs to reference macro variables that will contain the control value after it is specified.

On the **Template** tab, add the following SAS code:

```
/************************************************************
 Generate a PROC RANK requiring a single rank variable and
 optionally creating a new variable for the rank result.
************************************************************/

/* The %IF checks to see if the checkbox is checked and, if so,
   generates a RANKS statement to create a new variable. */

1 proc rank data=&inTable out=&outTable;
    var &rankBy_1_name;
 2  %if &createNewVariables = 1 %then %do;
        ranks rank_&rankBy_1_name;
    %end;
run;

/*More information about prompt controls and generated macro
  variables is available in the sample controls custom step.*/
```

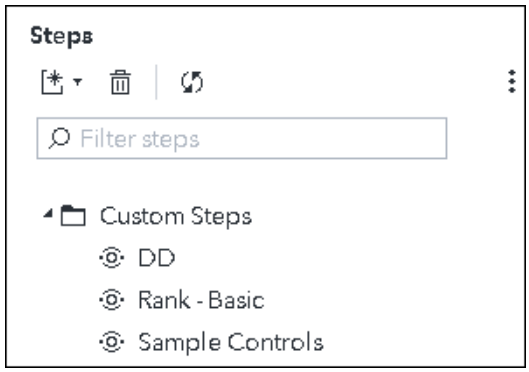
1 The &inTable and &outTable macro variables contain the table value of the table node in the flow.

2 The &createNewVariables macro variable contains the value of the createNewVariables (checkbox) control. A checkbox control can either be checked (true or 1) or unchecked (false or 0).

Step 3: Save the Rank Data Step

The code for your new custom step is complete. To save this custom step, click  . You want to save this step to a folder in SAS Content. Enter **`Rank - Basic`** as the name of the step.

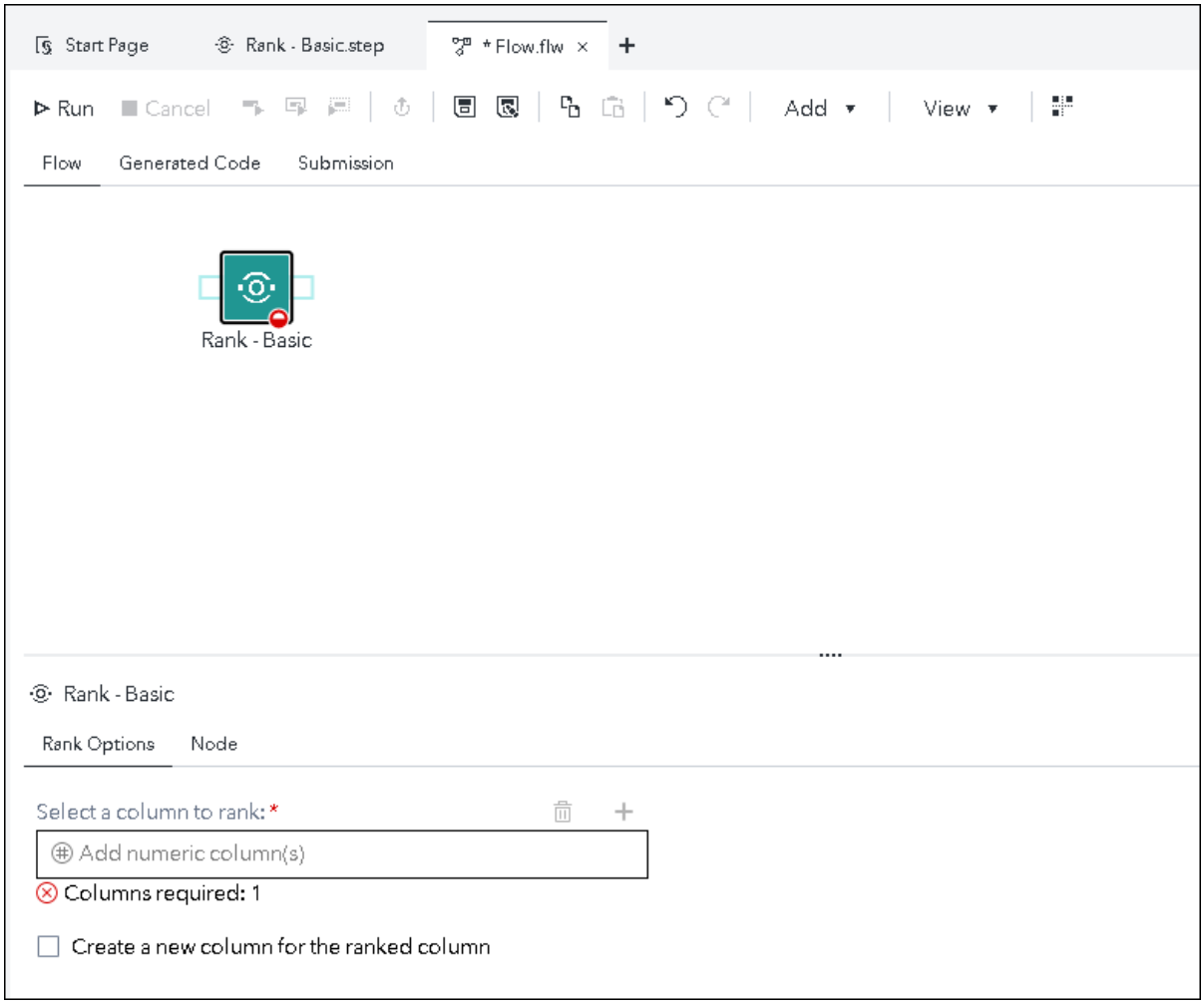
After you click **Save**, the Rank - Basic step appears in the Custom Steps folder in the Steps pane.



Step 4: Add the Rank - Basic Step to a Flow

To add the Rank - Basic step to a flow:

1. From the main menu, select **New** ⇨ **Flow**. A blank flow opens.
2. To add the Rank Data step to the flow, drag and drop the Rank - Basic step from the Steps pane.

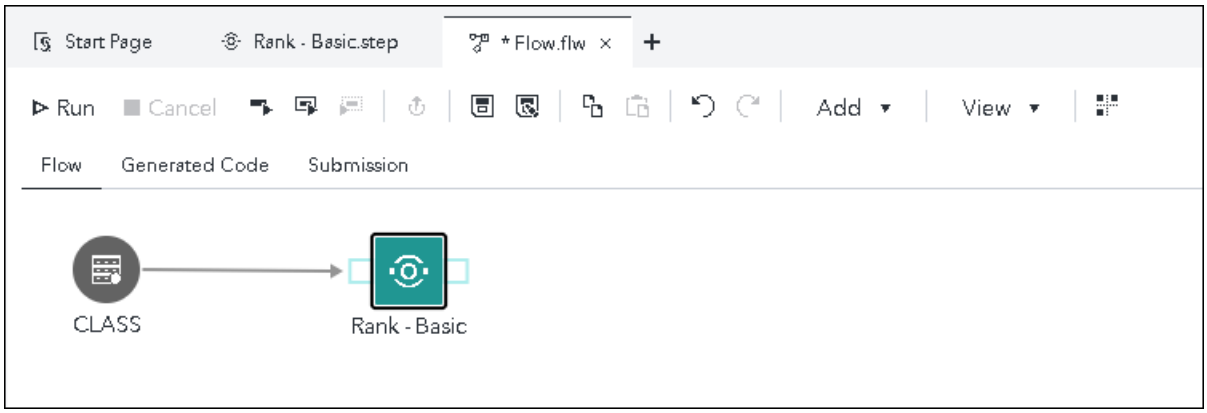


At the bottom of the canvas, you see part of the user interface that you defined in the JSON code.

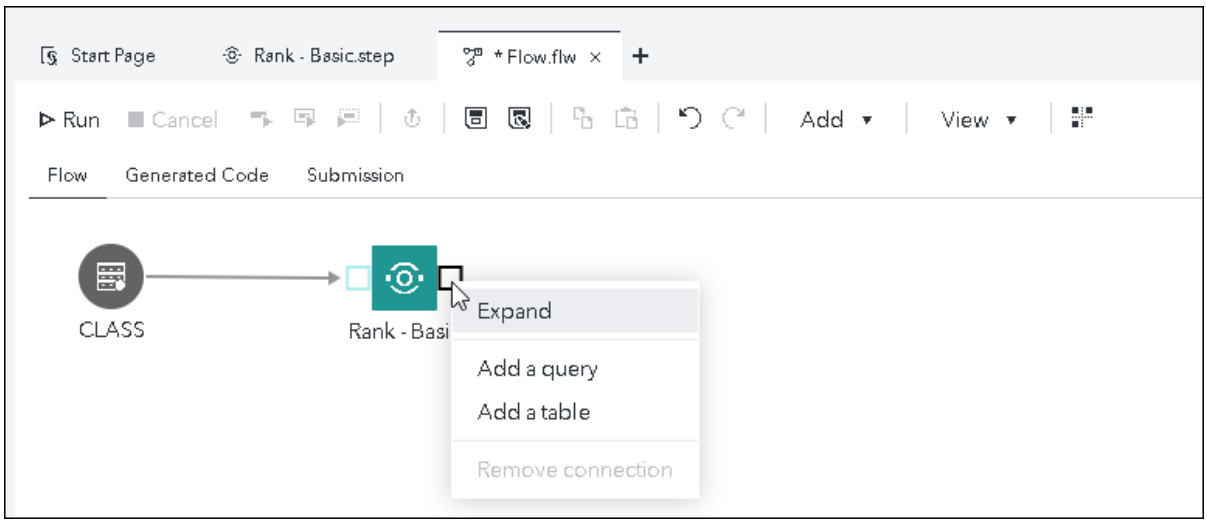
However, the controls for the input table and output table are missing. Because this step is part of a flow, you want the flow to determine the values of the input data and output data.

3 Add the SASHELP.CLASS data source to your flow.

4 To specify that SASHELP.CLASS is the input data source, create an arrow from the CLASS node to the input port for the Rank - Basic node.

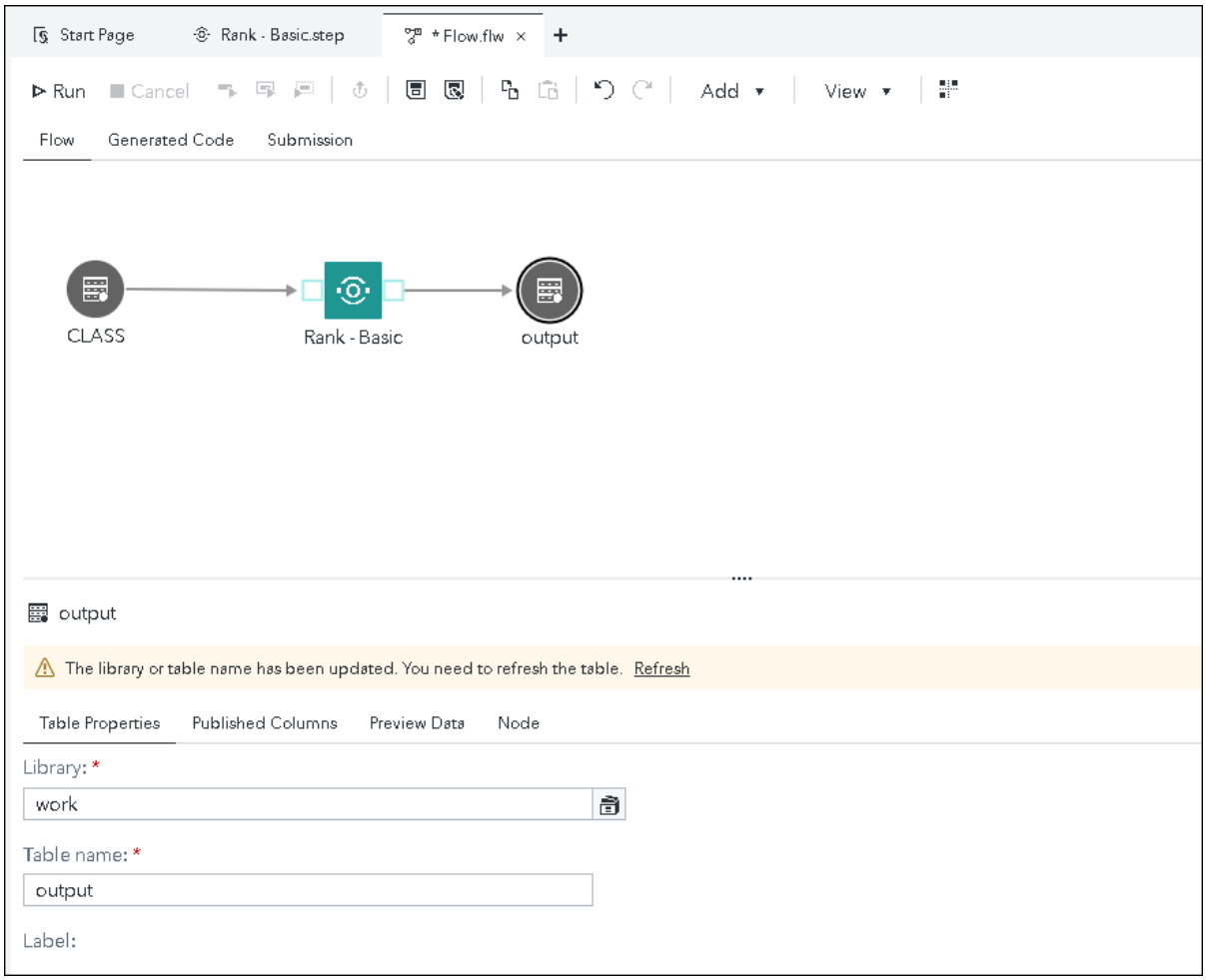


5 To create an output table, right-click the output port of the Rank - Basic node and select **Add a table**.



A new Table node is added to the flow.

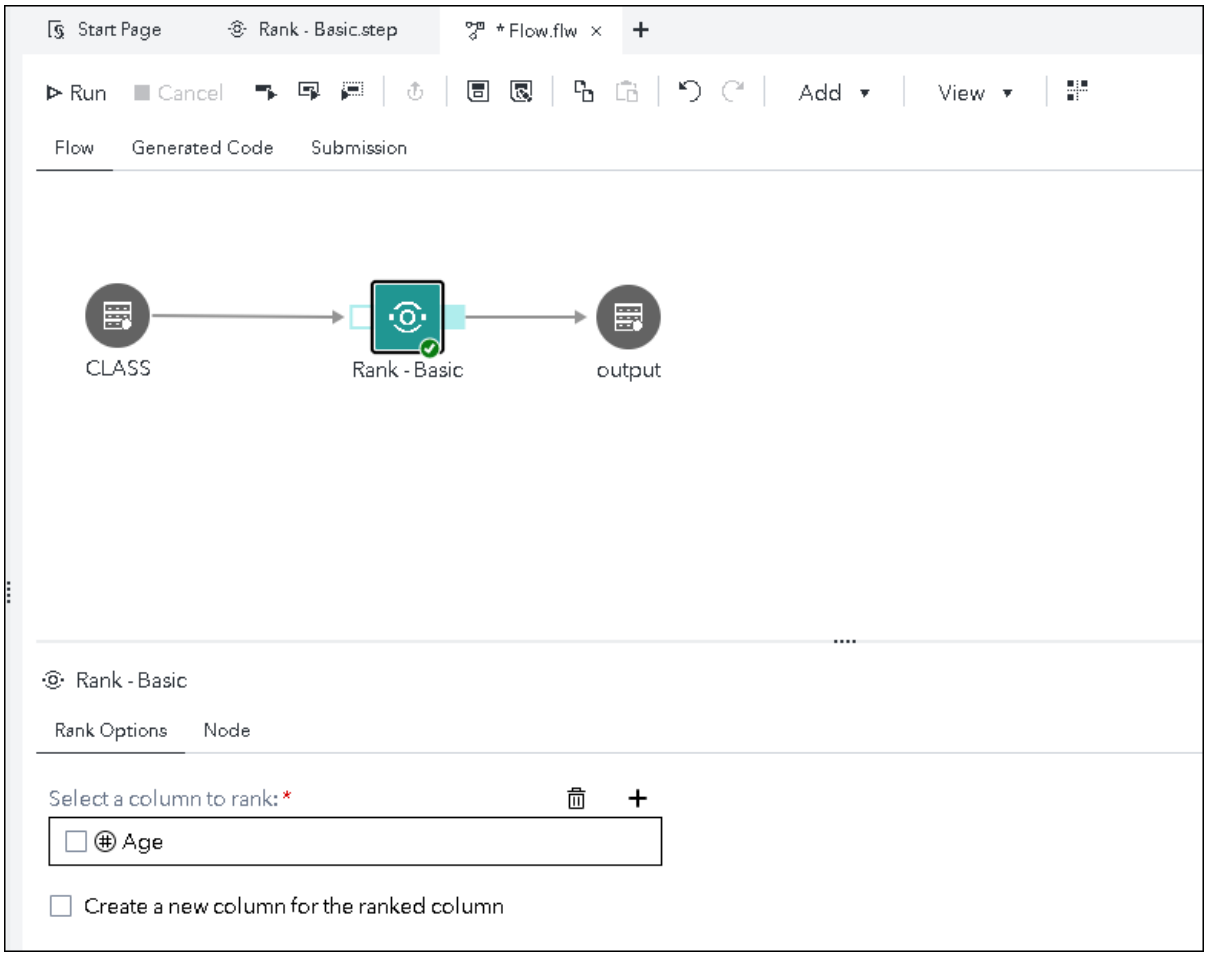
6 In the properties for the new table node, specify the library and name for the output table. For this example, enter `work` as the library and `output` as the table name.



Step 5: Run the Rank - Basic Data Node

1. In the flow, click the Rank - Basic Data node to view the columnselector and checkbox controls that you created.
2. For the **Select a column to rank** option, click [+].
3. From the Column Selection window, select **Age** and click **OK**.
4. To run the flow, click **Run**.

If the run is successful, you see a green check on the Rank - Basic node.



Example: Create an Advanced Rank Step

This example creates a more advanced Rank Data step. The Rank - Advanced step contains the four controls from the Rank - Basic step, but it also includes a dropdown list control. This step shows how you can specify the values for controls, such as the dropdown control. It also shows how you can loop through multiple columns to be ranked.

To access the code for this task, click  and select **Starter templates** ⇨ **Advanced - Rank**.

Step 1: Add a Dropdown Control

In this part of the example, you add a dropdown control to the JSON code.

Here is the code for the dropdown control in the Rank - Advanced step.

```
...
{
   "id": "rankBy",
        "type": "columnselector",
        "label": "Select columns to rank:",
        "table": "inTable",
        "columntype": "n",
        "min": 1
    },
    {
   1"id": "ties",
       "type": "dropdown",
        "label": "Select how to handle tied values:",
       2"items": [
          {
                "value": "High"
            },
            {
              "value": "Low"
            },
            {
               "value": "Mean"
            },
            {
               "value": "Dense"
            }
   ],
   3"required": false
    },
    {
       "id": "outTable",
        "type": "outputtable",
        "label": "Specify the output table:",
        "required": true
    },
...
```

1 The dropdown control has an ID of ties. In the user interface, this control is labeled **Select how to handle tied values**.

2 Use the items property to specify the values that appear in the drop-down list. The **Select how to handle tied values** dropdown control has these values: **High**, **Low**, **Mean**, and **Dense**.

Notice the opening and closing braces surrounding the lists of values.

3 A value is not required for the dropdown control.

Step 2: Specifying Default Values for Several Controls

This step shows how to specify default values for two controls in the Advanced - Rank step.

Here is the last section of code for the Advanced - Rank step.

```
...
1 ],
    2 "values": {
        3 "ties": {
            "value": "Mean"
        },
        4 "createNewVariables": true
 5    }
6 }
```

1 After the last closing square bracket for the pages, add a comma.

2 Specifies the start of the values section.

3 In the JSON code, you defined the **Select how to handle tied values** drop-down list and used the items property to specify the values that would appear in the drop-down list.

```
{
"id": "ties",
     "type": "dropdown",
     "label": "Select how to handle tied values:",
     "items": [
        {
            "value": "High"
         },
         {
            "value": "Low"
         },
        {
            "value": "Mean"
         },
   {
            "value": "Dense"
        }
],
     "required": false
 }
```

In the values section, map the unique ID (ties) for the **Select how to handled tied values** drop-down list to the default value of **Mean**.

4 This value specifies that the default value for the **Create new columns for each of the ranked columns** check box is true (or checked).

5 Closing the curly bracket for the values section.

6 Closing the curly bracket for the JSON code.

Step 3: Adding Your SAS Code

Now that you have designed the user interface, you need to add the SAS code for the Advanced - Rank step.

Here is the code on the **Template** tab.

```
/************************************************************
PROC RANK for multiple columns and additional option.
************************************************************/

/*TIES is an example of an option that can obtain its value
  from the control but is initially set using the VALUES section
  so the resulting code is valid if no value is input by the user.*/

/*Example of using a macro to loop through a variable set of
  values from a control to build a list of values*/
%let rankVarList=;
%let newRankVar=;
%macro rankVars;
   %do i = 1 %to &rankBy_count;
      %let rankVarList= &rankVarList &&rankBy_&i._name;
      %if &createNewVariables=1 %then %do;
         %let newRankVar=&newRankVar rank_&&rankBy_&i._name;
      %end;
   %end;
%mend;
%rankVars;

/*Take all of the input and put together the final code*/
proc rank data=&inTable out=&outTable ties=&ties;
    var &rankVarList;
    %if &createNewVariables=1 %then %do;
        ranks &newRankVar;
    %end;
run;

/*Note that the VALUES section in the JSON allows to you
  specify initial values for the macro variables which will be
  overridden if the user supplies values in the prompt UI*/
```

Step 4: Save the Advanced - Rank Step

The code for your new custom step is complete. To save this custom step, click  . You want to save this step to a folder in SAS Content. Enter **`Rank - Advanced`** as the name of the step.

After you click **Save**, the Rank - Advanced step appears in the Custom Steps folder in the Steps pane.

Edit a Custom Step

You can edit an existing custom step in these ways:

- From the main menu, select **Open** and select the file that you want to open. A custom step file has the extension .step.
- You can also open a custom step from the Steps pane. Under the Custom Steps folder, right-click the name of the step that you want to open and select **Edit**. A flow is required to open a custom step. If a flow is not open, you are prompted to create a flow.

The ***name-of-custom-step*.step** tab opens in Edit mode in the workspace.

Add a Custom Step to a Flow

To add a custom step to a flow:

1. Open the Steps pane.
2. Under the Custom Steps folder, right-click the name of the step that you want to add and select **Add to flow**.

By default, the step is added to the active flow. If no flow exists, you are prompted to create one.

Delete a Custom Step

To delete a custom step:

1. Open the Steps pane.
2. Under the Custom Steps folder, right-click the name of the step that you want to delete and select **Delete**.

Syntax for Custom Steps

Top-Level Object

The UI syntax is an object in JSON and has these top-level properties:

Table 5.1 *Properties of Top-Level Object*

Property Name	Type	Description
syntaxversion	semantic version string	This value represents the semantic version of the UI syntax that is used. This is the minimum version required at run time. An example of a value is `1.0.0`.
pages	array of page definitions	This property defines pages. These pages could contain prompt and layout properties.
values	object mapping a prompt ID to the current value	(Optional) Each prompt has a unique ID and each prompt type has a documented JSON value structure. The values map can be used for persistence and represents current values (which could be initial values).

Common Prompt and Layout Properties

Prompts and layouts are defined as JSON objects and share these properties:

Table 5.2 *Common Prompt and Layout Properties*

Property Name	Type	Description
id	string	IDs are defined in the same namespace and must be unique. An ID string cannot be empty. The first character can be a–z, A–Z or _. Subsequent characters can be 0–9, a–z, A–Z, or _.

Property Name	Type	Description
		The ID is used as the name of the macro variable that is used in code generation templates. There is no limit on the length of the ID. However, if the name of the macro variable is greater than 32 characters, you cannot run the SAS code or the flow.
type	string	This value cannot be blank. The type must be one of the defined type values.
label	string	(Optional) The value of the label property is the name of the prompt in the user interface.
placeholder	string	(Optional) The value of the placeholder property is the text that appears in the prompt when the field is blank.
required	Boolean	(Optional) The default value is `false`. If this value is set to `true`, the prompt is invalid if empty or unspecified. Some prompts could use a property for the minimum number of entries instead of using the required property.

Layout Grouping

Layouts can appear as elements with the top-level visuals array or as children of other layouts. Each layout is defined as a JSON object with these properties:

Table 5.3 Properties for Layouts

Property Name	Type	Description
id	string	See Table 5.2 on page 167.
type	string	See Table 5.2 on page 167.
children	array of objects	This is an array of objects. These objects could be prompt definitions.

Prompt Definition

Prompt definitions could appear as elements with the top-level visuals array or as children of other layouts. Some prompt types support nested prompt elements. Each definition is defined as a JSON object with these properties:

Table 5.4 Properties for Prompt Definitions

Property Name	Type	Description
id	string	See Table 5.2 on page 167.
type	string	See Table 5.2 on page 167.

Layout Types

page

The page layout type creates a tab on the tab bar. Pages can be used only at the top-level of visuals. All top-level elements of visuals must be pages. The page is defined as a JSON object with these properties:

Table 5.5 page Properties

Property Name	Type	Description
type	string = “page”	See Table 5.2 on page 167.
id	string	See Table 5.2 on page 167.

Property Name	Type	Description
label	string	This value is required. See Table 5.2 on page 167.
children	array of objects	See Table 5.2 on page 167.

To view an example of a page control, click  and select **Sample controls**.

The following code defines a page called **Data**:

```
"syntaxversion": "1.0.0",
    "pages": [
        {
            "id": "page1",
            "type": "page",
            "label": "Data",
            "children": [
                {
        ...
```

section

The section layout type creates a collapsible group with an optional label. The section is defined as a JSON object with these properties:

Table 5.6 *section Properties*

Property Name	Type	Description
type	string = “section”	See Table 5.2 on page 167.
id	string	See Table 5.2 on page 167.
label	string	(Optional) See Table 5.2 on page 167.
open	Boolean	(Optional) By default, the value is `false` and the section is collapsed.
children	array of objects	See Table 5.2 on page 167.

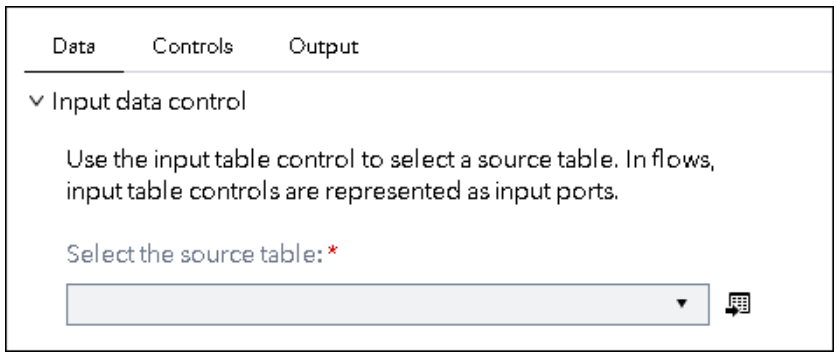
To view an example of a section control, click  and select **Sample controls**.

The **Data** page contains a section called Input data control. Because `"open": true`, this section is open by default in the user interface. As a result, you can see the controls in that section when you click the **Data** tab.

```
"syntaxversion": "1.0.0",
```

```
"pages": [
    {
        "id": "page1",
        "type": "page",
        "label": "Data",
        "children": [
            {
          "id": "section_data",
                    "type": "section",
                    "label": "Input data control",
                    "open": true,
                    "children": [


...
```



Syntax for Prompt Types

checkbox

Check boxes can be used to turn an option on or off. The checkbox control is a JSON object with these properties:

Table 5.7 *Properties for checkbox Control*

Property Name	Type	Description
type	string = “checkbox”	See Table 5.2 on page 167.
id	string	See Table 5.2 on page 167.
label	string	(Optional) See Table 5.2 on page 167.

The value of this control is a JSON Boolean. A blank value is considered false.

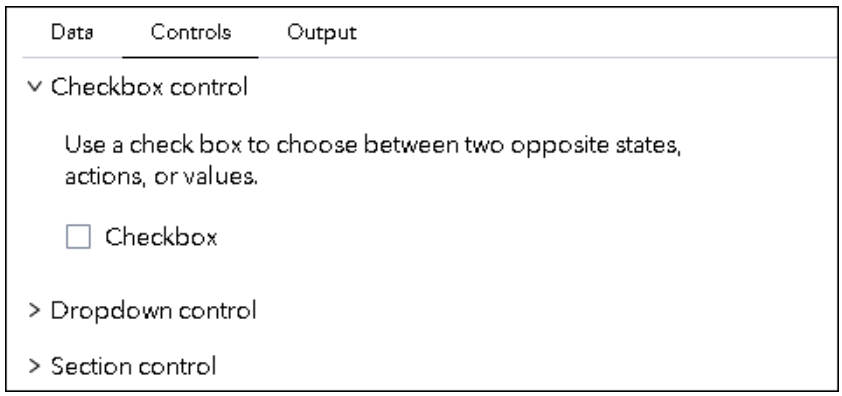
To view an example of a checkbox control, click  and select **Sample controls**.

```
...
{
        1 "id": "section_checkbox_section",
          "type": "section",
          "label": "Checkbox control",
          "open": false,
          "children": [
             {
                 2 "id": "checkbox_control",
                   "type": "checkbox",
                   "label": "Checkbox"
              }
        ]
    },
...
```

1 This code creates a section called **Checkbox control**. Because `"open": false`, the section is collapsed by default, which means that the checkbox control is not visible by default.

2 This code creates a checkbox control called **Checkbox**.

To view the checkbox control, click **Preview**, and then click the **Controls** tab.



On the **Template** tab, a macro variable is generated based on the ID of the checkbox control. In this example, the macro variable is `%put &=checkbox_control;`.

columnselector

The columnselector control enables the user to select a list of columns. The available columns are referenced in an inputtable control. This JSON object has these properties:

Table 5.8 Properties for columnselector Control

Property Name	Type	Description
type	string = “columnselector”	See Table 5.2 on page 167.
id	string	See Table 5.2 on page 167.
table	string	This value is the ID from the inputtable control that refers to the available columns.
columntype	string with value of either `a`, `c`, or `n`	(Optional) ■ If the value is `a`, all column types are available. ■ If the value is `c`, only character columns are available. ■ If the value is `n`, only numeric columns are available.
label	string	(Optional) See Table 5.2 on page 167.
max	integer	(Optional) The default value is `0`. Use this property to specify the maximum number of columns that can be selected. ■ A value of `0` indicates that there is no limit to the number of columns that can be selected. ■ A value greater than 0 means that the prompt is invalid if the number of selected columns exceeds the maximum value.
min	integer	(Optional) The default value is `0`. Use this property to specify the minimum number of columns that must be selected. ■ A value of `0` indicates that there is no limit to the number of columns that can be selected. It also indicates that adding columns is optional. ■ A value greater than 0 means that the prompt is invalid if the number of selected columns is less than the minimum number of values.
order	Boolean	(Optional) The default value is `false`. If this value is `true`, the user can change the order of selected columns and that order is saved.

The value of this control is a JSON array of objects. The array contains the name of each selected column.

Table 5.9 value Property of columnselector Control

Property	Type	Description
value	string	Name of the selected column

A blank value is an empty array.

To view an example of a columnselector control, click  and select **Sample controls**.

```
...
{
        1 "id": "section_columnselector",
          "type": "section",
          "label": "Column selector controls",
          "open": true,
          "children": [
             {
                  2 "id": "colSelector_single",
                    "type": "columnselector",
                    "label": Select a single column: ",
                    3 "table": "input_table"
           "order": false,
           4 "max": 1,
           "min": 0
            },
              {
                  5 "id": "colSelector_multi",
                    "type": "columnselector",
                    "label": "Select multiple character columns:",
                    "table": "input_table",
                    "order": true,
                    "columntype": "c",
                    "max": 0,
                    "min": 0
             }

...
```

1 This code creates a section called **Column selector controls**. Because `"open": true`, the section is expanded by default, which means that the columnselector control is visible by default.

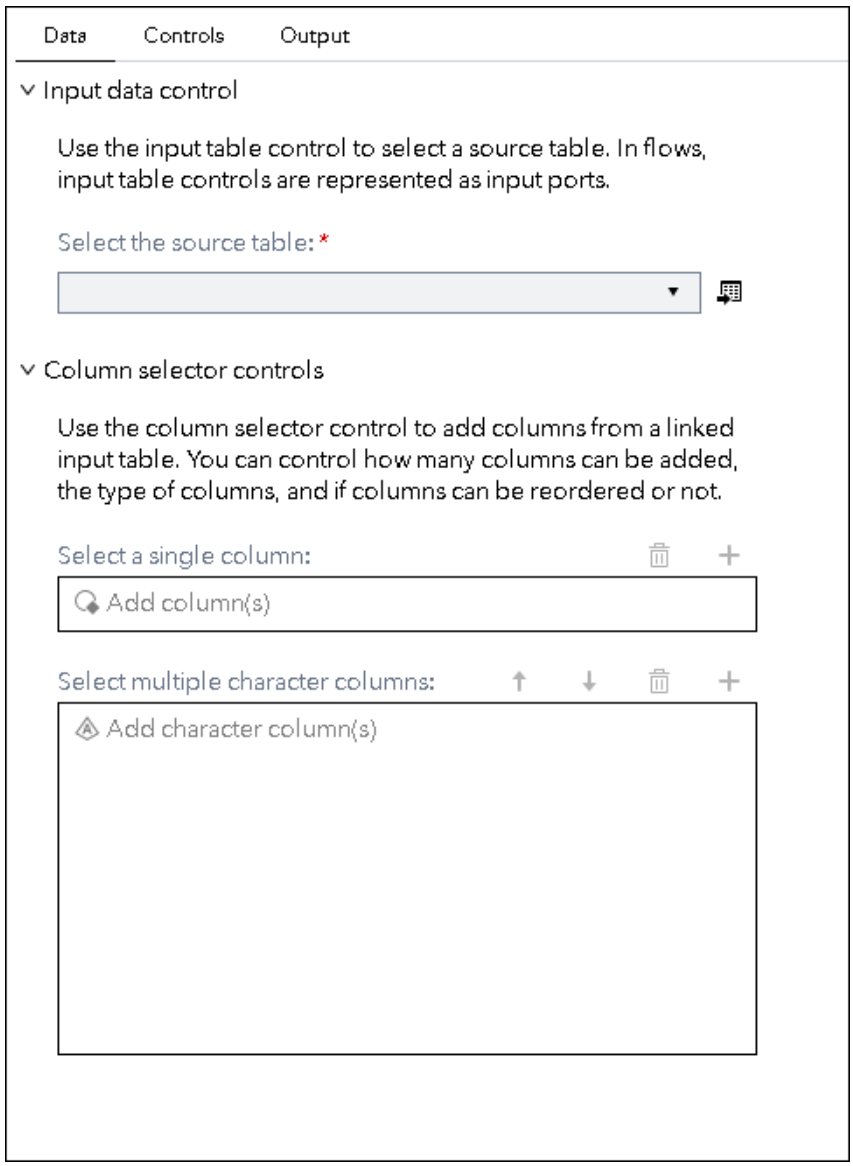
2 This code creates a columnselector control labeled **Select a single column**.

3 The table property specifies the unique ID of the inputtable control that contains the columns that you want available from the columnselector control.

4 The max property specifies that a maximum of one column can be selected. The min property is set to 0, which means that no selection is required to run the step.

5 This code creates a columnselector control labeled **Select multiple character columns**. The max property is set to 0, which means the user can select an unlimited number of columns. The min property is set to 0, which means that no column is required to run the step.

To view the columnselector control, click **Preview**, and then click the **Data** tab.



On the **Template** tab, macro variables are generated based on the ID of the columnselector control.

```
/* How many columns were selected in the column selector */
%put &=colSelector_single_count;
/* Column selectors support name and type so check */
/* the count and only output the defined name and */
/* type if we have at least one column selected. */
%if &colSelector_single_count > 0 %then %do;
    %put &=colSelector_single_1_name;
    %put &=colSelector_single_1_type;
%end;
```

dropdown

The dropdown control enables users to select a value from a menu. This control is defined as a JSON object with these properties:

Table 5.10 Properties for dropdown Control

Property Name	Type	Description
type	string = “dropdown”	See Table 5.2 on page 167.
id	string	See Table 5.2 on page 167.
items	array of object	See Table 5.11 on page 176.
label	string	(Optional) See Table 5.2 on page 167.
placeholder	string	(Optional) See Table 5.2 on page 167.
required	Boolean	(Optional) The default value is `false`. See Table 5.2 on page 167.

The items property is a JSON array object with these properties:

Table 5.11 Properties for items property

Property Name	Type	Description
label	string	(Optional) This value is displayed on the screen.
value	string | number	Use this value when defining the code generation variables.

The value of this control is an object, the chosen item.

To view an example of a dropdown control, click  and select **Sample controls**.

```
...
{
        "id": "section_dropdown",
         "type": "section",
         "label": "Dropdown control",
         "open": false,
         "children": [
            {
                 "id": "dropdown_control",
                  "type": "dropdown",
                  "label": "Dropdown:",
```

```
                3 "items": [
                    {
                        "value": "Value 1"
                    },
                    {
                       "value": "Value 2"
                    }
                ],
                4 "required": false,
            "placeholder": ""
        }
         ]
    },
...
```

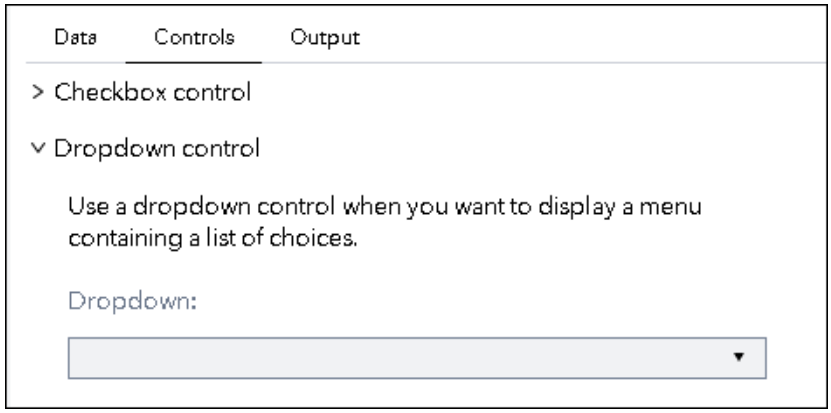
1 This code creates a section called **Dropdown control**. Because `"open": false`, the section is collapsed by default, which means that the dropdown control is not visible by default.

2 This code creates a dropdown control called **Dropdown**.

3 The items property defines the values in the drop-down list. In this example, there are two possible values: **`Value 1`** and **`Value 2`**.

4 Because `"required": false`, you do not have to select a value for the **Dropdown control** in order for the step to run successfully.

To view the dropdown control, click **Preview**, and then click the **Controls** tab.



On the **Template** tab, a macro variable is generated for the dropdown control. In this example, the macro variable is `%put &dropdown_control;`.

inputtable

The inputtable control specifies the table to use as input. This JSON object has these properties:

Table 5.12 Properties for inputtable Control

Property Name	Type	Description
type	string = “inputtable”	See Table 5.2 on page 167.
id	string	See Table 5.2 on page 167.
label	string	(Optional) See Table 5.2 on page 167.
placeholder	string	(Optional) See Table 5.2 on page 167.
required	Boolean	(Optional) The default value is `false` and an input table is not required. To require the user to specify an input table, set this property to `true`.

The value of this control is a JSON object that lists the library name and table name for the input table.

To view an example of an input table control, click  and select **Sample controls**.

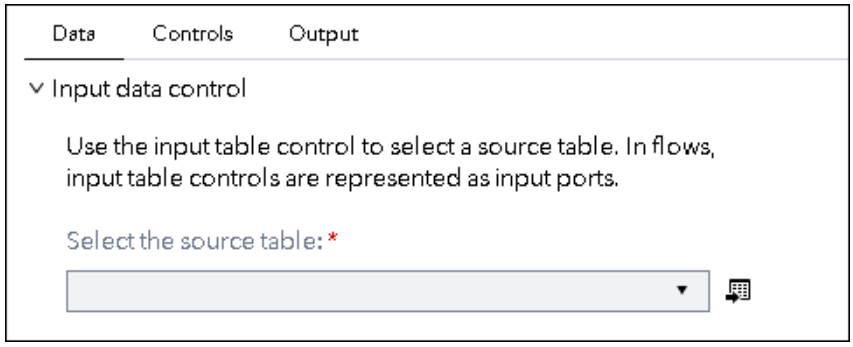
```
...
{
       1"id": "section_data",
        "type": "section",
        "label": "Input data control",
        "open": true,
        "children": [
           {
              2"id": "input_table",
               "type": "inputtable",
               "label": "Select the source table:",
               3"required": true
            }
...
```

1 This code creates a section called **Input data control**. Because `"open": true`, the section is expanded by default.

2 This code creates an inputtable control called **Select the source table**.

3 Because `"required": true`, you must specify an input table in order for the step to run successfully. In the user interface, a required field is indicated by a red asterisk.

To view the inputtable control for this sample, click **Preview**, and then click the **Data** tab.



IMPORTANT When the custom step is in a flow, you do not see the table selector for the inputtable control. Instead, you see an input port for the node. For more information, see "Specify Port Details for Input and Output Tables in a Flow" on page 151.

On the **Template** tab, a macro variable is generated using the ID of the inputtable control. In this example, here is the code for the inputtable control:

```
/* Data page */
/* Table connected to the input port */
%put &=input_table;
```

numberfield

The numberfield control enables the user to enter a number. This JSON object has these properties:

Table 5.13 *Properties for numberfield Control*

Property Name	Type	Description
type	string = "numberfield"	See Table 5.2 on page 167.
id	string	See Table 5.2 on page 167.
excludemax	Boolean	(Optional) The default value is `false`. If the value is true, the specified value must be less than the maximum value.
excludemin	Boolean	(Optional) The default value is `false`. If the value is true, the specified value must be

Property Name	Type	Description
		greater than the minimum value.
label	string	(Optional) See Table 5.2 on page 167.
max	number	(Optional) For the value in the numberfield control to be valid it must be equal to or less than this number. If `excludemax=true`, the value must be less than this number.
min	number	(Optional) For the value in the numberfield control to be valid it must be equal to or greater than this number. If `excludemin=true`, the value must be greater than this number.
placeholder	string	(Optional) The default value is **`false`**. See Table 5.2 on page 167.
required	Boolean	(Optional) The default value is **`false`**. See Table 5.2 on page 167.

The value of this control is a JSON number. The blank value is null.

To view an example of an input table control, click  and select **Sample controls**.

```
...
{
        1 "id": "section_textfield",
          "type": "section",
          "label": "Text field controls",
          "open": false,
          "children": [

                              ...

            {
                  2 "id": "text_numericfield",
                    "type": "numberfield",
                    "label": "Numeric field:",
```

```
                3 "placeholder": "Enter a number between 10 and 20,
inclusive",
                4 "required": false,
                5 "max": 100,
                  "min": 0,
          "excludemin": false,
          "excludemax:" false
             }

...
```

1 This code creates a section called **Text field controls**. Because `"open": false`, the section is collapsed by default.

2 This code creates a numberfield control called **Numeric field**.

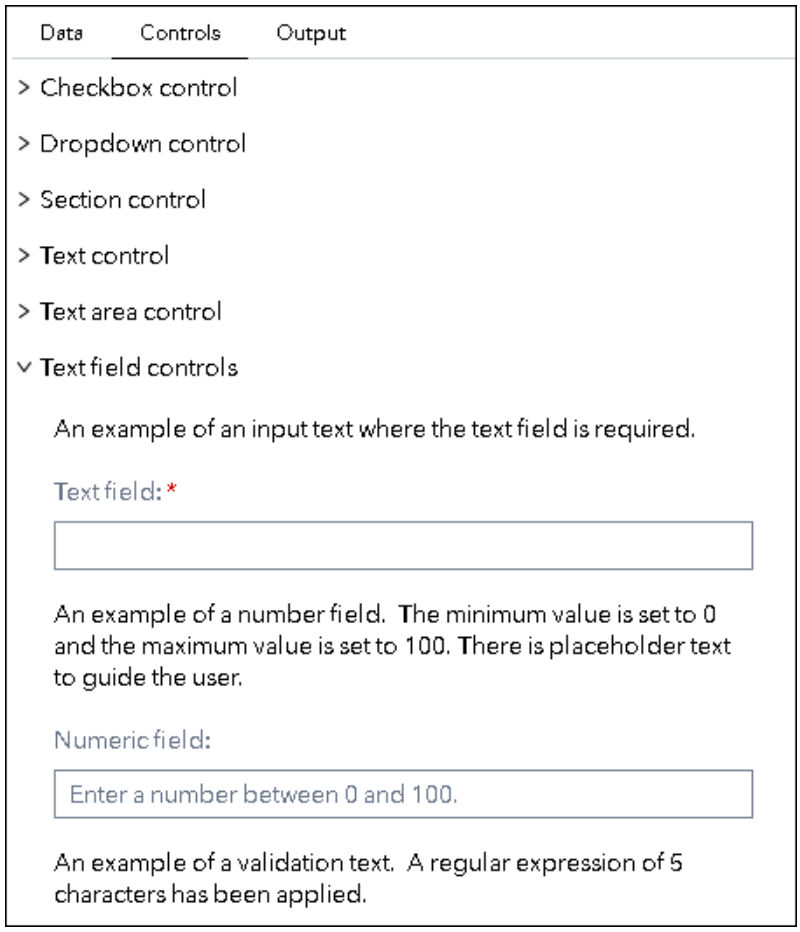
3 Use the placeholder property to specify any instructional text that should appear in the field when no value is specified.

4 Because `"required": false`, you do not need to specify a value for this control in order for the step to run successfully.

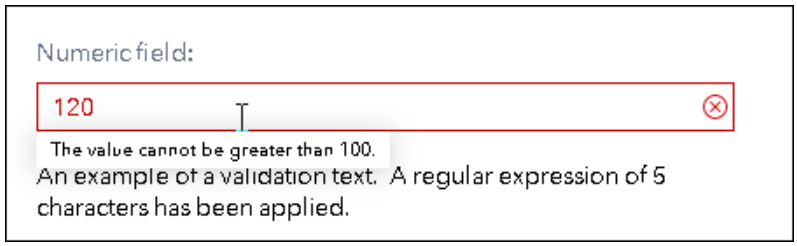
5 Use the min and max parameters to specify the minimum and maximum values for this field. In this example, valid values are between 100 and 0 (inclusive). If the user specifies a value outside this range, an error is generated.

To view the numberfield control, click **Preview**, and then click the **Controls** tab. Expand the **Text input controls** section.

Because you specified the placeholder property, the placeholder text appears in the Numeric field.



If you enter a value that exceeds the maximum value (100), you receive an error message.



On the **Template** tab, a macro variable is generated using the ID for the numberfield control. In this example, the macro variable for the numberfield control is `%put &numberfield_control;`.

outputtable

The outputtable control specifies the table to use as output. The output table is required. This JSON object has these properties:

Table 5.14 Properties for outputtable Control

Property Name	Type	Description
type	string = “outputtable”	See Table 5.2 on page 167.
id	string	See Table 5.2 on page 167.
label	string	(Optional) See Table 5.2 on page 167.
required	Boolean	(Optional) The default value is `false` and you do not have to specify an output table. To require the user to specify an output table, set this property to `true`.

The value of this control is a JSON object that lists the library name and table name of the output table.

To view an example of an outputtable control, click  and select **Sample controls**.

```
...
{
            1 "id": "page3",
            "type": "page",
            "label": "Output",
            "children": [
                {
                    2 "id": "output_section",
                    "type": "section",
                    "label": "Output data control",
                    "open": true,
                    "children": [
                        {
                            3 "id": "output_table",
                            "type": "outputtable",
                            "label": "Output data control:",
                            "required": true
                        }

...
```

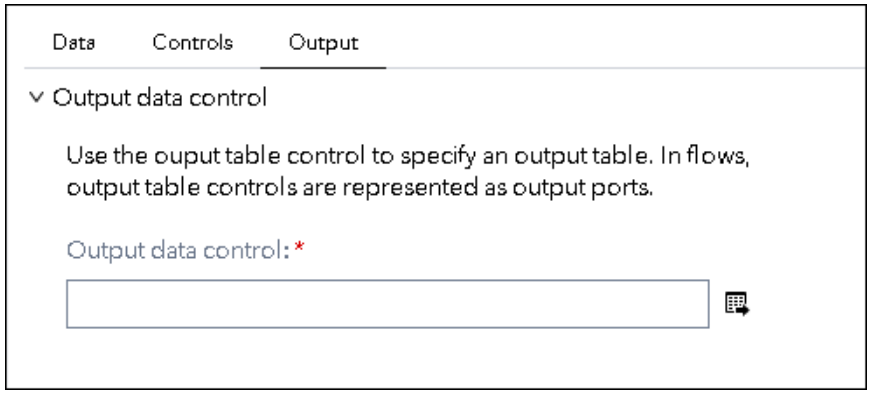
1 This code creates a page called **Output**.

2 This code creates a section called **Output**. Because `"open": true`, the section is expanded by default.

3 This code creates an outputtable control called **Output data control**.

4 Because `"required": true`, you must specify a name for the output table in order for the step to run successfully. In the user interface, a required field is indicated by a red asterisk.

To view the outputtable control, click **Preview**, and then click the **Output** tab.



IMPORTANT When the custom step is in a flow, you do not see the table selector for the outputtable control. Instead, you see an output port for the node. For more information, see "Specify Port Details for Input and Output Tables in a Flow" on page 151.

On the **Template** tab, you can specify a macro variable that contains the value for each control. In this example, here is the code for the output_table control:

```
/* Output page */
/* Table connected to the output port */
%put &=output_table;
```

text

The text control provides a single line of text that is usually used to provide a simple instruction or some information. This JSON object has these properties:

Table 5.15 Properties for text Control

Property Name	Type	Description
type	string = "text"	See Table 5.2 on page 167.
id	string	See Table 5.2 on page 167.
text	string	The string specifies the text to be displayed.

A text prompt is for display purposes only and has no value.

To view an example of a text control, click  and select **Sample controls**.

```
...
{
            "id": "page3",
            "type": "page",
```

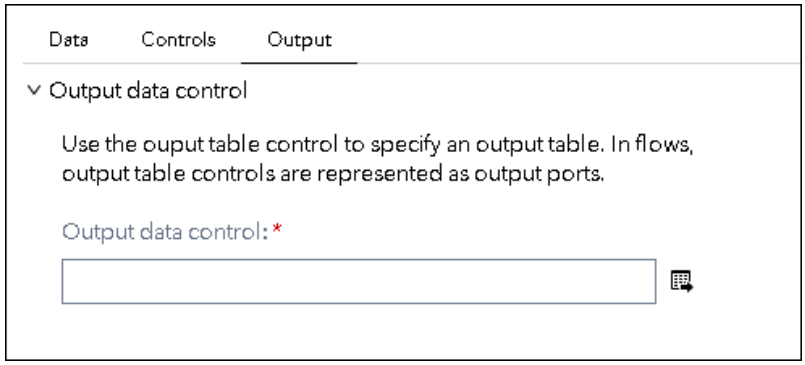
```
            "label": "Output",
            "children": [
                {
                    "id": "output_section",
                    "type": "section",
                    "label": "Output data control",
                    "open": true,
                    "children": [
                        {
                            2 "id": "text_outputtable",
                            "type": "text",
                            "text": "Use the ouput table control to
specify an output table. In flows, output table controls are
represented as output ports."
                        },
                        {
                            "id": "output_table",
                            "type": "outputtable",
                            "label": "Output data control:",
                            "required": true
                        }

...
```

1 This code creates a page called **Output**.

2 This code creates a text control that displays the following text: Use the output table control to specify an output table. In flows, output table controls are represented as output ports.

To view the text control, click **Preview**, and then click the **Output** tab.



Because the text control is for display purposes only, you do not add any content for this control to the **Template** tab.

textarea

The textarea control provides an area for multi-line text input. This JSON object has these properties:

Table 5.16 Properties for textarea Control

Property Name	Type	Description
type	string = “textarea”	See Table 5.2 on page 167.
id	string	See Table 5.2 on page 167.
required	Boolean	The default value is `false` and no text input is required.

The value of this control, which is a string, is a JSON object. A blank value is an empty string.

To view an example of a textarea control, click  and select **Sample controls**.

```
...
{
                    1 "id": "section_textarea",
                    "type": "section",
                    "label": "Text area control",
                    "open": false,
                    "children": [


...
                        {
                            2 "id": "textarea_control",
                            "type": "textarea",
                            "label": "Text area:",
                            3 "placeholder": "",
                            4 "required": false
                        }
...
```

1 This code creates a section called **Text area controls**. Because `"open": false`, the section is collapsed by default.

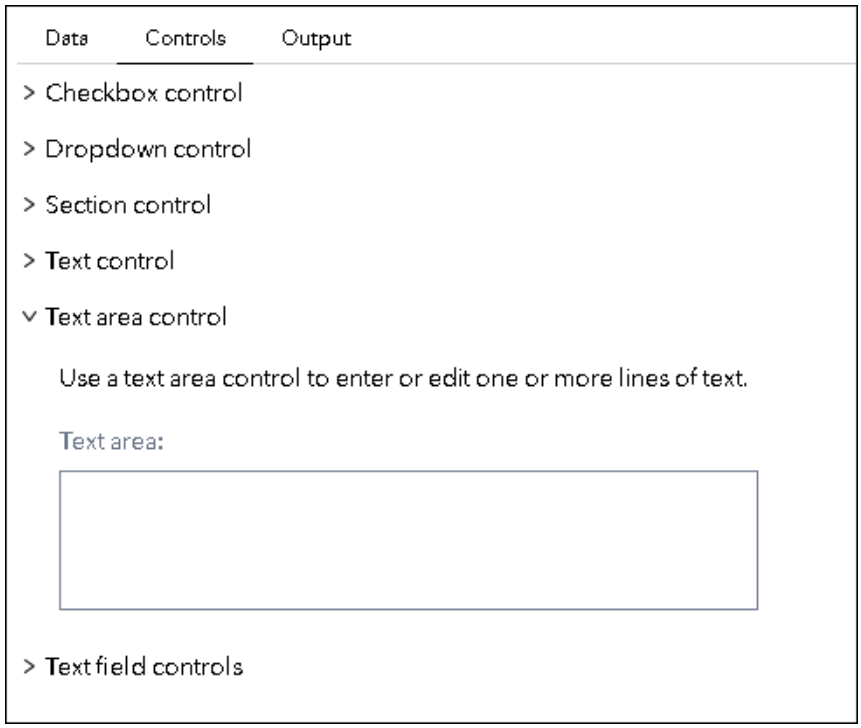
2 This code creates a textarea control called **Text area**.

3 Use the placeholder property to specify instructional text to be displayed in the textarea control when the field is empty. In this example, no placeholder text is displayed.

4 Because `"required": false`, you do not need to specify a value for this control.

5 In the values section of the JSON code, you specify default values for your controls. To map the default value to the correct control, use the unique ID that you specified for the control. In this example, the default value for the text_area control is null and no default value is displayed. (Instead, the placeholder text is displayed.)

To view the textarea control, click **Preview**, and then click the **Controls** tab. Expand the **Text Input controls** section.



On the **Template** tab, a macro variable is generated based on the ID of the textarea control. In this example, the macro variable for the text_area control is `%put &=textarea_control;`.

textfield

The textfield control is used for single-line input of a text value. This JSON object has these properties:

Table 5.17 *Properties for textfield Control*

Property Name	Type	Description
type	string = “textfield”	See Table 5.2 on page 167.
id	string	See Table 5.2 on page 167.
label	string	(Optional) See Table 5.2 on page 167.
placeholder	string	(Optional) See Table 5.2 on page 167.
regexp	string	(Optional) This property is the regular expression that the value must match in order to be considered valid.
required	Boolean	(Optional) The default value is `false`. See Table 5.2 on page 167.

The value of this control, which is a string, is a JSON object. A blank value is an empty string.

To view an example of a textfield control, click  and select **Sample controls**.

```
...
{
                    1 "id": "section_textfield",
                    "type": "section",
                    "label": "Text field controls",
                    "open": false,
                    "children": [


...
                        {
                            2 "id": "textfield_control",
                            "type": "textfield",
                            "label": "Text field",
                            4 "placeholder": "",
                            5 "required": true
                        },
...
```

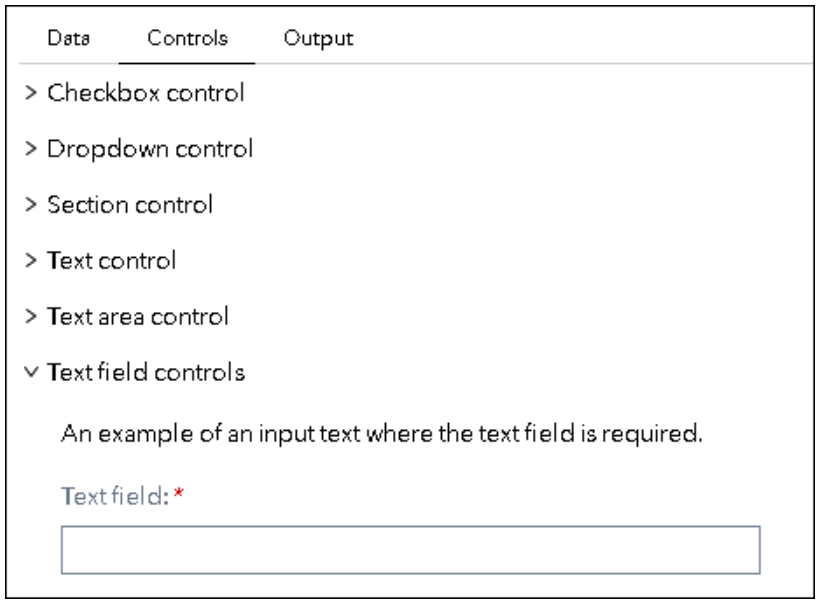
1 This code creates a section called **Text field controls**. Because `"open": false`, the section is collapsed by default.

2 This code creates a textfield control called **Text field**.

3 Use the placeholder property to specify instructional text to be displayed in the textfield control when the field is empty. In this example, no placeholder text is specified.

4 Because `"required": true`, you must specify a value for this control.

To view the textfield control, click **Preview**, and then click the **Controls** tab. Expand the **Text Input controls** section.



On the **Template** tab, a macro variable is generated based on the ID of the textfield control. In this example, the macro variable for the textfield control is `%put &=textfield_control;`.

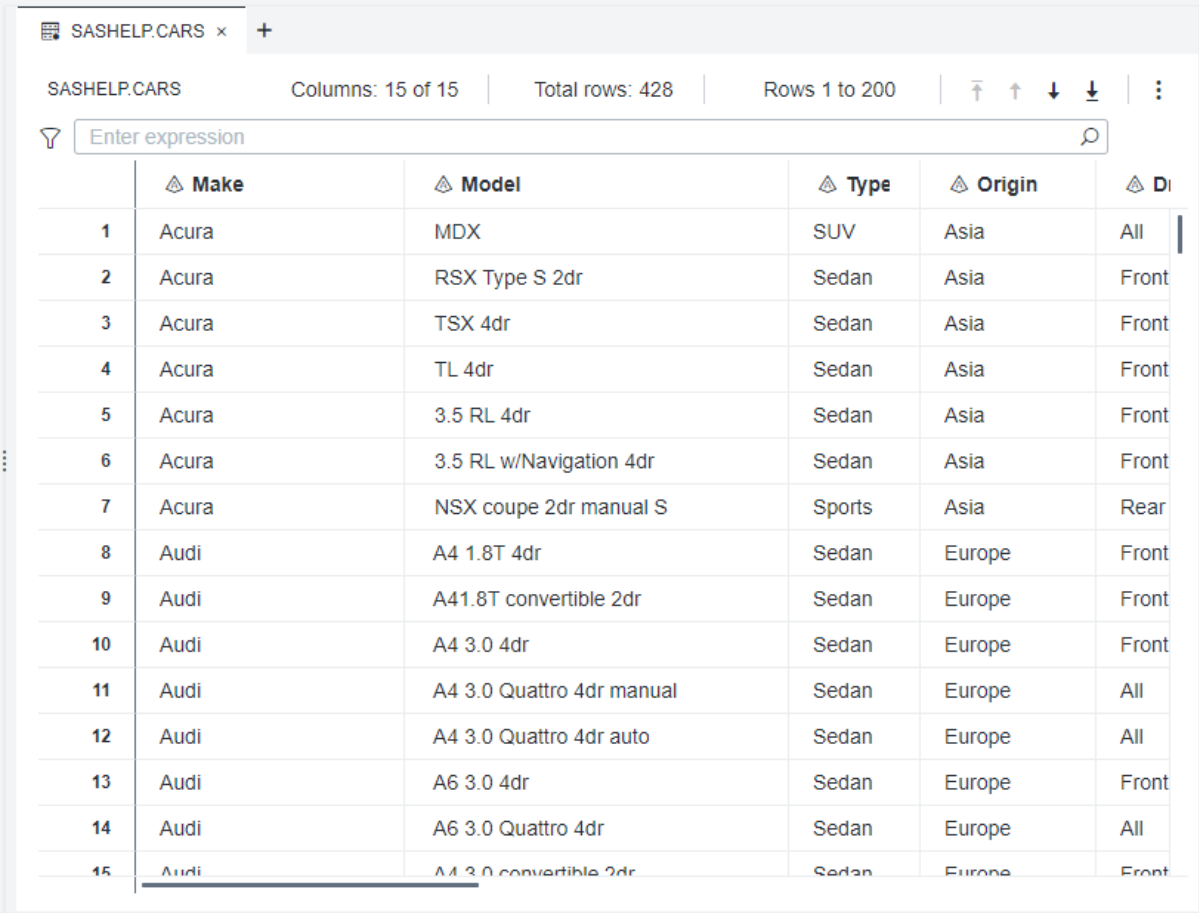
6

Working with Data

About the Table Viewer

The table viewer is used to display your data in SAS Studio. You can click ↓ and ↑ to navigate through the data.

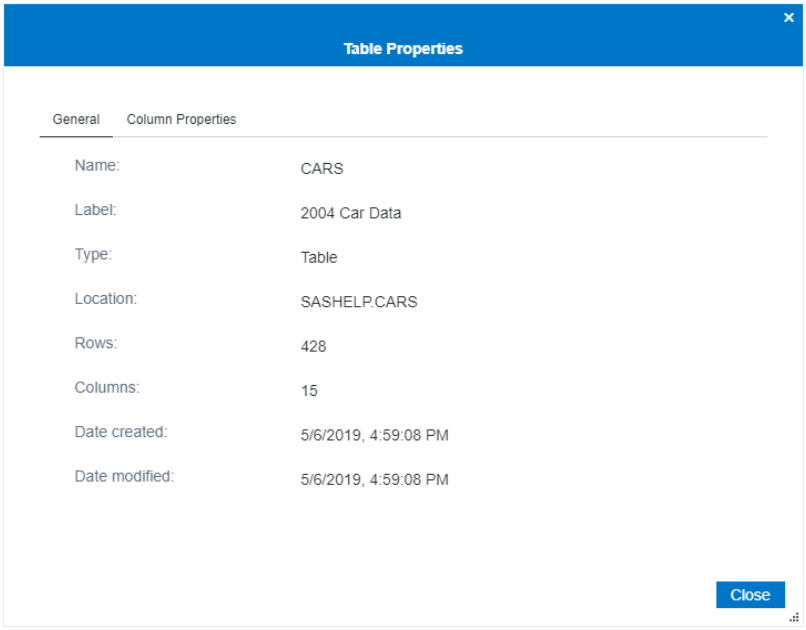
Note: You can click ⤓ and ⤒ to advance to the last or first page in your data. You cannot navigate directly to the last page of a CAS table, any table with an unknown number of columns, or a data view.

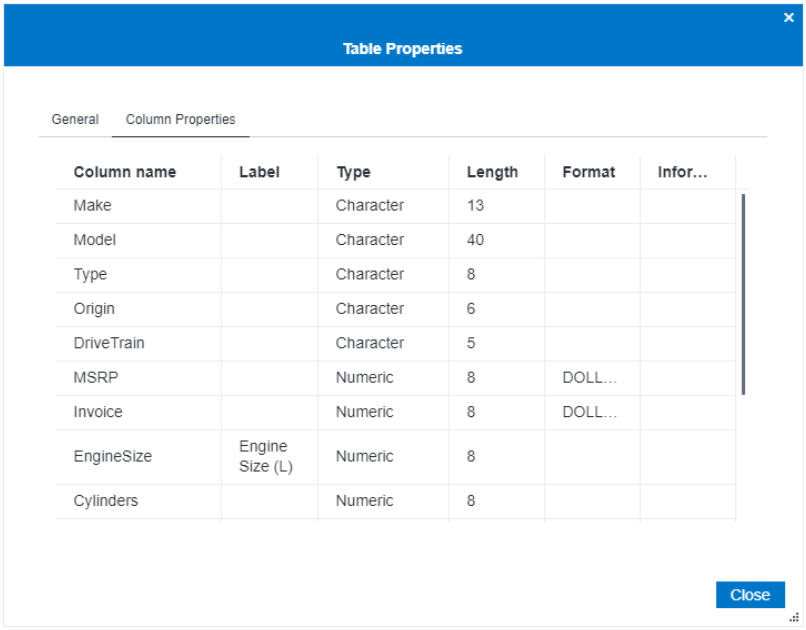


	Make	Model	Type	Origin	Di
1	Acura	MDX	SUV	Asia	All
2	Acura	RSX Type S 2dr	Sedan	Asia	Front
3	Acura	TSX 4dr	Sedan	Asia	Front
4	Acura	TL 4dr	Sedan	Asia	Front
5	Acura	3.5 RL 4dr	Sedan	Asia	Front
6	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front
7	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear
8	Audi	A4 1.8T 4dr	Sedan	Europe	Front
9	Audi	A41.8T convertible 2dr	Sedan	Europe	Front
10	Audi	A4 3.0 4dr	Sedan	Europe	Front
11	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All
12	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All
13	Audi	A6 3.0 4dr	Sedan	Europe	Front
14	Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All
15	Audi	A4 3.0 convertible 2dr	Sedan	Europe	Front

Note: By default, the table viewer displays the first 200 rows and 100 columns of the table. If the structure or data values of the table change while the table is open, you must refresh the table viewer to see the changes. If the structure of the table changes and you do not refresh the table, the columns that are listed in the **Libraries** section of the navigation pane might be different from the columns that are displayed in the table viewer. For more information, see "Refreshing Your Data" on page 199.

You can view the properties of the table and its columns by clicking on the toolbar of the **Libraries** section in the navigation pane.





Opening and Viewing Data

You can open files in SAS Studio in several ways:

- You can double-click a file in the Explorer and Libraries sections of the navigation pane.
- You can drag a file from the Explorer and Libraries sections of the navigation pane to the work area.
- You can search for a file and open it by double-clicking it in the search results.

- You can open a file by using a file reference in the File References section of the navigation pane. You can open the file by double-clicking it or by dragging it to the work area.

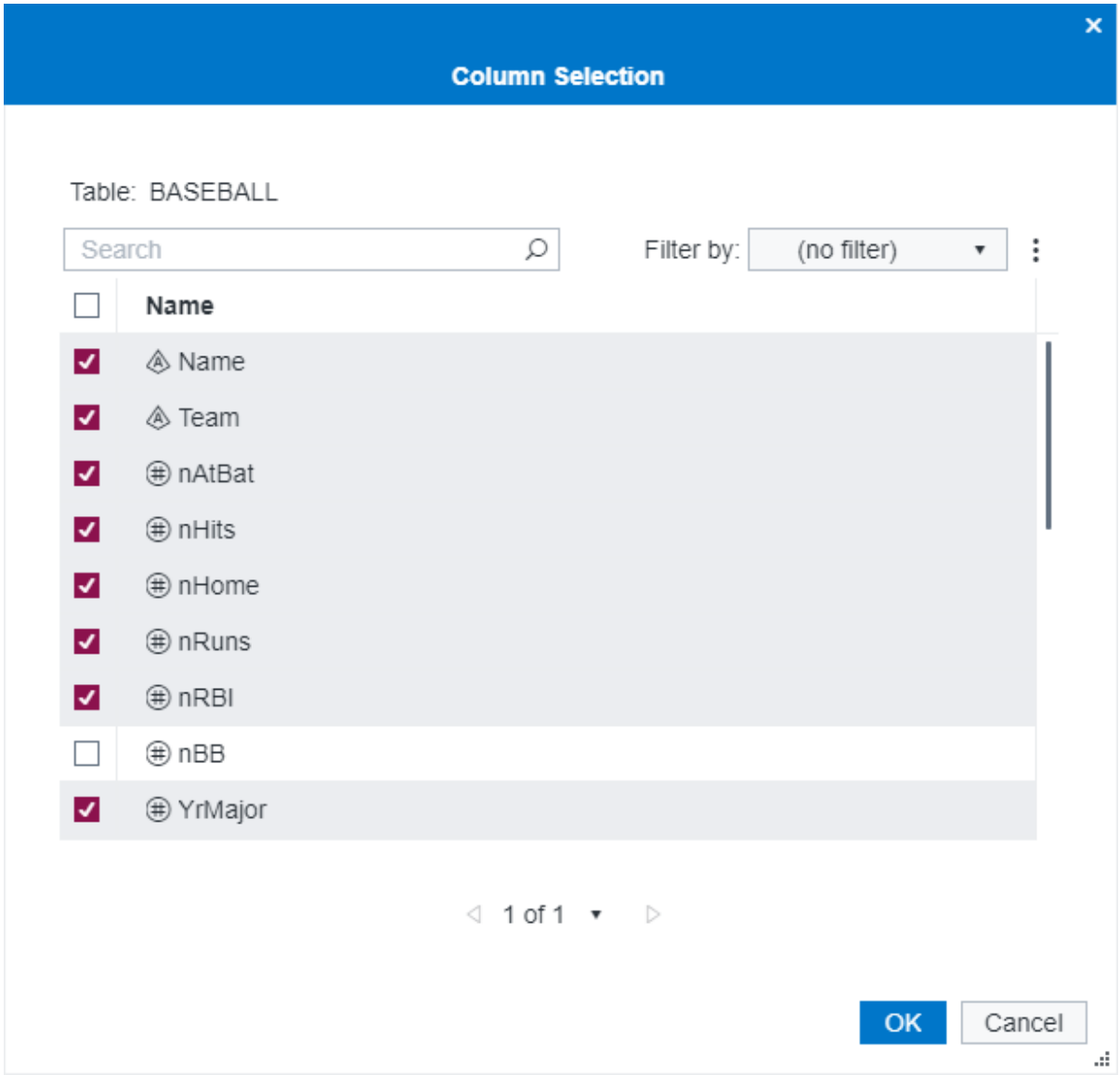
Note: You cannot access password-protected data or data in a SAS Content folder that includes "&" in the folder name.

Note: If you open a table that was created with an encoding that has a different byte size than the encoding of your SAS compute server, you must use the CVP (character variable padding) engine in your LIBNAME statement. For more information, see Avoiding Character Data Truncation by using the CVP Engine in *SAS National Language Support (NLS): Reference Guide*.

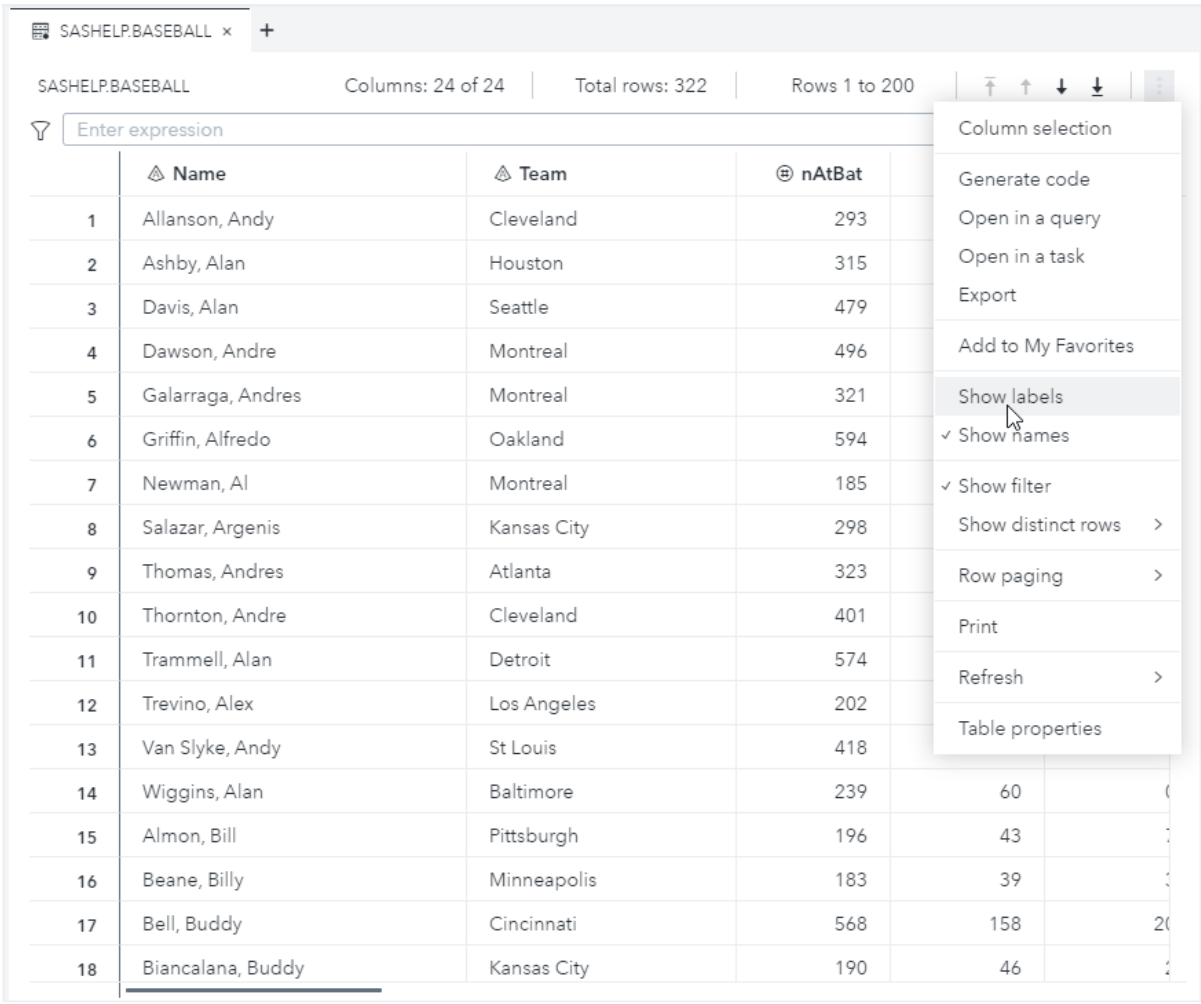
When you open a table, the first 100 columns in the table are displayed by default. The number of columns that can be displayed is determined by the **Columns displayed** option on the Tables page of the Preferences window. For more information, see "Setting Tables Preferences" on page 269.

To specify which columns you want to include in the table viewer, click ⋮ on the toolbar and select **Column selection**. You can use the Column Selection window to search for columns and to specify whether to show or hide specific columns in the data. You can also right-click a column in the table viewer and select **Hide** to hide a column.

Note: You can click ⋮ in the Column Selection window to specify whether to display the column name, label, or both, how many columns to display per page, and how to sort the columns. These settings affect only the Column Selection window.



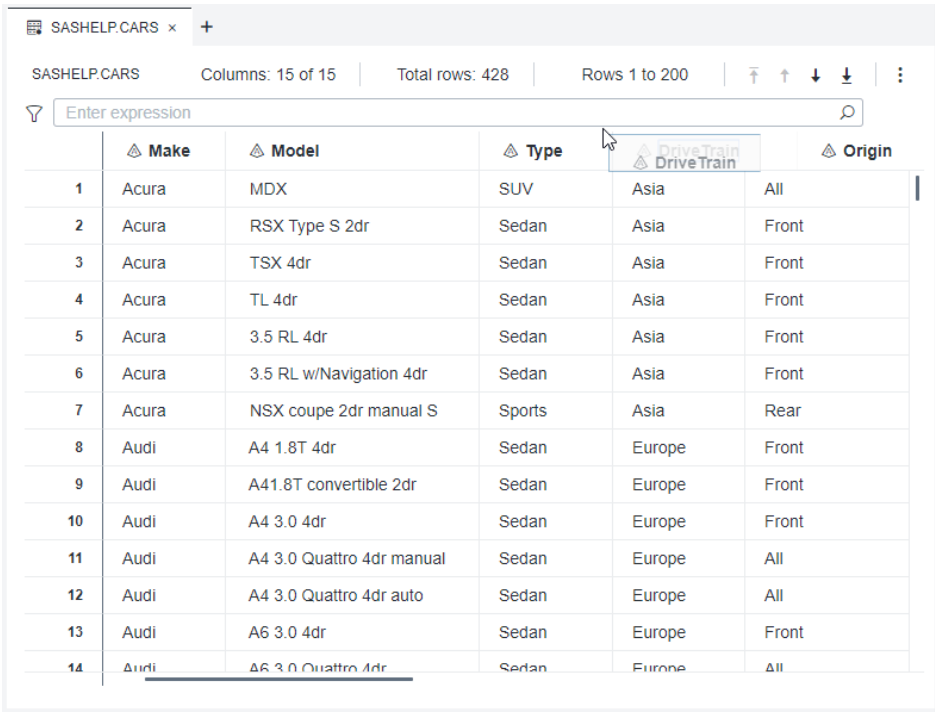
By default, the column names are displayed, but you can choose to display the column labels by clicking ⋮ on the toolbar and selecting **Show labels**.



	Name	Team	nAtBat		
1	Allanson, Andy	Cleveland	293		
2	Ashby, Alan	Houston	315		
3	Davis, Alan	Seattle	479		
4	Dawson, Andre	Montreal	496		
5	Galarraga, Andres	Montreal	321		
6	Griffin, Alfredo	Oakland	594		
7	Newman, Al	Montreal	185		
8	Salazar, Argenis	Kansas City	298		
9	Thomas, Andres	Atlanta	323		
10	Thornton, Andre	Cleveland	401		
11	Trammell, Alan	Detroit	574		
12	Trevino, Alex	Los Angeles	202		
13	Van Slyke, Andy	St Louis	418		
14	Wiggins, Alan	Baltimore	239	60	
15	Almon, Bill	Pittsburgh	196	43	
16	Beane, Billy	Minneapolis	183	39	
17	Bell, Buddy	Cincinnati	568	158	
18	Biancalana, Buddy	Kansas City	190	46	

Note: By default, the table viewer displays the total number of rows in the table and the total number of filtered rows, if you have filtered the data. However, if SAS Studio is unable to determine the row counts without affecting performance, then the row counts are not displayed. Row counts are not available for relational tables, views, and Hadoop data.

You can change the order of the columns in the table viewer by dragging a column to a new position in the table viewer or in the Columns area. You can also sort and freeze the data by one or more columns. For more information, see “Sorting and Freezing Data” on page 199.



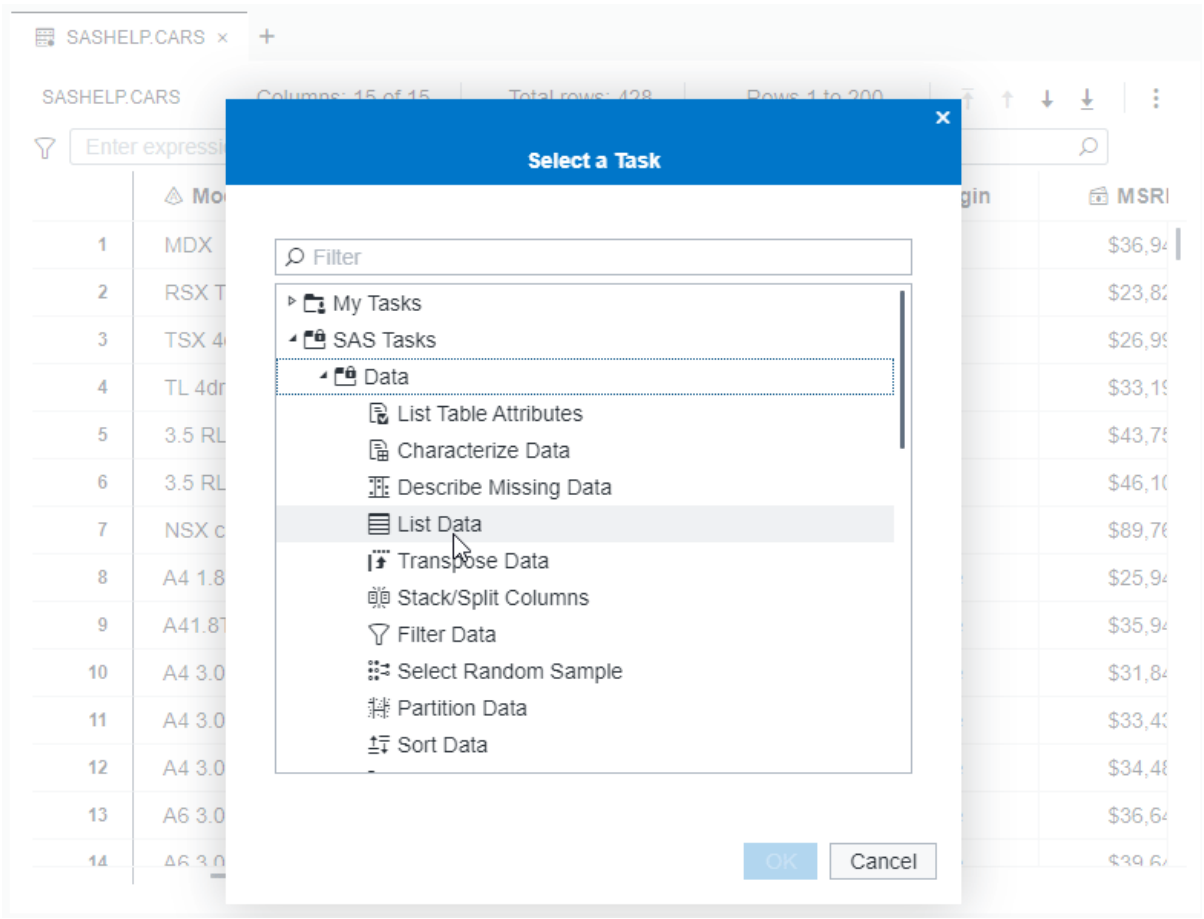
	Make	Model	Type	DriveTrain	Origin
1	Acura	MDX	SUV	Asia	All
2	Acura	RSX Type S 2dr	Sedan	Asia	Front
3	Acura	TSX 4dr	Sedan	Asia	Front
4	Acura	TL 4dr	Sedan	Asia	Front
5	Acura	3.5 RL 4dr	Sedan	Asia	Front
6	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front
7	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear
8	Audi	A4 1.8T 4dr	Sedan	Europe	Front
9	Audi	A41.8T convertible 2dr	Sedan	Europe	Front
10	Audi	A4 3.0 4dr	Sedan	Europe	Front
11	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All
12	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All
13	Audi	A6 3.0 4dr	Sedan	Europe	Front
14	Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All

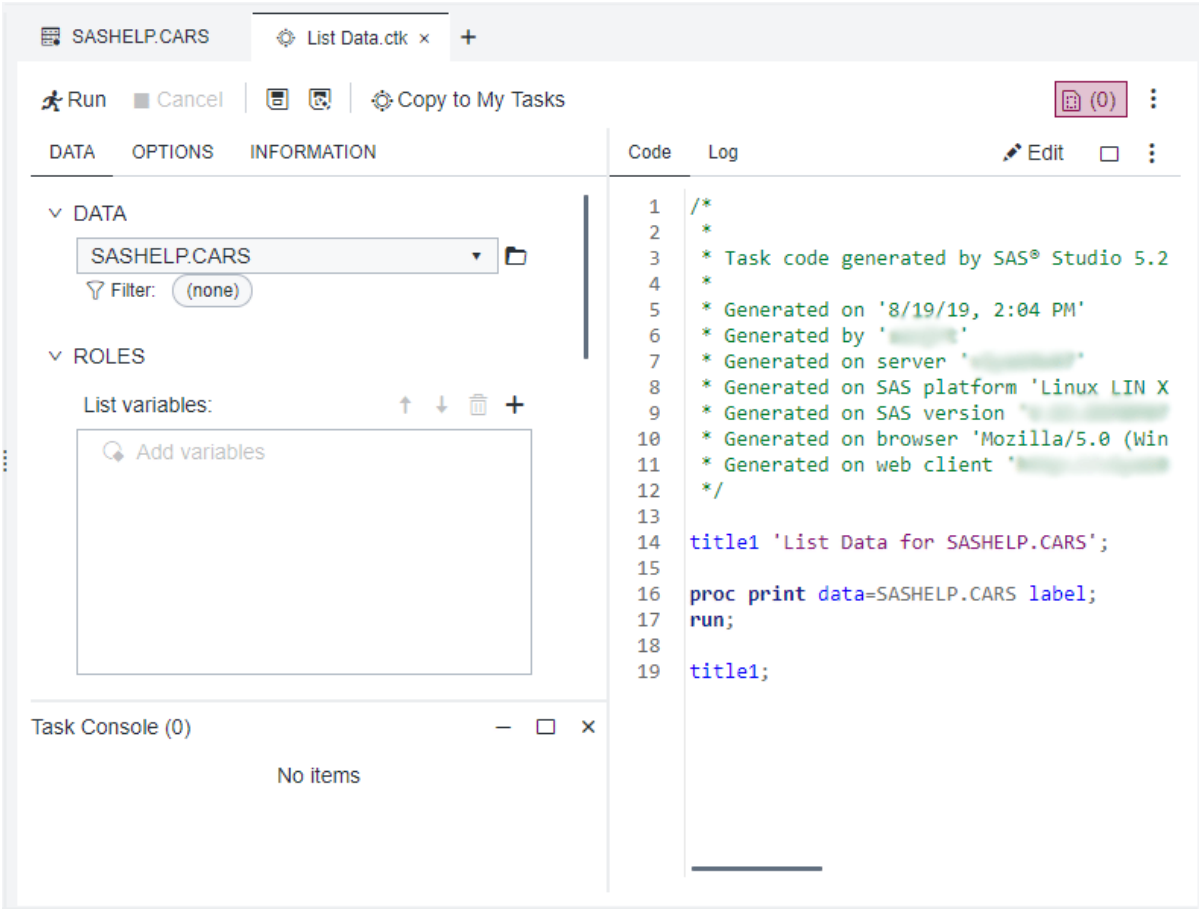
Viewing the Code That Is Used to Create a Table

While you select options and customize the table to look the way you want it to, SAS Studio is generating SAS code that you can use. To view the code, click ⋮ on the toolbar and select **Generate code**. A new program window appears with the code that was used to create the view of the table in the table viewer. The program is a copy of the code and is no longer associated with the original code. Editing the code does not affect the data that is displayed in the table viewer, and modifying the table viewer does not affect the contents of the code.

Opening Data in a Task or Query from the Table Viewer

You can use the table viewer to open a task or query with the data that you are currently viewing. The task or query opens with the data automatically selected as the input source. To open a task or query with the current data selected, click ⋮ on the toolbar and select **Open in a task** or **Open in a query**.


SASHELP.CARS
Select a Task
Filter
My Tasks
SAS Tasks
Data
List Table Attributes
Characterize Data
Describe Missing Data
List Data
Transpose Data
Stack/Split Columns
Filter Data
Select Random Sample
Partition Data
Sort Data
OK
Cancel



Refreshing Your Data

You can refresh the table viewer to update the data that is displayed in the table. When you refresh your data, you can choose to either reset any settings, such as filters or sort order, that you specified, or you can keep your settings.

To refresh the data and reset your settings, click ⋮ on the table viewer toolbar and select **Refresh** ⇨ **Table**.

To refresh the data and keep your settings, click ⋮ on the table viewer toolbar and select **Refresh** ⇨ **Data**.

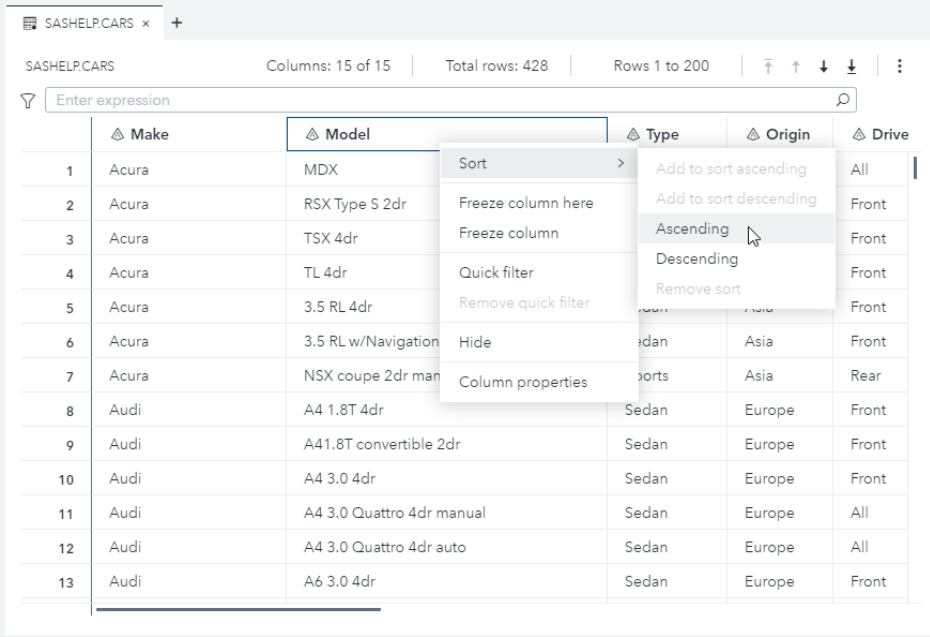
Sorting and Freezing Data

In the table viewer, you can right-click a column heading to sort the data by that column or freeze the data at that column. You can sort the data in ascending or

descending alphabetical order or display the columns in the order in which they appear in the data table.

To sort the data, right-click the appropriate column and select **Sort** ⇨ **Ascending** or **Sort** ⇨ **Descending**. If the data is already sorted by one or more columns, you can add another column to the sort criteria by right-clicking the column and selecting **Sort** ⇨ **Add to sort ascending** or **Sort** ⇨ **Add to sort descending**.

To freeze a column so that column and any columns to the left remain visible while you scroll through the remaining columns, right-click the appropriate column and select **Freeze column here**. To freeze only the selected column and move it to the leftmost position in the table viewer, right-click the column and select **Freeze column**.



	Make	Model	Type	Origin	Drive
1	Acura	MDX			All
2	Acura	RSX Type S 2dr			Front
3	Acura	TSX 4dr			Front
4	Acura	TL 4dr			Front
5	Acura	3.5 RL 4dr			Front
6	Acura	3.5 RL w/Navigation		Asia	Front
7	Acura	NSX coupe 2dr man		Asia	Rear
8	Audi	A4 1.8T 4dr	Sedan	Europe	Front
9	Audi	A41.8T convertible 2dr	Sedan	Europe	Front
10	Audi	A4 3.0 4dr	Sedan	Europe	Front
11	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All
12	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All
13	Audi	A6 3.0 4dr	Sedan	Europe	Front

Filtering Data

About Filtering Data

You can filter your data in order to display only rows that meet certain criteria, based on values in the data. You can filter data in the table viewer either by creating a quick filter based on a single column or by creating a filter expression with one or more columns.

Note: If you are creating a filter with the `Equal to` operator on a numeric column that includes values with a fractional component, you must create the filter by using a range of values to ensure that there are no rounding errors. For example, suppose

your data includes a value of `123.45678`. To create a filter that displays rows that include that value, you should use the following filter expression:

```
(column-name >= 123.456775 and column-name < 123.456785)
```

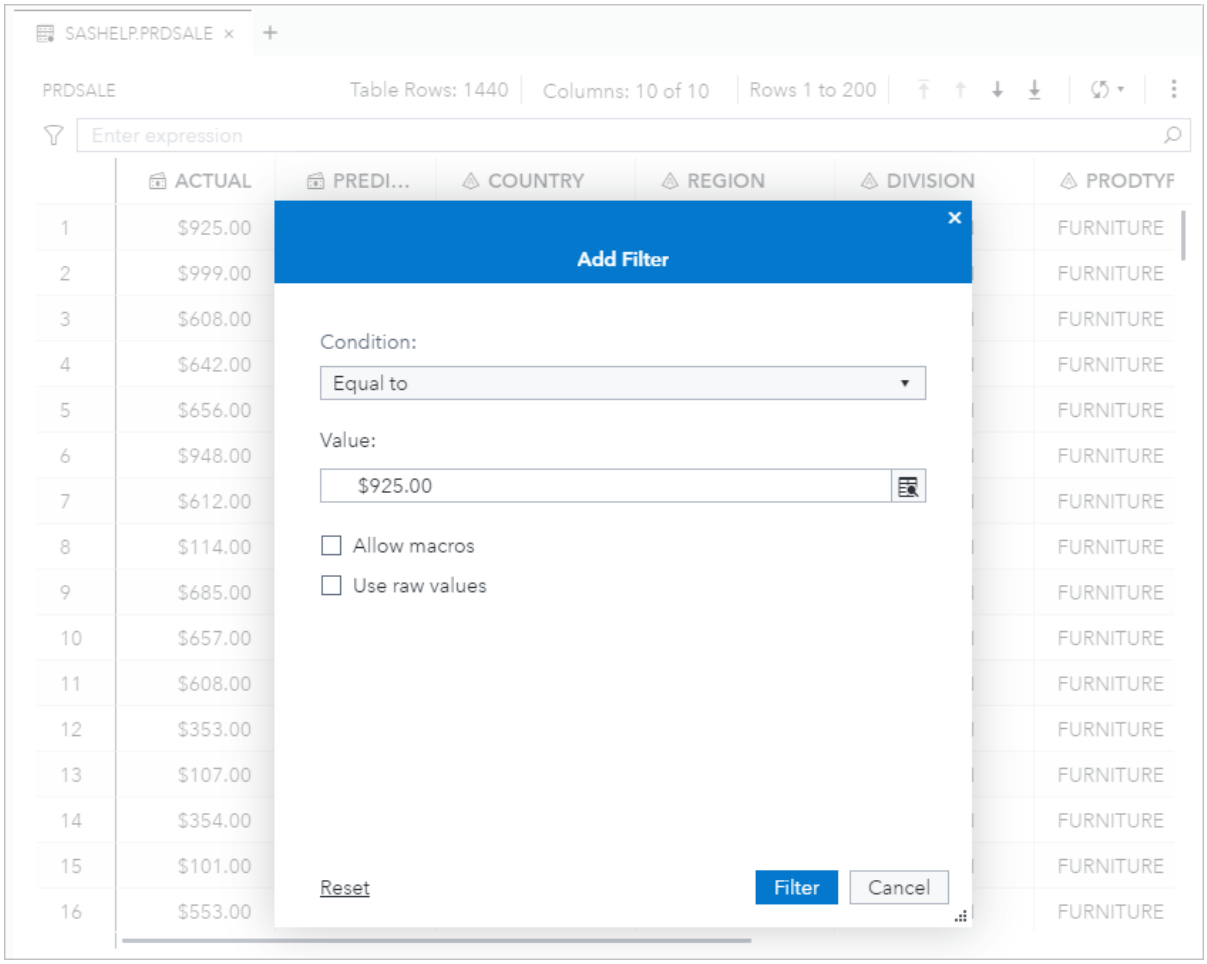
Note: If you want to search for a value when you are creating a filter, use the unformatted value.

Creating a Quick Filter

You can create a quick filter on a single column in your data by right-clicking the column heading and selecting **Quick filter**. In the Add Filter window, specify the operator and values that you want to use. You can either enter values or select from a list of values for the column.

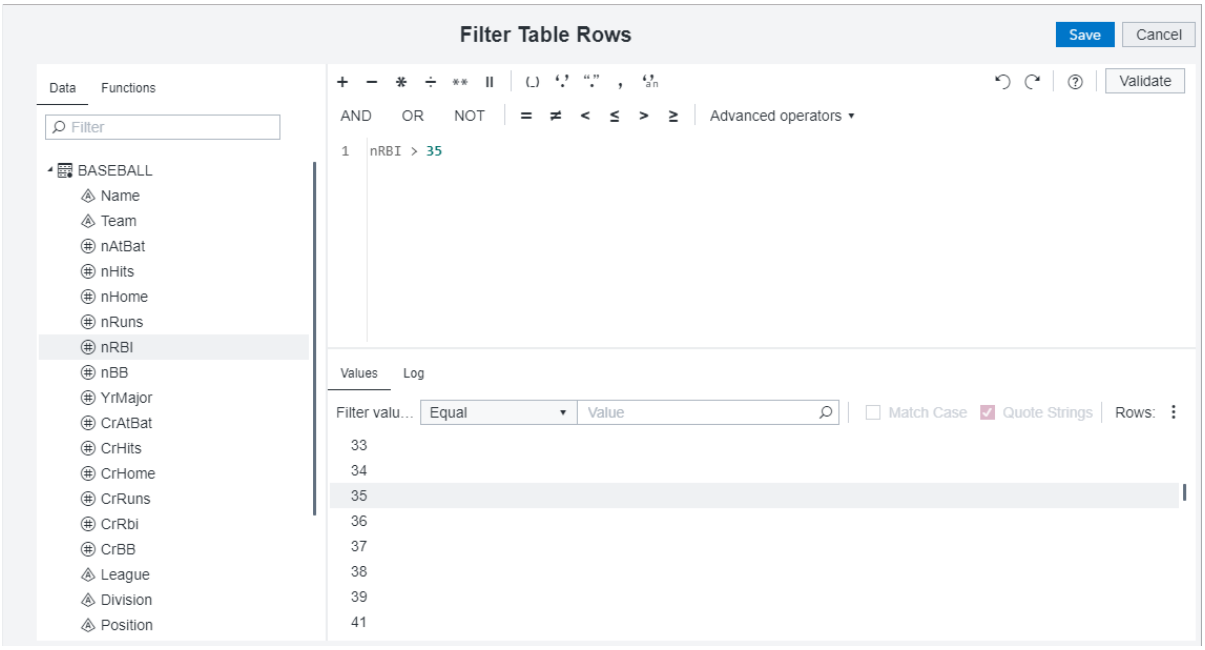
Depending on the data type of the column that you are using in the filter, you can choose from among the following options:

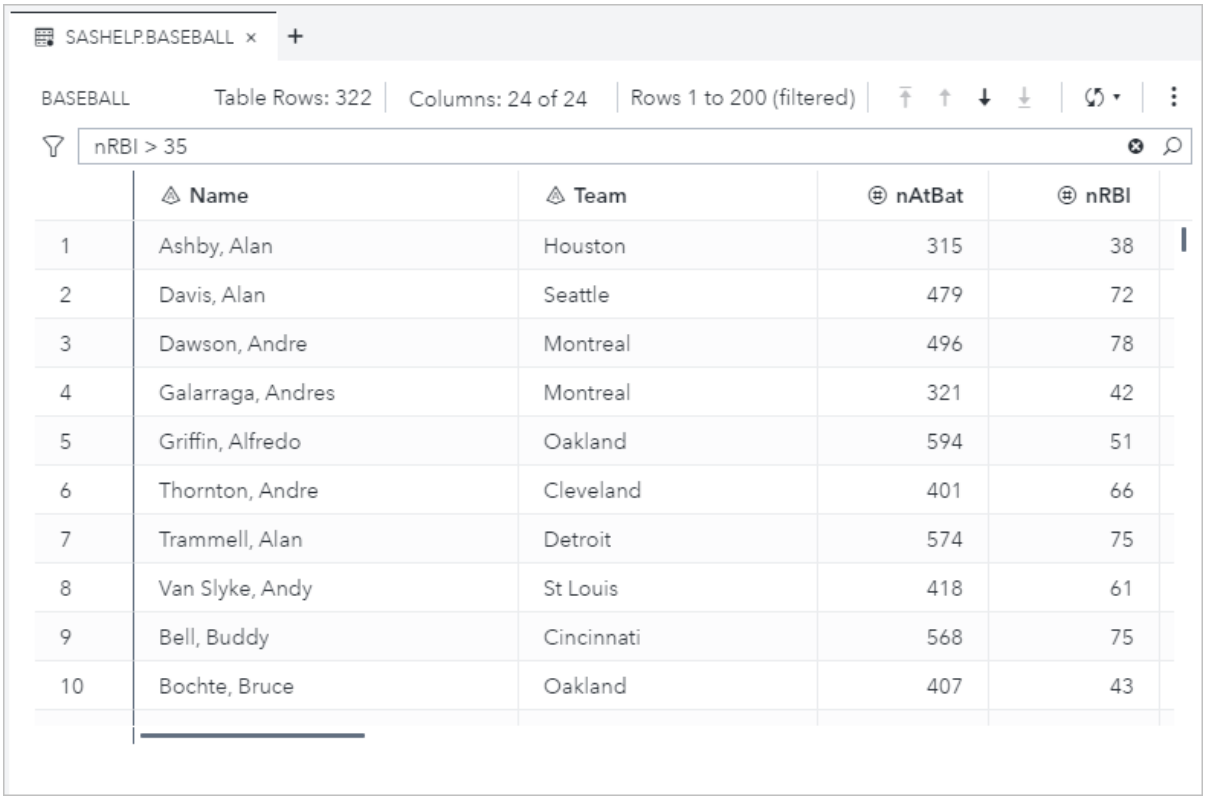
- **Match case** – retrieves only rows that match the capitalization of the value that you specify. If this option is not selected, the UPPER function is applied to the expression. This option is not selected by default and is available only for character columns.
- **Quote string** – encloses values in single quotation marks. This option is selected by default and is available only for character columns. If you are using a macro variable or other value that is evaluated when the filter is run, you should clear this option.
- **Allow macros** – enables you to enter character values in a filter on a numeric column. This option is not selected by default and is available only for numeric columns.
- **Use raw values** – uses unformatted numeric, date, time, and datetime values instead of the formatted values. This option is selected by default and is available only for numeric, date, time, and datetime columns. If you want to create a filter based on formatted values, clear this option.



Creating a Filter Expression

You can create a filter expression to filter your data by one or more columns. Click ▽ to use the expression builder in the Filter Table Rows window to create a filter expression. For more information, see “Building an Expression” on page 273.





	Name	Team	nAtBat	nRBI
1	Ashby, Alan	Houston	315	38
2	Davis, Alan	Seattle	479	72
3	Dawson, Andre	Montreal	496	78
4	Galarraga, Andres	Montreal	321	42
5	Griffin, Alfredo	Oakland	594	51
6	Thornton, Andre	Cleveland	401	66
7	Trammell, Alan	Detroit	574	75
8	Van Slyke, Andy	St Louis	418	61
9	Bell, Buddy	Cincinnati	568	75
10	Bochte, Bruce	Oakland	407	43

Importing Data

About Importing Data to SAS Studio

SAS Studio has two types of import functionality. The quick import enables you to quickly import data from the file system or from SAS Content by selecting a few options.

You can also import data into a flow. This import functionality is more robust and provides additional options. For more information, see “Importing Data from an External File” on page 113.

You can import these types of data files into SAS Studio:

- comma-delimited files (CSV).
- dBASE (dBase III, III+, IV, V).
- delimited file (DLM)
- Stata files (DTA).
- JMP files (JMP).
- Paradox DB files (DB).
- SPSS files (SAV).
- tab-delimited (TSV)
- XLS (Microsoft Excel 5.0, 95, 97, 2000-2003)
- XLSX (Microsoft Excel 2007 or later workbook)

If your data file is saved to your local computer, you must upload the file to SAS Studio before you can import it. For more information, see “Using the Explorer ” on page 5.

Whether data from another locale imports correctly depends on whether the SAS server supports the locale of the data that is being imported. If you are importing data that contains characters that are different from the current locale, use a Unicode (UTF–8) server to import your data. If you do not use a UTF–8 server and the locale of the data is not supported, unsupported values might appear as questions marks (?) in your imported data. For more information about how to set the **Default text encoding** option, see “Setting General Preferences” on page 261.

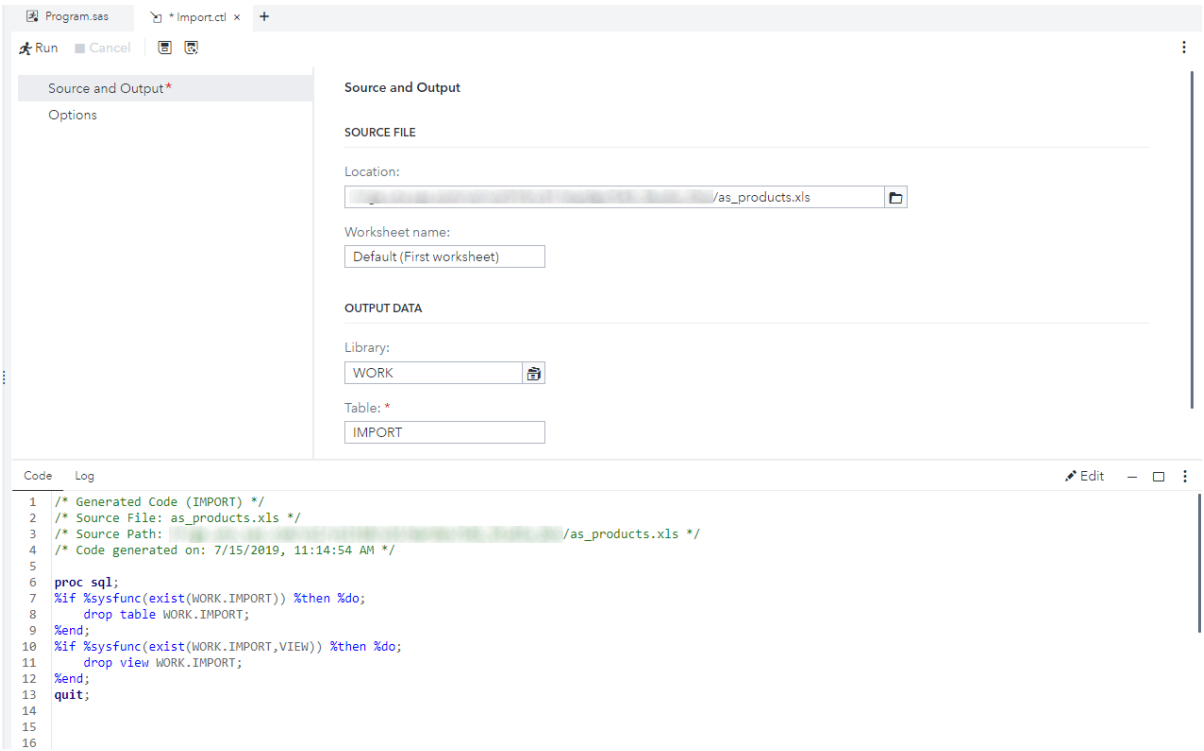
Note: You cannot import remote files (files that are available through FTP file shortcuts).

Import an Excel Worksheet

To import an Excel worksheet:

1. Select **New** ⇨ **Quick import**.
2. In the work area, click **Select Server File**.
3. Select the file that you want to import and click **Open**. The Import.ctl tab shows the name and location of this Excel file.

   This example shows importing the as_products.xls file.



4. To import the data from a specific worksheet, enter the name of that worksheet in the **Worksheet name** box. By default, SAS Studio imports the data from the first worksheet.
5. To specify the location to save the output data set, click . By default, the output data set is saved to the Work library, which is a temporary location. The contents in this library are deleted when you exit SAS Studio.
6. Click **Options**.
   - Select the file type.
   - To generate SAS variable names from the data values in the first row of the worksheet, select **Generate SAS variable names**. If a data value in the first row in the input file is read and it contains special characters that are not valid in a SAS name, such as a blank, then SAS converts the character to an underscore.
   - Specify the encoding.

7 To import the Excel worksheet, click .

Import a Delimited File (CSV, TSV, DLM)

Note: To import a tab-delimited file, the filename must have a TSV extension.

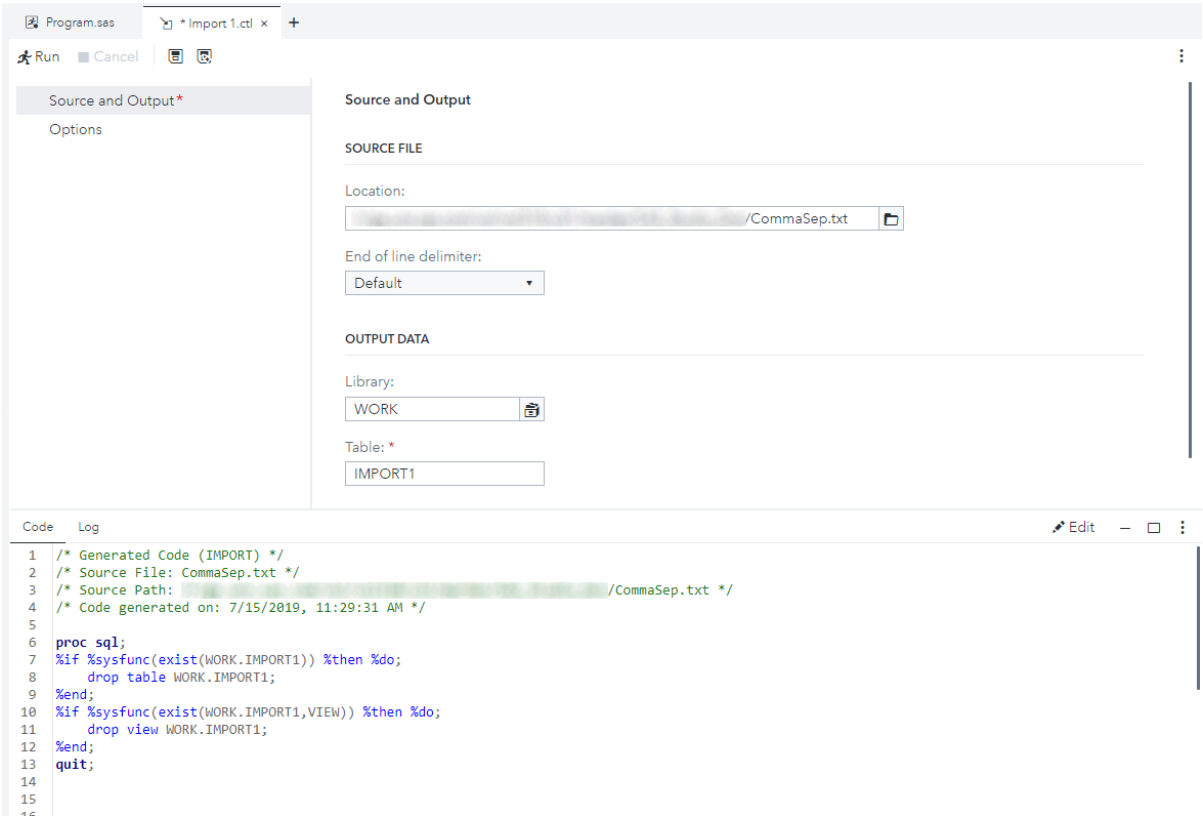
To import a delimited file:

1 Select **New** ⇨ **Quick import**.

2 In the work area, click **Select Server File**.

3 Select the file that you want to import and click **Open**. The Import.ctl tab shows the name and location of this text file.

This example shows importing a text file called CommaSep.txt.



4 To specify the location to save the output data set, click . By default, the output data set is saved to the Work library, which is a temporary location. The contents in this library are deleted when you exit SAS Studio.

5 Click **Options**.

- Specify the file type.
- To generate SAS variable names from the data values in the first row in the text file, select **Generate SAS variable names**. If a data value in the first row

in the input file is read and it contains special characters that are not valid in a SAS name (such as a blank), SAS converts the character to an underscore.

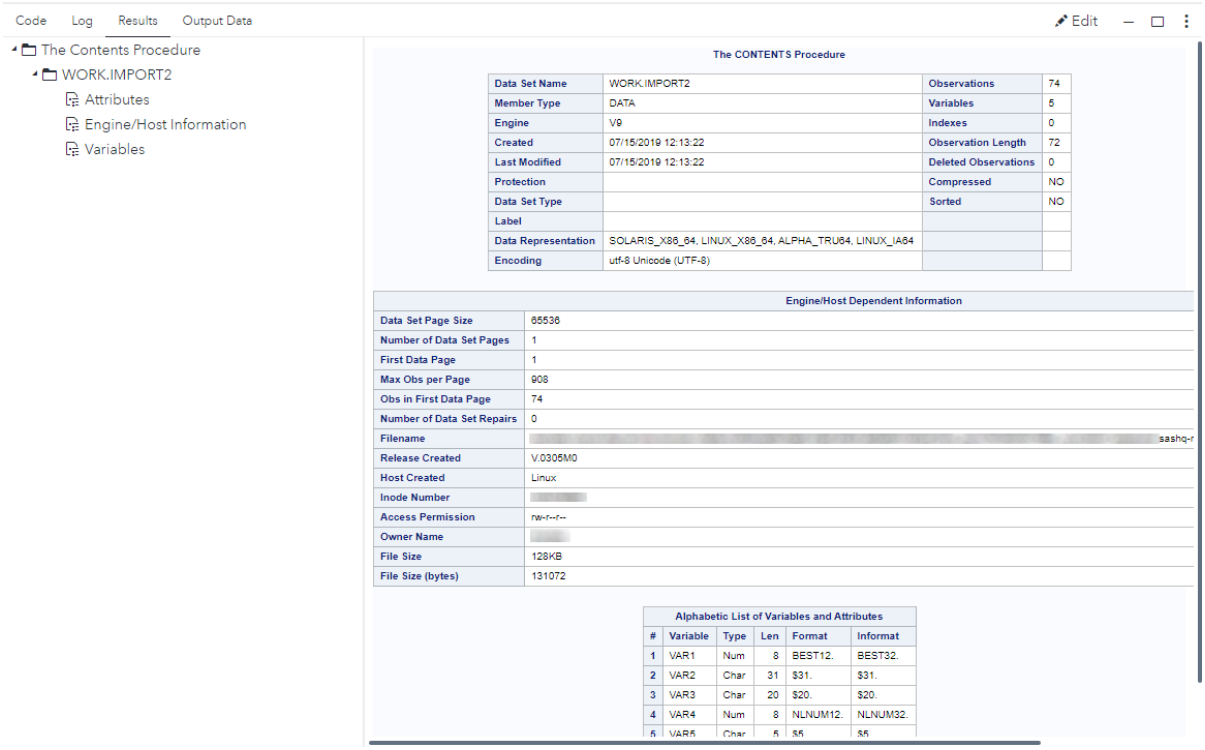
- In the **Delimiter** box, enter the delimiter for the values in the file. The default delimiter is a space.

  Note: If you use a hexadecimal value to specify the delimiter, you do not need to select the **Quote delimiter** check box.

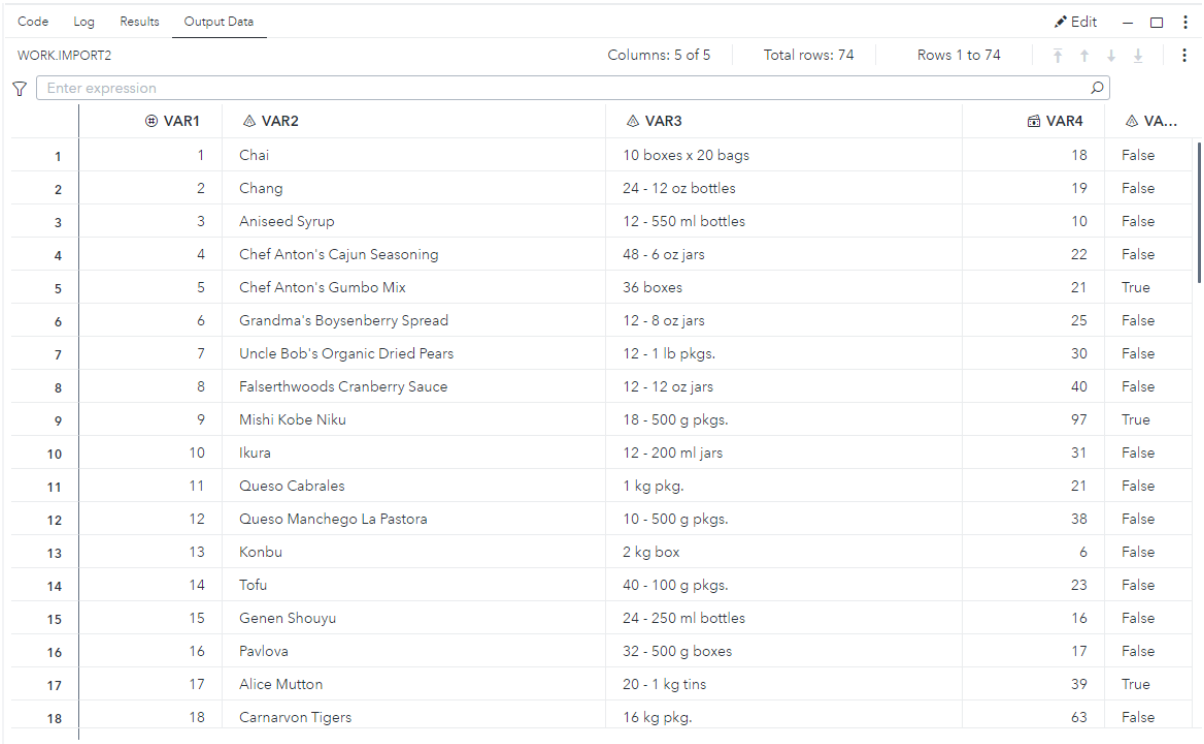
- To start reading data from a specified row in the delimited text file, enter the starting row in the **Start reading data at row** box. You might want to use this option if you have comments at the top of the text file or the first row of the file is column headings. By default, SAS Studio starts reading at row 2.
- For SAS Studio to determine the appropriate data type and length of the variables, enter a value in the **Guessing rows** box. The task scans the input data file from row 1 to the number that you specified. By default, the first 20 rows are scanned.
- Specify the encoding.

6 To import the data, click .

Click the **Results** tab to see the attributes of the imported data set.



Click the **Output Data** tab to view the new SAS data set. If this data set is in the Work library (as shown in this example), you might want to save it to a more permanent location. Data in the Work library is temporary and is deleted when you exit SAS Studio.



Code Log Results Output Data Edit

WORK.IMPORT2 Columns: 5 of 5 Total rows: 74 Rows 1 to 74

Enter expression

	VAR1	VAR2	VAR3	VAR4	VA...
1	1	Chai	10 boxes x 20 bags	18	False
2	2	Chang	24 - 12 oz bottles	19	False
3	3	Aniseed Syrup	12 - 550 ml bottles	10	False
4	4	Chef Anton's Cajun Seasoning	48 - 6 oz jars	22	False
5	5	Chef Anton's Gumbo Mix	36 boxes	21	True
6	6	Grandma's Boysenberry Spread	12 - 8 oz jars	25	False
7	7	Uncle Bob's Organic Dried Pears	12 - 1 lb pkgs.	30	False
8	8	Falserthwoods Cranberry Sauce	12 - 12 oz jars	40	False
9	9	Mishi Kobe Niku	18 - 500 g pkgs.	97	True
10	10	Ikura	12 - 200 ml jars	31	False
11	11	Queso Cabrales	1 kg pkg.	21	False
12	12	Queso Manchego La Pastora	10 - 500 g pkgs.	38	False
13	13	Konbu	2 kg box	6	False
14	14	Tofu	40 - 100 g pkgs.	23	False
15	15	Genen Shouyu	24 - 250 ml bottles	16	False
16	16	Pavlova	32 - 500 g boxes	17	False
17	17	Alice Mutton	20 - 1 kg tins	39	True
18	18	Carnarvon Tigers	16 kg pkg.	63	False

Import a DBMS File

When you import a file from a database management system (DBMS), the available options depend on the file type. For a list of the supported file types, see “About Importing Data to SAS Studio” on page 204.

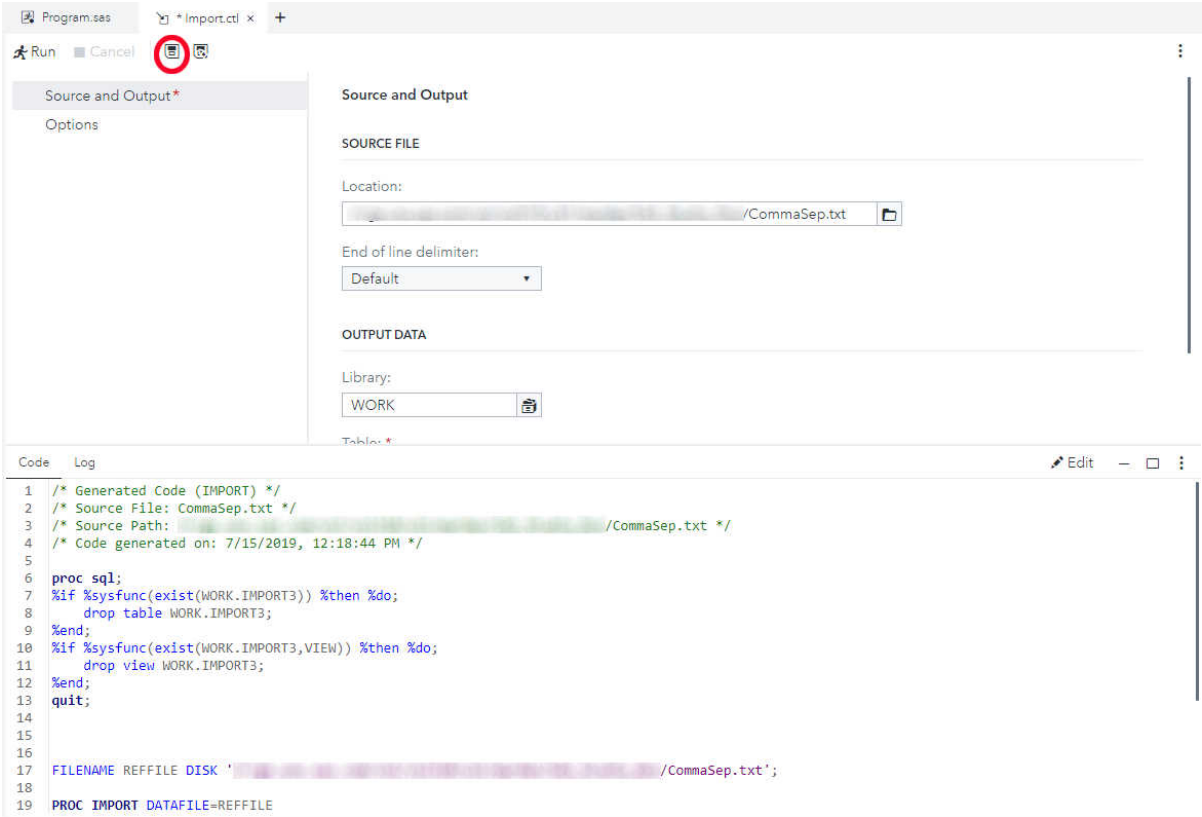
1. Select **New** ⇨ **Quick import**.
2. In the work area, click **Select Server File**.
3. Select the file that you want to import and click **Open**. The Import.ctl tab shows the name and location of this text file.
4. To specify the location to save the output data set, click . By default, the output data set is saved to the Work library, which is a temporary location. The contents in this library are deleted when you exit SAS Studio.
5. Click **Options**.
   - Specify the file type.
   - To generate SAS variable names from the data values in the first row in the text file, select **Generate SAS variable names**. If a data value in the first row in the input file is read and it contains special characters that are not valid in a SAS name (such as a blank), SAS converts the character to an underscore.
   - Specify the encoding.
6. To import the data, click .

Save the Import Data Task

You might want to save an instance of the Import Data task so that you can share these settings for importing a specific file with others at your site. SAS Studio saves these instances as a CTL file. CTL files must be run in the same operating environment where they were created. For example, if you create a CTL file using Windows, this CTL file must be run in Windows.

To save the import task:

1 Click .



2 In the Save As window, specify the name and location, and then click **Save**. The file is saved with a CTL extension.

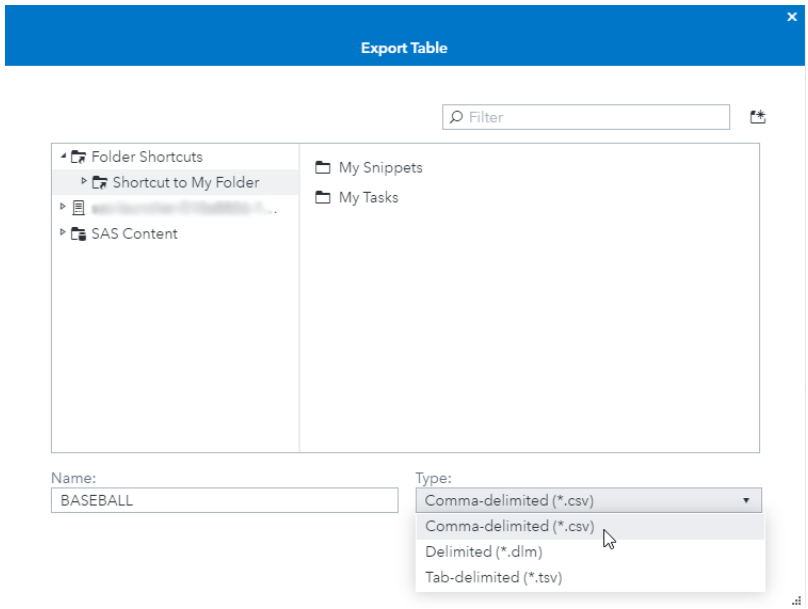
Exporting Data

In the Standard perspective, you can use SAS Studio to export your data as another file type to a folder that you specify.

Note: You cannot export password-protected data in SAS Studio, and you cannot export data from the Interactive perspective. For more information, see "Understanding Perspectives" on page 25.

To export your data:

1. Click **Libraries** in the navigation pane and browse to find the file that you want to export.
2. Right-click the file that you want to export and select **Export**. The Export Table window opens.
3. Select the folder in which you want to save the exported file.
4. In the **Filename** box, enter the name of the exported file.
5. From the **Type** drop-down list, select the format of the exported file.



6. Click **Save** to export the file.

Note: When you export data, the entire file is exported. Any filters you might have applied to the data in the table viewer are not applied to the exported data.

7

Working with Results and Output Data

Viewing Results

When you run a task or a program in SAS Studio, the results are displayed in the work area. By default, results are generated as HTML5 output and displayed on the **Results** tab. Here are ways that you can work with your results:

- Add additional output formats, such as PDF, Word, RTF, Excel, and PowerPoint, by using the Preferences window.
- Download your generated output. You can download results that have been created in any of the available formats.
- Change the default output style for each format by using the Preferences window.

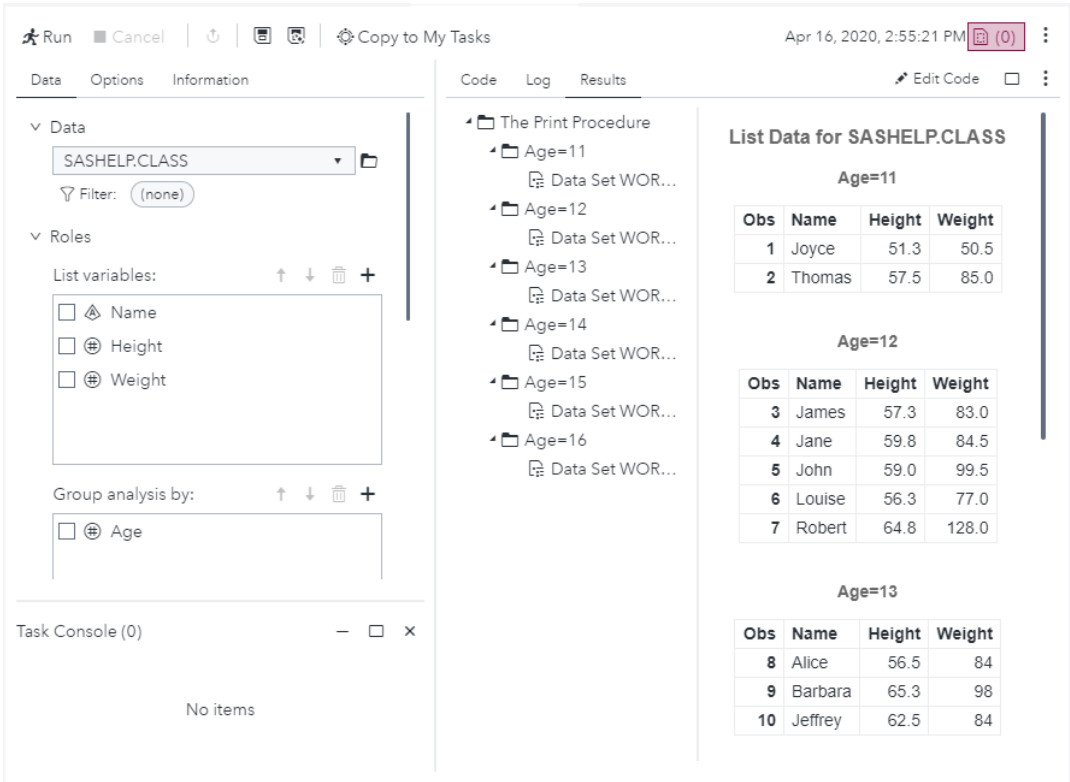
Default SAS Studio Output

Viewing Default Results

In SAS Studio, by default, output is generated in HTML5.

If you want to add output formats, you can use the Preferences window to select other formats, such as PDF and Word. You can also change the default style for your output to any of the ODS styles that are available. For more information, see "Setting the Results Preferences" on page 264.

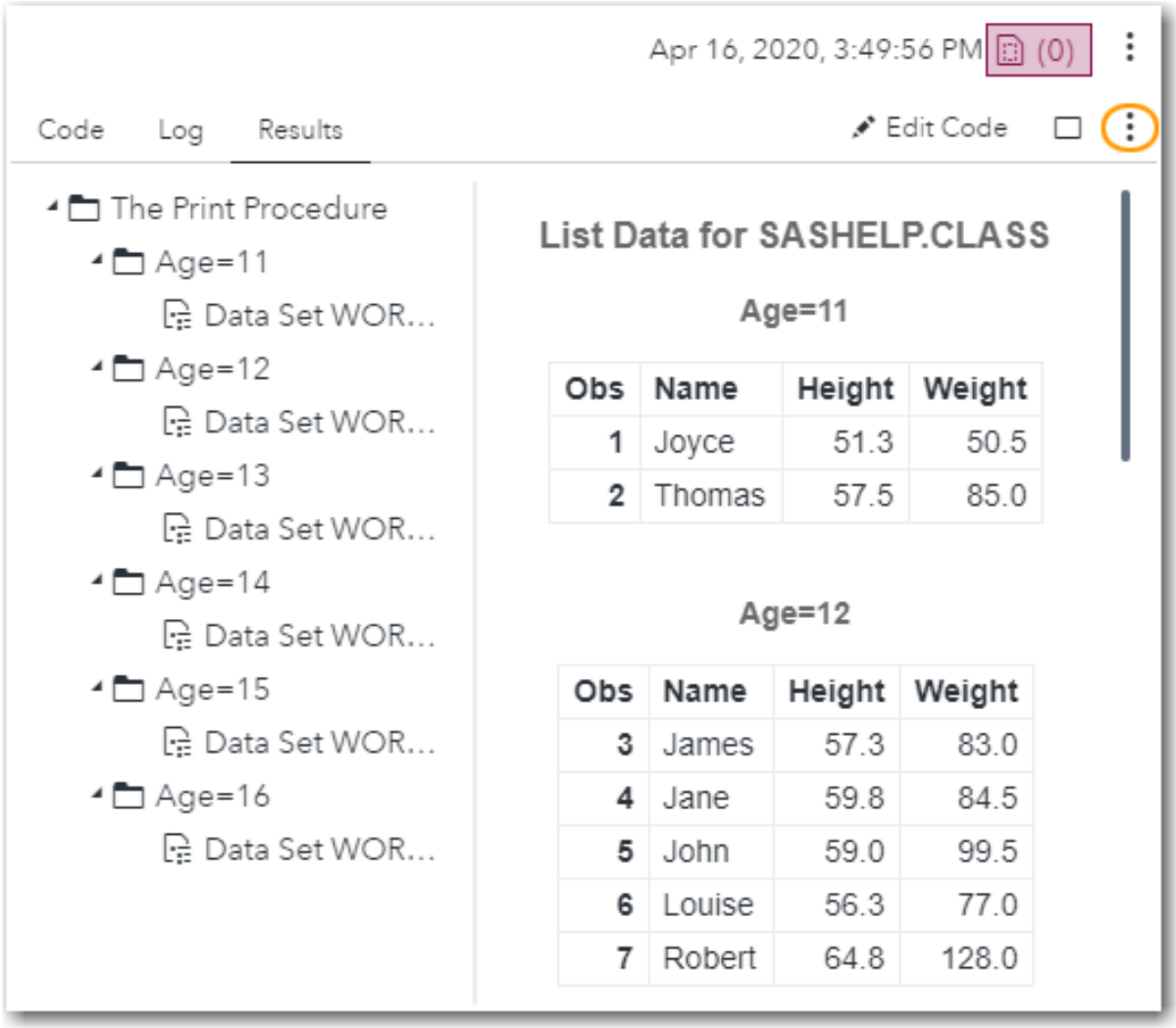
By default, the HTML5 results are the only results that are displayed on the **Results** tab. When you view your results, you can use the table of contents to navigate the different sections. By default, the table of contents is expanded, and you can click the section that you want to navigate to.



Downloading Generated Data

To download results that were generated in other formats, such as PDF or RTF, or as other types of data, such as a .csv or XML file:

1 Click ⋮ for the **Results** tab.



2 Select **Download** and the type for the results.

Sending Your Results to Another User

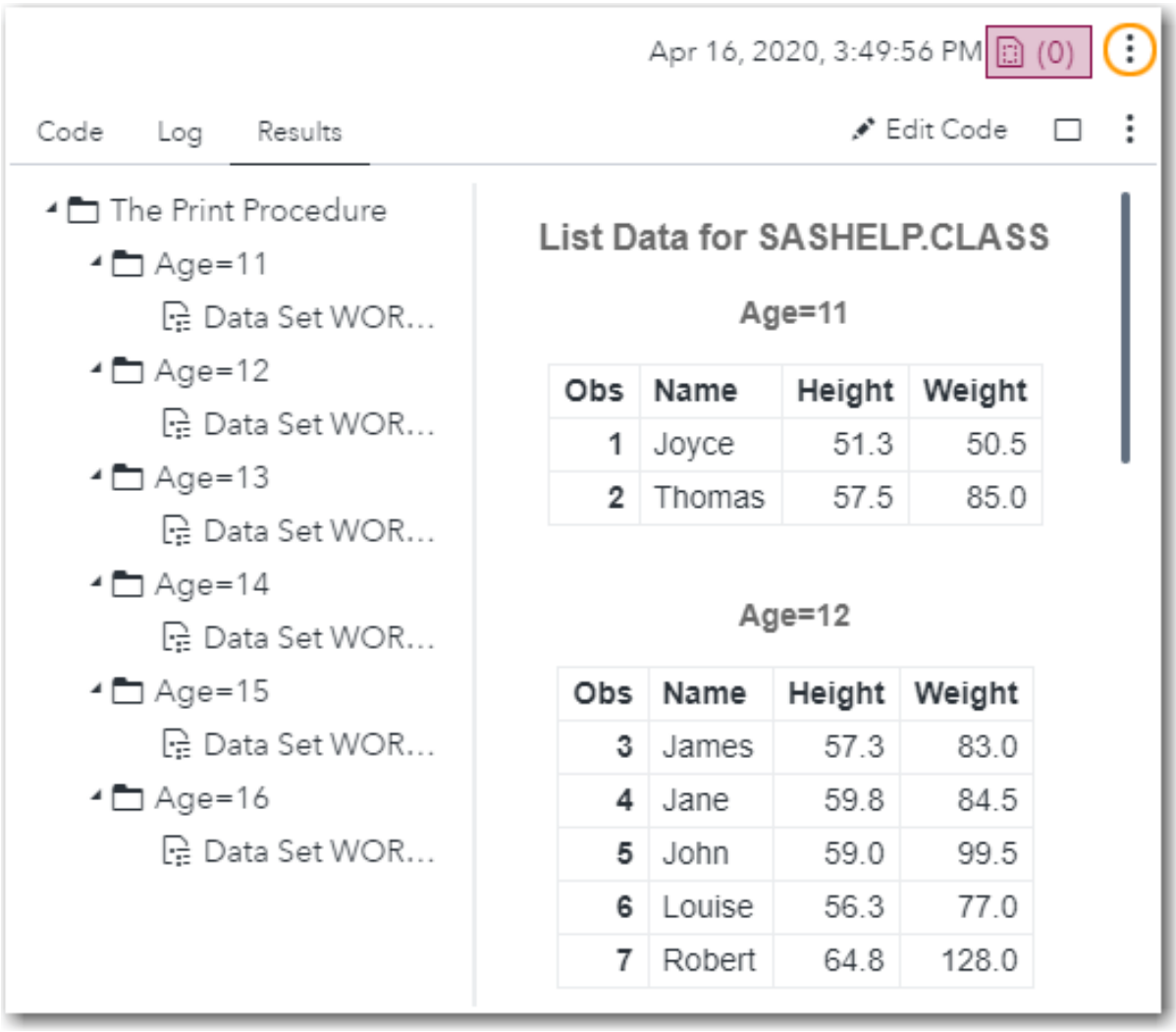
You can send a copy of your results and the associated code and log files to another user through email. Files that you can send include results in HTML5, RTF, and PDF formats as well as the code and log files that are associated with the

results. The code is sent as a SAS program file, and the log and program summary files are sent as HTML5 files. To send files through email, you need access to an SMTP server. For more information, contact your site administrator.

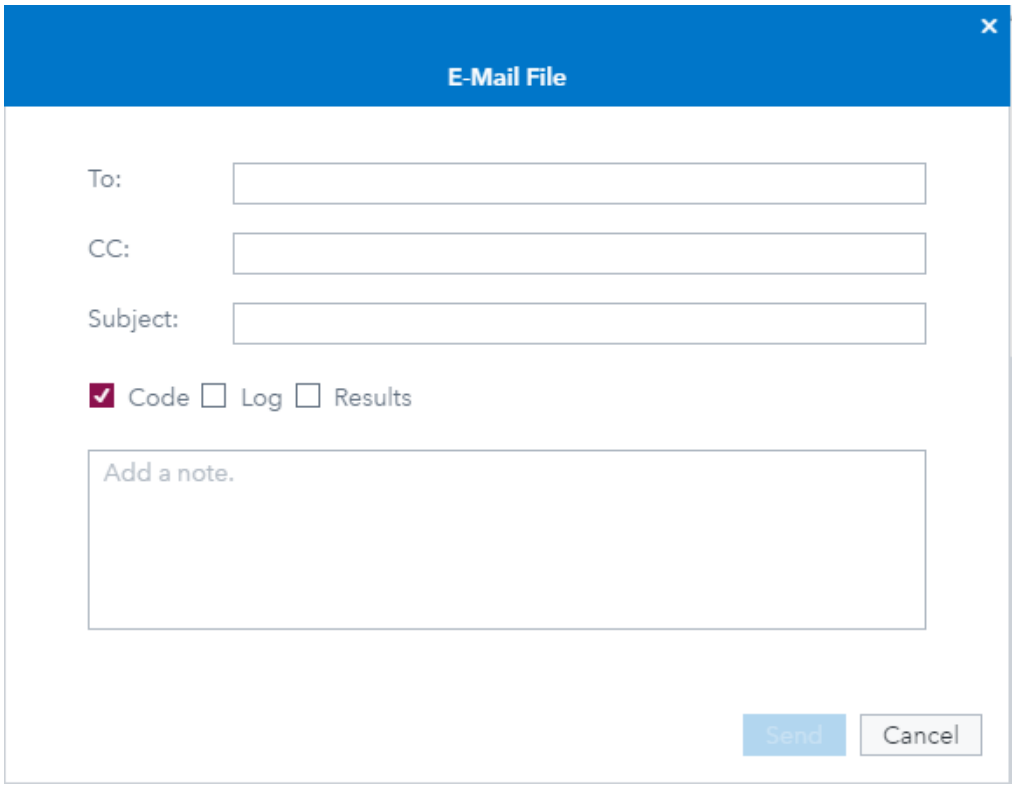
Note: If your SAS Studio email messages are being marked as junk, see "Configuring the Mail Server" in *SAS Studio: Administrator's Guide*.

To send results by email:

1 On the toolbar for the program, query, or task, click ⋮.



2 Select **E-Mail**. The E-Mail File window appears.



3 In the **To** box, enter the email addresses to which you want to send the files. Separate addresses with a semicolon.

4 If you want to send a copy of the email to another address, enter the address in the **CC** box.

5 In the **Subject** box, enter a subject for the email. You can also add a message to include in the body of the email.

6 Select the items that you want to include as attachments to your email. By default, the code is selected. If you select **Results**, all results that were generated are included in the email.

7 Click **Send** to send the message and attachments.

Viewing Output Data

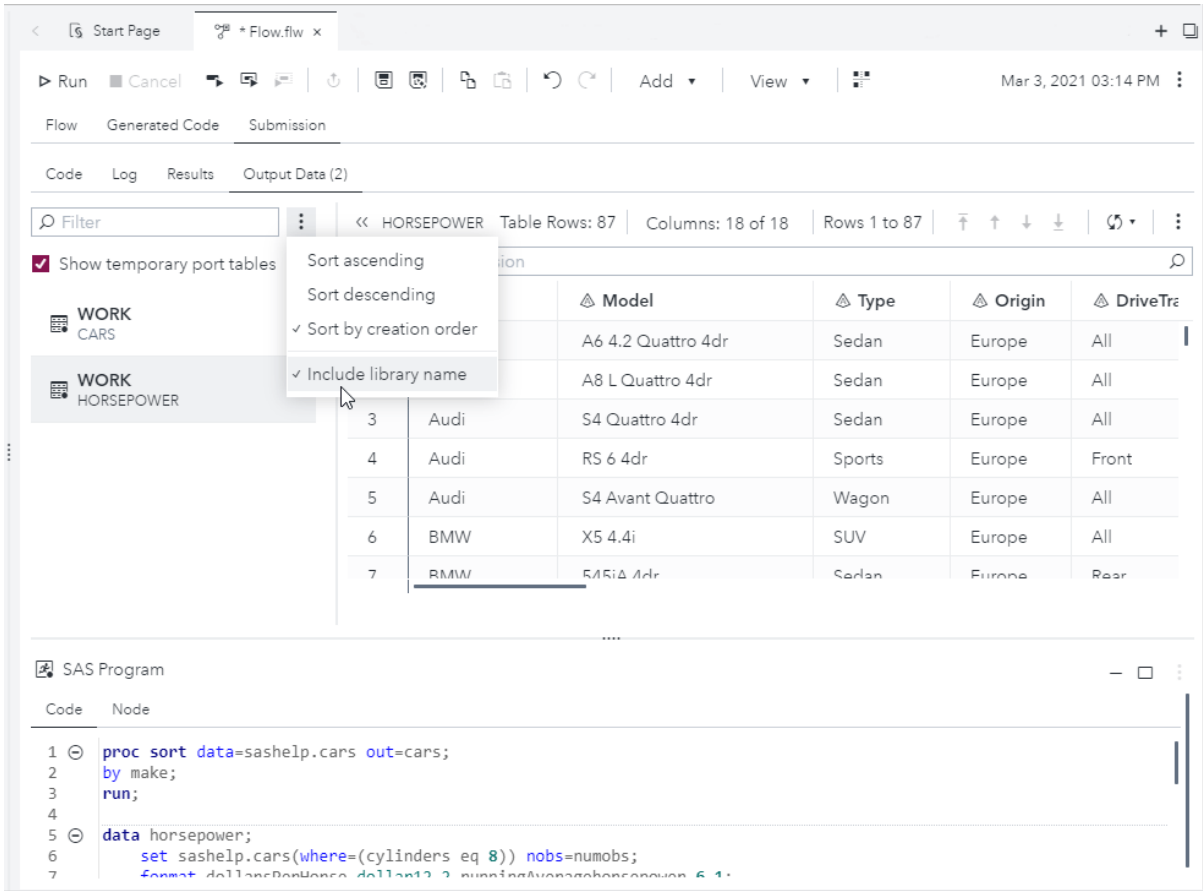
When you run a program, task, query, or flow that generates a table or view, the output data is displayed on the Output Data tab. You can view and use data on the Output Data tab in the same ways that you can use a table or view that you open from the **Libraries** section of the navigation pane. For more information, see "About the Table Viewer" on page 191.

By default, if your program, task, or flow generates multiple output tables, the list of tables is displayed in a collapsible pane.

Note: You can use the **View output data in a pane** option to switch between displaying the list of output tables in a pane or in a compact menu. For more information, see “Setting General Preferences” on page 261.

You can customize how the output tables are displayed in the pane in these ways:

- Use the filter box at the top of the list to specify filter criteria for the table names.
- Specify a sort order for the tables by clicking ⋮ and selecting the appropriate order. By default, the tables are displayed in the order in which they are created.
- Specify whether to include the library name of each table by clicking ⋮ and selecting or clearing **Include library name**.
- If the output tables are in a flow, use the **Show temporary port tables** option to specify whether you want to include tables that are associated with output ports but not connected to Table nodes.



About the SAS Output Delivery System and SAS ODS Statistical Graphics

The SAS Output Delivery System (ODS) gives you greater flexibility in generating, storing, and reproducing SAS procedure and DATA step output along with a wide range of formatting options. ODS provides formatting functionality that is not available when using individual procedures or the DATA step without ODS.

SAS Studio uses very specific ODS options and the GOPTIONS statements so that the output is displayed properly in the web environment. To view all of the ODS options in your code, click **Options** > **Preferences**. In the Preferences window, select **SAS Programs** > **Code and Log**, and then select the **Show generated code in the SAS log** option.

Note: To ensure that your output is displayed properly, do not change the settings of the ODS options or GOPTIONS statements in the generated code.

SAS ODS Statistical Graphics, more commonly referred to as SAS ODS Graphics, is an extension of the SAS Output Delivery System (ODS). ODS manages all output that is created by procedures and enables you to display the output in a variety of forms, including HTML and PDF.

Many SAS analytical procedures use ODS Graphics functionality to produce graphs. ODS Graphics uses the Graph Template Language (GTL) syntax, which provides the power and flexibility to create many complex graphs. The GTL is a comprehensive language for defining statistical graphics.

8

Understanding Git Integration in SAS Studio

About Git Integration in SAS Studio

SAS Studio includes integration with Git, a system for tracking changes and managing version control among multiple users. Git can be used with many different repository hosting services such as GitHub and Bitbucket.

Note: This document assumes that you have a basic familiarity with Git technology and your repository hosting service.

Here are the requirements for using the Git functionality in SAS Studio:

- Standard perspective. For more information, see "Understanding Perspectives" on page 25.
- Red Hat Enterprise Linux (RHEL), Windows WX6, or Windows W32 operating system. If you are using a SUSE Linux Enterprise Server, you need to install additional libraries. Contact SAS Technical Support at support@sas.com for assistance.

  Note: If you are using a RHEL8 deployment, you must install the compat-openssl10 rpm. For more information, see "Using Git Functions on a RHEL 8 Deployment" in *SAS Studio: Administrator's Guide*.
- Your SAS Administrator must have specified the Git configuration properties. For more information, see "Configuration Properties for Git Integration" in *SAS Studio: Administrator's Guide*.

You can use the **Git Repositories** ( ) section of the navigation pane to access the Git features that are available from within SAS Studio. These features include cloning repositories, committing and stashing file changes, pulling and pushing files, viewing your repository history, creating and merging branches, rebasing a branch, and performing a basic differentiation between files in your local repository. By default, SAS Studio uses SSH keys to authenticate with your repository hosting service.

Note: For information about authenticating using HTTPS with a user name and password, contact your site administrator.

You can also interact with a Git repository from SAS by using SAS Git functions in a SAS program. For more information, see "Using Git Functions in SAS" in *SAS Functions and CALL Routines: Reference*.

Working with Git Profiles

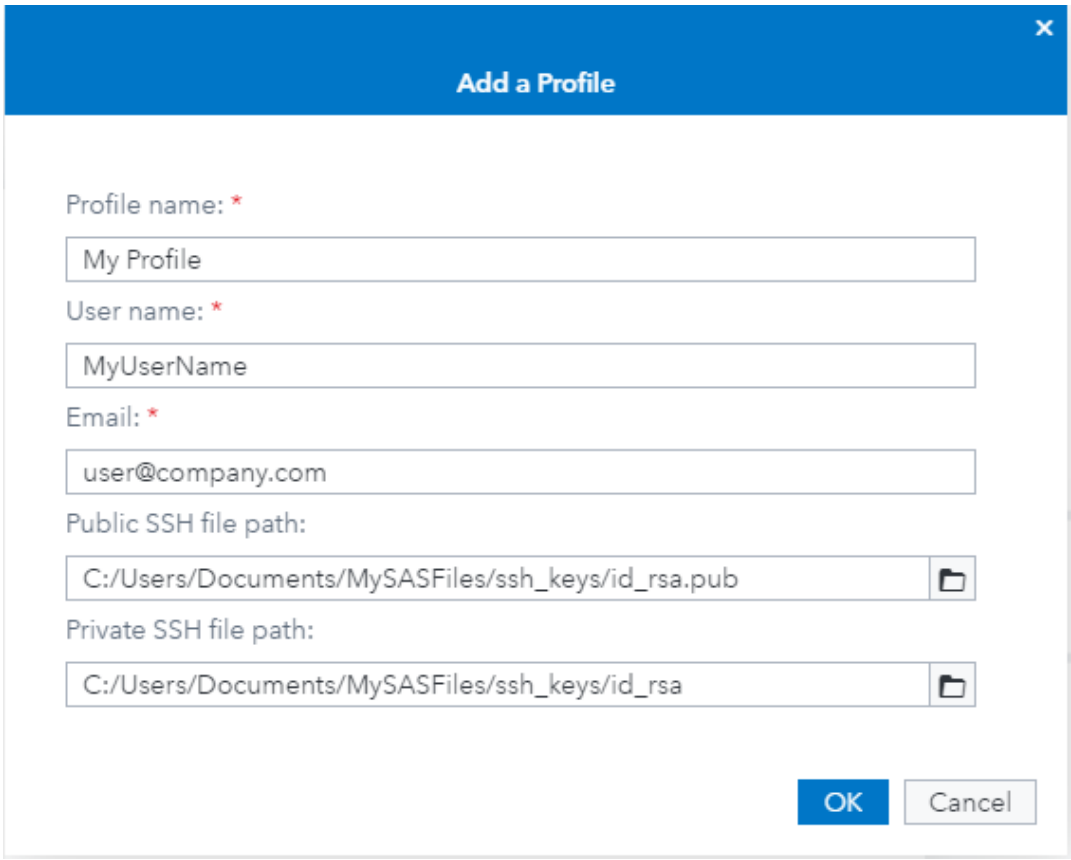
Creating a Git Profile

Before you can get started using any of the Git features in SAS Studio, you must create a Git profile. By default, SAS Studio uses SSH keys for authentication. In order to use the SSH authentication, you must upload your public and private SSH keys to the server file system. For information about generating SSH keys, see the documentation for your repository hosting service.

> **TIP** You can upload your SSH key files to the server file system by selecting the appropriate directory in the **Explorer** section of the navigation pane and clicking . Click + and browse for the SSH key files. Select the files and click **Open**. Click **Upload** to upload the files to the directory on your server file system.

To create a Git profile, select **Options** ⇨ **Manage Git Connections** and click +. The Add a Profile window appears.

To create a Git profile using SSH keys, enter your profile name, user name, and a valid email address in the appropriate boxes. Click **Browse** to find your public and private SSH key files or enter the path names in the appropriate boxes. Click **OK** to create your profile.



To create a Git profile using HTTPS authentication, contact your site administrator.

> **TIP** To specify a default profile, select **Options** ⇨ **Manage Git Connections** and click **Profiles**. Select the profile that you want to use as your default profile, and click **Set as default**.

Deleting a Git Profile

To delete a profile, select **Options** ⇨ **Manage Git Connections** and click **Profiles**. Select the profile that you want to delete and click 🗑.

If the profile is used in the active repository, you cannot delete it until you close the active repository. If you delete a profile that is associated with other repositories, you cannot open those repositories until you use the Manage Git Connections window to associate them with another profile.

Cloning and Opening a Git Repository

In order to access the files from a Git repository in SAS Studio, you must clone the repository to a local repository on your server file system. When you clone the repository, a copy of all the files on the remote repository are copied to a working directory that you specify on your server file system. You can clone a repository from a remote server using SSH or HTTPS authentication, and you can also clone a local repository from your server file system.

Note: If an existing Git repository has been cloned to the server file system, then the Git repository is already a local repository and is available for you to access. However, if the Git repository is not on the server file system, you must clone it before you can access it.

You can access the files in the cloned repository by navigating to the directory in the **Explorer** section of the navigation pane. You can make changes to the files in the working directory, and then commit those changes to your local repository.

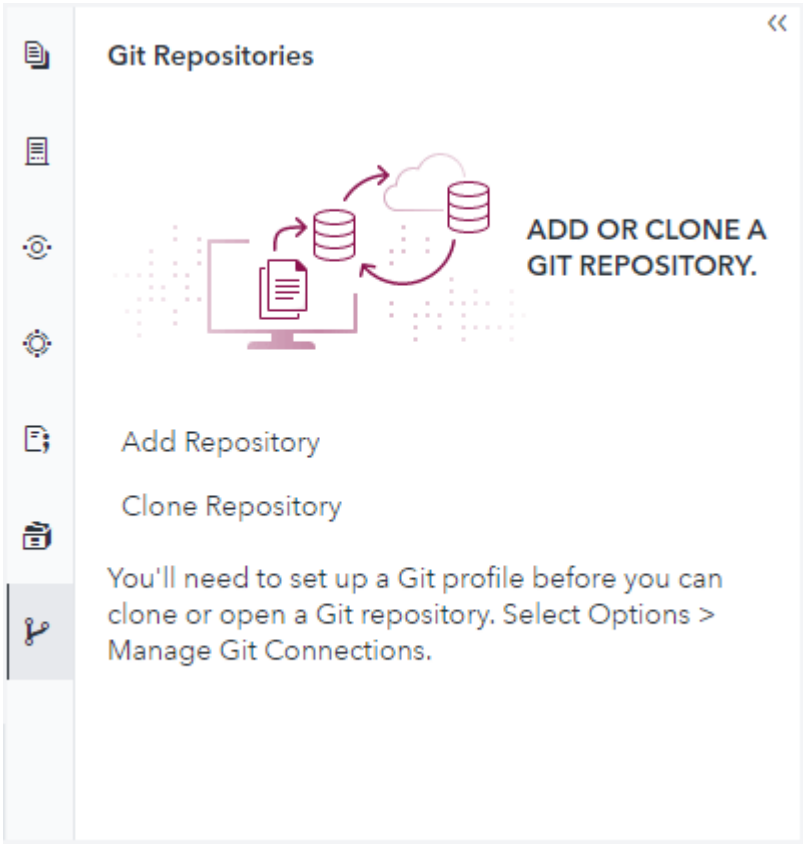
Note: Changes that you make to files in your local repository are not visible to other users until you push those changes to the remote repository.

To clone a Git repository:

1 In the navigation pane, click , and then click **Clone Repository** or click **Clone** on the toolbar of the **Git Repositories** section of the navigation pane. The Clone a Repository window appears.

TIP You can access the option to clone a repository in three ways:

- In the **Git Repositories** section of the navigation pane, click **Clone Repository**. This option is available only if there are no repositories listed in the **Git Repositories** section of the navigation pane.
- Click **Clone** on the toolbar of the **Git Repositories** section of the navigation pane. This option is available only if there are already repositories listed in the **Git Repositories** section of the navigation pane.
- Select **Options** ⇨ **Manage Git Connections**. In the Manage Git Connections window, click **Repositories**. Click + and select **Clone a repository**.



Note: To open an existing local repository, click **Add Repository** or use the **Add** toolbar button in the **Git Repositories** section of the navigation pane. In the Select a Folder window, browse to find the repository that you want to open and specify the Git profile.

2 In the **Repository** box, enter the location of the repository that you want to clone. You can clone a repository in three ways:

- Use SSH keys to authenticate with your repository hosting service - enter a repository name in a format similar to one of these examples:

  ```
  git@github.com:folder/sasgit.git
  ```

  or

  ```
  git@github.com:jquery/jquery-ui.git
  ```

  Note: By default, SAS Studio uses SSH keys to authenticate with your repository hosting service.

- Use HTTPS authentication - enter a repository name in a format similar to this example:

  ```
  https://github.com/example/repo.git
  ```

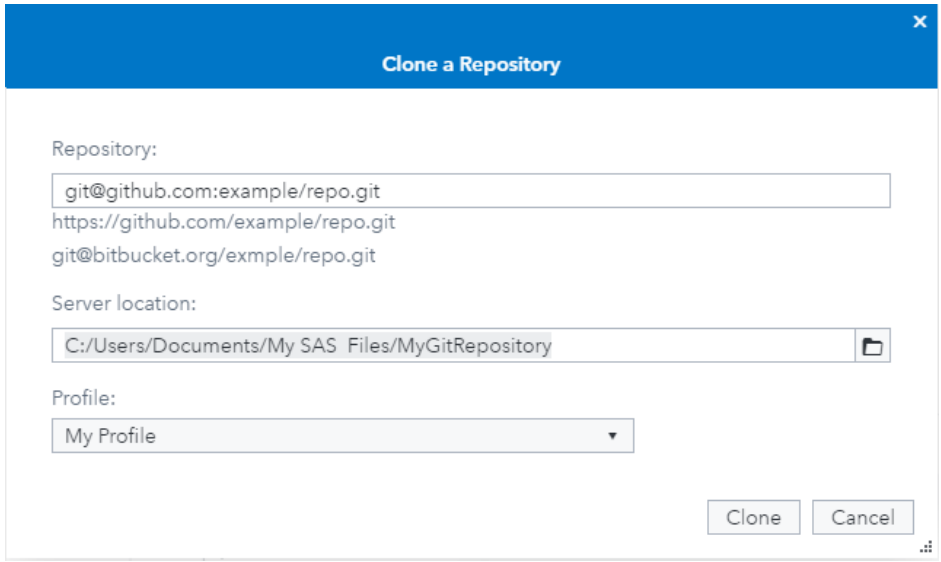
- Use a local repository - enter the full pathname of the local repository on your server file system.

Note: When you clone a local repository, you can pull files from the repository, but you cannot push any changes.

3 In the **Server location** box, enter the path name of a folder on your server file system for the cloned repository.

Note: You must select an empty folder for the cloned repository. If there are any files in the folder, the cloning operation cannot be completed.

4 From the **Profile** drop-down list, select the profile that you want to use to connect to the Git repository.



5 Click **Clone** to clone the repository. A Git repository tab opens in the work area. You can also view the files in the cloned repository by using the **Explorer** section of the navigation pane.

Note: You can rename a repository from the **Explorer** section of the navigation pane. Right-click the repository folder and select **Properties**. Enter the new name in the **Name** box.

Creating a Local Repository

You can create a local repository from a server file system folder in the **Explorer** section of the navigation pane. You can access the local repository in the **Git Repositories** section of the navigation pane and track and commit your changes. You can also associate the local repository with a remote repository and push your changes to the remote repository.

Note: You cannot create a local repository from a SAS Content folder.

To create a local repository:

1. In the **Explorer** section of the navigation pane, right-click the folder that you want to use as a local repository and select **Git initialize**. The folders tree refreshes and the folder is identified in the tree as a Git repository.
2. To view the new local repository, click in the navigation pane. Double-click the new local repository to open it in the work area, if it does not open automatically. Notice that all of the files in the folder have been added to the repository as unstaged changes. To add the files to the repository, you must first stage the files and then commit them. For more information, see "Committing Changes to Your Local Repository" on page 228.
3. To push the committed files to a remote repository, click **Push** on the Git repository tab. In the Create Remote Repository window, enter the location of the remote repository and click **Create**.

Understanding the Git Repository Tab in SAS Studio

When you clone a Git repository or open an existing Git repository from your server, a Git repository tab opens in the work area.

The Git repository tab contains two subtabs: History and Commit. The History subtab displays the history of all the changes, or commits, in a repository and enables you to view the differences between the selected and parent commits. The Commit subtab enables you to stage, unstage, reset, and commit changes to files in a repository. You can also view the differences between versions of a file.

To open your local Git repository tab, click in the navigation pane and double-click the repository that you want to open. The Git repository tab opens in the work area.

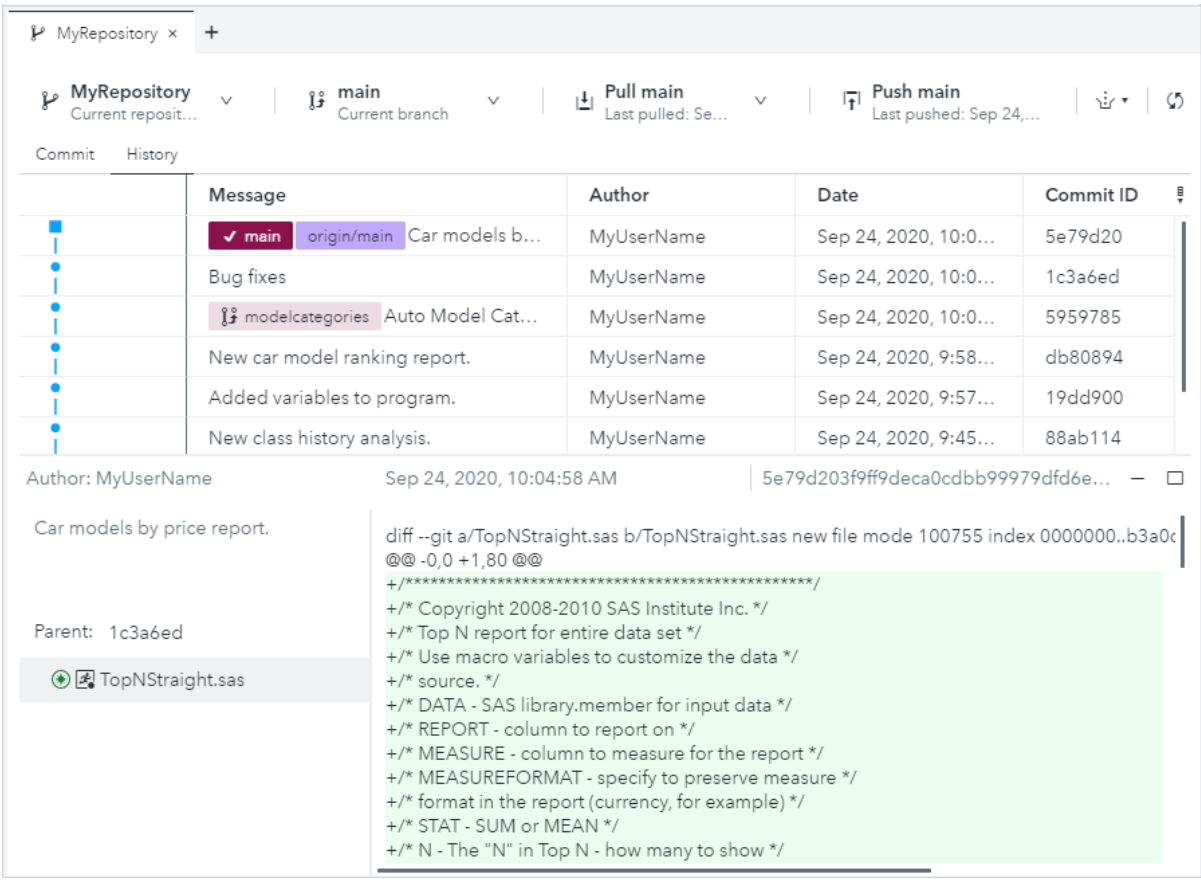
Note: You can view only one Git repository in SAS Studio at a time.

From the main Git repository toolbar, you can perform these tasks:

- Specify the local repository that you want to open.
- Check out a local branch.
- Pull or fetch data from the remote repository. For more information, see "Pulling and Fetching Files" on page 232.
- Push committed changes to the remote repository. For more information, see "Pushing Files" on page 232.
- Stash and reapply changes. For more information, see "Stashing Changes" on page 237.

- Refresh the local repository.

Note: You can refresh the **Git Repositories** section of the navigation pane by clicking ⋮ on the Git Repositories tab and selecting **Refresh**.



Viewing the Commit History

You can use the commit history to view the history of all the changes to a repository. You can also view the details of a specific commit as well as the differences between two commits.

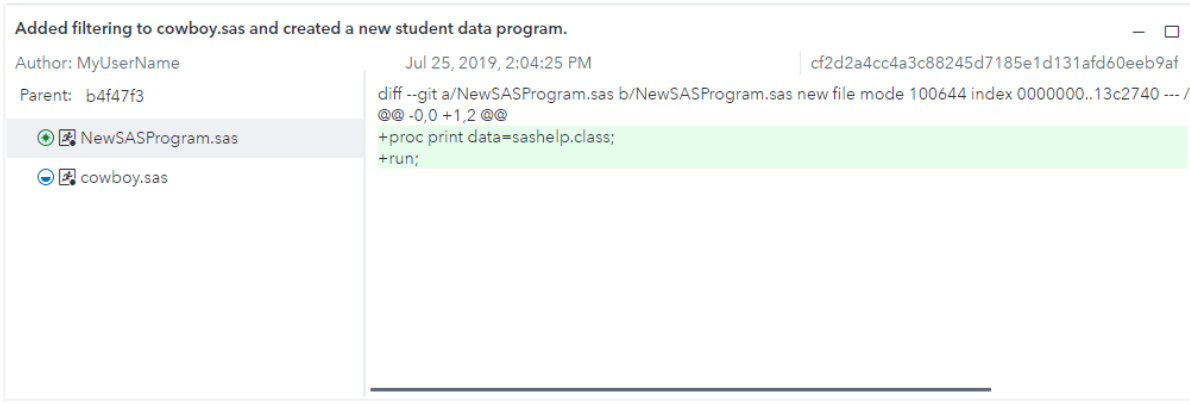
After you have saved your changes to the files in your working directory, make sure the tab for your local Git repository tab is open in the work area. Click the **History** subtab. A list of all the commits made in that repository is displayed in the work area in reverse chronological order. For each commit, the commit message, the author, the date and time of the commit, and the commit ID are displayed by default. To modify which columns are displayed, click ⋮ and select the columns that you want to be displayed.

Note: By default, up to 1,000 rows are loaded in the commit history. You can specify a different limit by selecting **Options** ⇨ **Manage Git Connections**. Click

Options and enter a new value in the **Maximum rows loaded in history log** option. Increasing the number of rows that are loaded might affect your performance.

To view the details of a specific commit, select the commit from the commit history. The details of the commit are displayed in the lower section of the work area. You can view the details of the parent of the selected commit by clicking the parent commit ID in the details.

To view the differences between changes in the selected commit and the parent commits, click a file in the lower section of the work area. You can also compare two adjacent commits from the commit history list by selecting the commits that you want to compare.



Note: To view the file differences in a separate window, right-click the differences section of the work area and select **Open in new window**.

Committing Changes to Your Local Repository

Before you can push one or more changes from your local Git repository to the remote repository, you must first commit the changes from your working directory to your local repository. When you commit files, you stage the files that you want to include in the commit by moving the files from the Unstaged changes area to the Staged changes area. Next, you must enter a comment describing the commit, and then commit the files. When you select files to commit, it is a good practice to group changes that are related and represent a state in your project development that you want to capture.

Note: When you commit a change to the local repository, those changes are still not viewable by other users, but the changes are available for you to push to the remote repository.

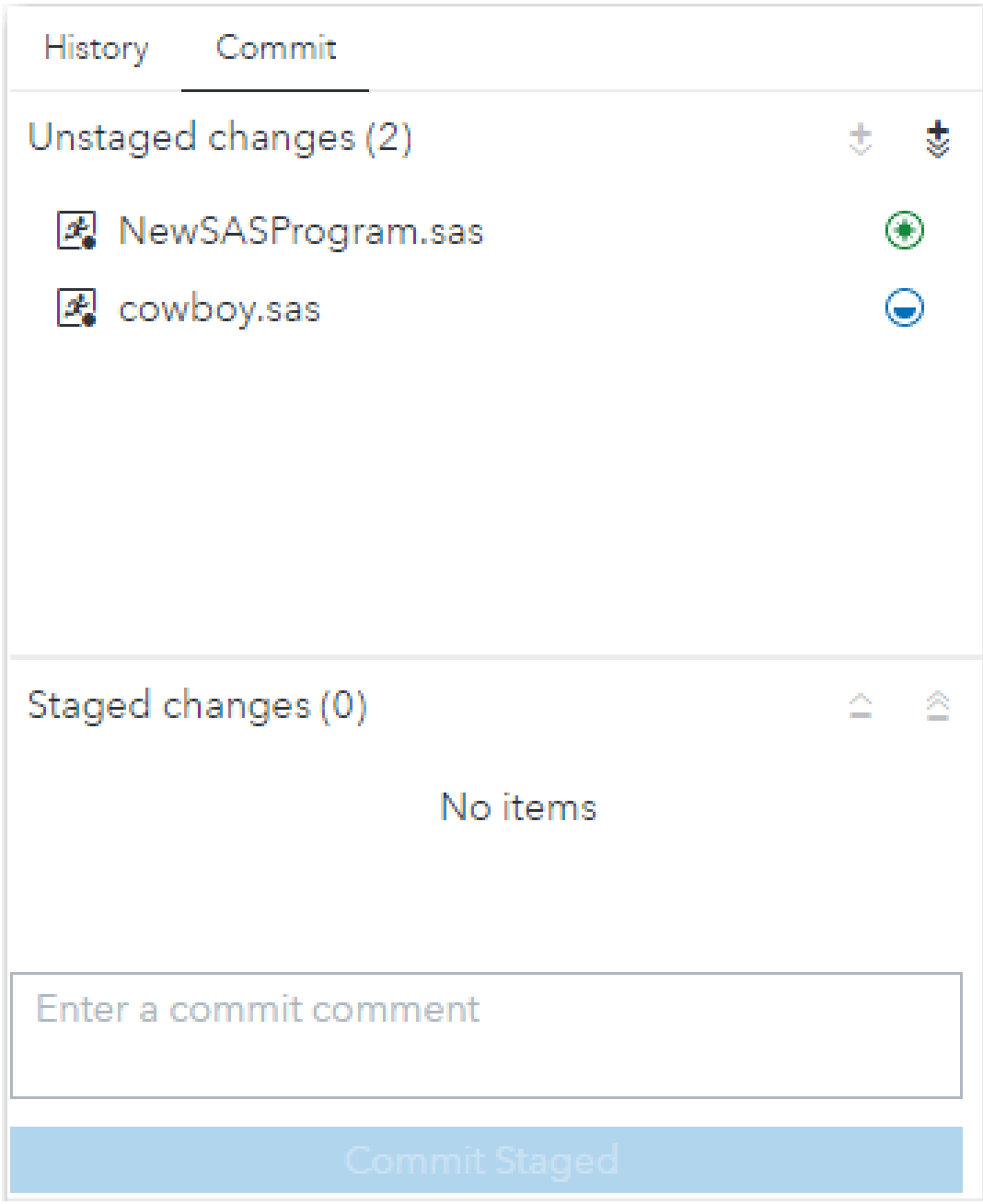
To commit changes to your local repository:

1. After you have saved your changes to the files in your working directory, make sure your local Git repository tab is open in the work area. Click the **Commit** subtab.

2. If your changed files are not listed in the Unstaged changes area, click to refresh the list.

   Files in the staging area are identified by a status icon that indicates the type of change:

   - – a new file to be added to the local repository.
   - – a renamed file.
   - – an existing file with changes.
   - – a deleted file.
   - – a file with content that is very similar or identical to another file in the local repository.
   - – a file with a merge conflict.

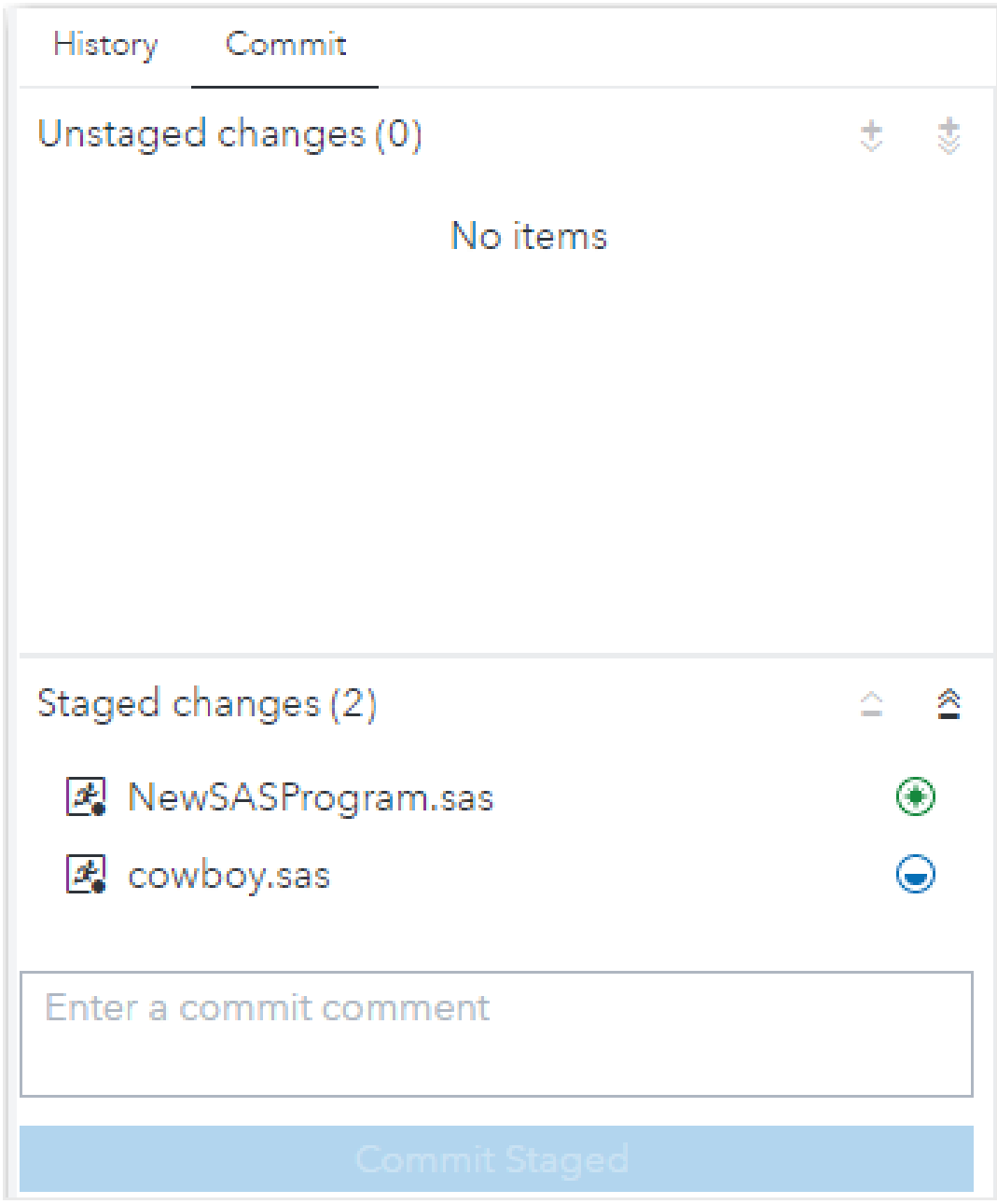


Note: Autosaved files that include ~ in the file name are not included in the list of unstaged files.

3 Select one or more changes that you want to commit and click . When you select files to commit, it is a good practice to group changes that are related and belong together in a single commit.

TIP You can also stage and unstage files by double-clicking them.

Note: Changes that you do not stage are not included in the commit. Your changes remain in your working directory and in the Unstaged changes area until you commit them.



Note: To stage all changed files at one time, click .

> **TIP** You can view the differences between the versions of a SAS program file in the staging area and your local repository by clicking the file on the **Commit** tab. File differences are not highlighted for task, query, and flow files, so you should consider documenting changes in the commit comment.
>
> To discard the changes to a file in the staging area and reset the file to the version in your local repository, right-click the file and select **Reset**. If the file that you have reset is open in the work area, you are prompted to either reload the version from your local repository or retain the edited version.

4 In the comments box, enter a description of the changes in the files that you are committing, and then click **Commit Staged**. The changed files are now available to be pushed to the remote repository.

Pulling and Fetching Files

You can download data from the remote repository to your local repository in two ways: pulling and fetching. The pull feature enables you to download the files from the remote repository and update your local repository to make sure you have the latest content. If you have made changes to some of the files in your local repository, Git merges the content from the remote repository with your local files. To pull files from the remote repository, make sure the tab for your local Git repository is open in the work area, and then select **Pull *branch-name*** ⇨ **Pull**.

Note: SAS Studio displays a message if there is a conflict between files in the remote and local repositories when you pull the remote files. If a conflict occurs, the conflicted files are moved to the Unstaged changes area. After you resolve the conflicts, you can commit the files again. Files that are pulled from the remote repository and do not have any conflicts are staged for the commit.

The fetch feature downloads the files from the remote repository, but fetching does not merge any changes into your current working branch. To fetch files from the remote repository, make sure the tab for your local Git repository is open in the work area. Next, select **Pull *branch-name*** ⇨ **Fetch** and specify whether you want to fetch changes from a specific branch or from all branches.

Pushing Files

After you have staged and committed your changed files from your working directory to your local repository, you can push them to the remote repository. When you push your committed changes to the remote repository, you are synchronizing files in your local repository with the remote repository.

To push your files to the remote repository, make sure the tab for your local Git repository is open in the work area. Click **Push *branch-name*** on the toolbar. The commit history shows your committed changes merged into the remote repository.

Resetting Your Local Repository

You can undo commits that you have made by resetting your local repository to a prior commit.

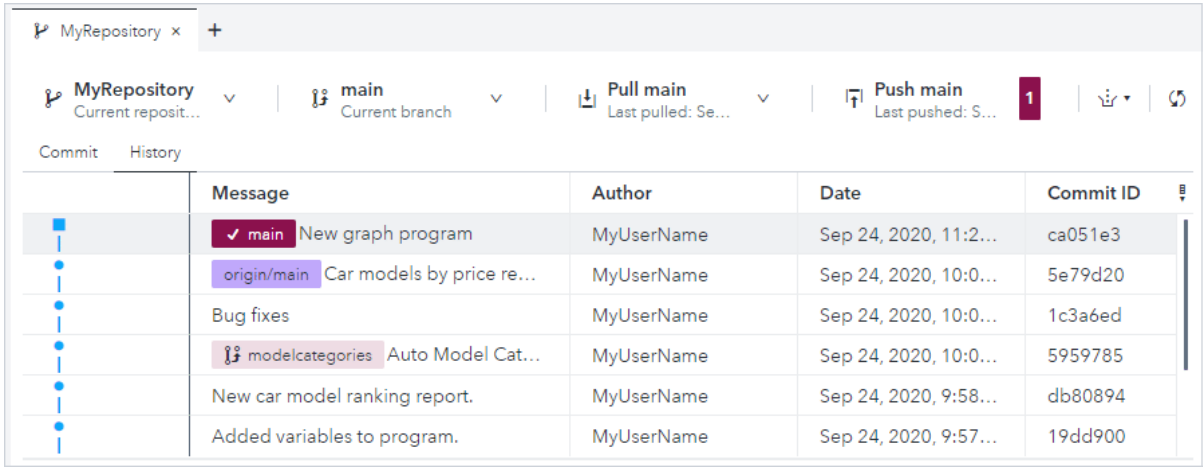
To reset your local repository:

1. Make sure the commit history is open for your local repository. Right-click the commit that you want to reset and select **Reset**. The Reset Local Repository window appears.
2. Select the type of reset that you want to perform:
   - **Soft** – resets the current branch to the prior commit. No changes are made to the staged changes (the index) or your working directory.
   - **Mixed** – resets the current branch to the prior commit and resets your staged changes (the index). No changes are made to your working directory.
   - **Hard** – resets the current branch to the prior commit, resets your staged changes (the index), and discards all changes in your working directory. You cannot undo a hard reset.
3. Click **OK** to reset your repository.

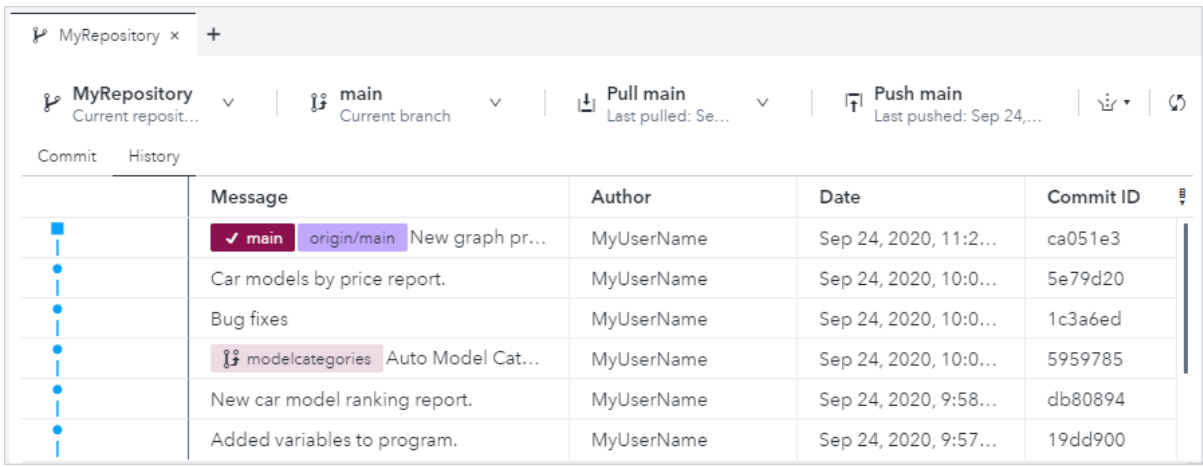
Working with Branches in Git

Your work in Git is managed in branches. The default branch in your remote repository is identified as the origin/main branch. The default branch in your local repository is identified as the main branch. You can create additional branches and merge branches. When you push your committed changes to the remote repository, you are synchronizing files in a branch in your local repository with the branch in the remote repository. You can use the History subtab to see the commits that are associated with a branch and whether branches are synchronized. For more information, see “Viewing the Commit History” on page 227.

In this example, the main branch is ahead of the origin/main commit by one commit. This lack of synchronization indicates that you have made changes to the local main branch that are not included in the remote origin/main branch because the "New graph program" commit has not yet been pushed to the remote repository. For more information, see “Pushing Files” on page 232.



After the “New graph program” commit has been pushed to the remote repository, the origin/main and main branches are synchronized and are listed on the same line in the commit history.



For more information about managing branches in Git, see the documentation for your repository hosting service.

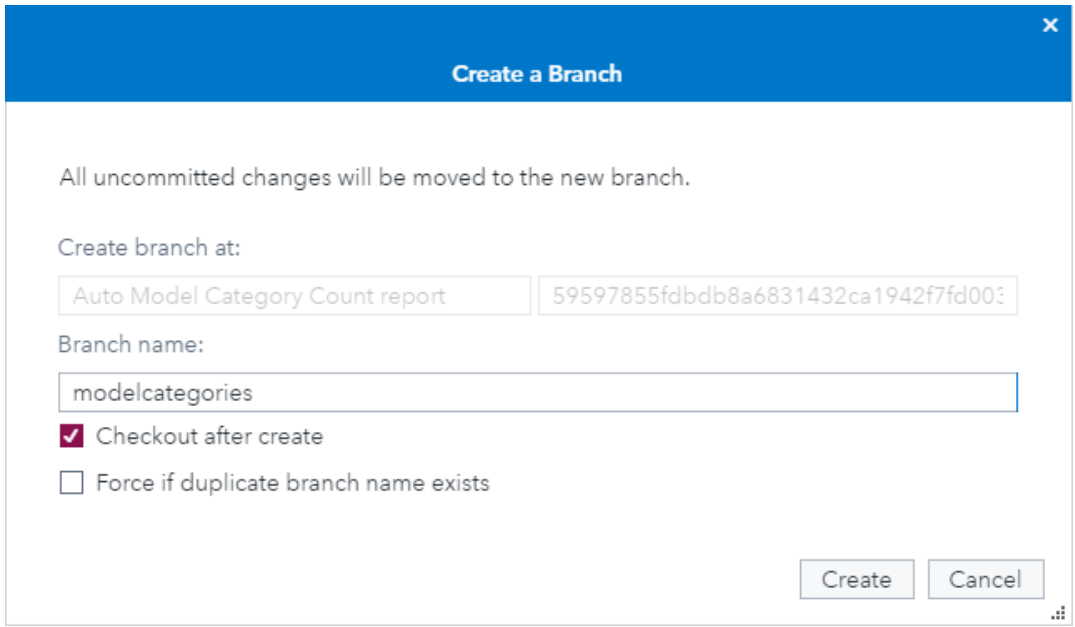
Creating a Branch

You can create a branch in Git in order to separate work that you are doing in the repository from other work that you are doing. When you create a branch, the commits in that branch are separated from your other commits until you merge the branch with another branch.

Note: When you have multiple branches, you can check out the branch that you want to work in by using the **Current branch** drop-down list at the top of your Git repository tab. You can also right-click the branch in the commit history and select **Checkout branch** ⇨ ***branch-name***. For more information, see “Checking Out Branches” on page 235.

To create a branch:

1. Make sure the History subtab is open for your local repository. Right-click the commit with which you want to start a new branch and select **Create new branch**. The Create a Branch window appears.
2. In the **Branch name** box, enter a name for your new branch. The branch name can contain only letters, numbers, and special characters such as hyphens or underscores.
3. If you want to create a branch but remain in the current branch, clear the **Checkout after create** check box. This option is selected by default.
4. If you want to discard any changes in your working directory that you have not committed when the branch is created, select the **Force if duplicate branch name exists** check box. This option is not selected by default.
5. Click **Create** to create the branch. Your local repository now contains the files from the commit that you selected for creating the branch.



Checking Out Branches

To open a branch in your working directory, you must check out the branch.

You can check out local branches by using the **Current branch** drop-down list at the top of the Git tab or by right-clicking the local branch in the commit history and selecting **Checkout branch** ⇨ ***branch-name***.

If you want to work with a remote branch that does not have a corresponding local branch, you can create a local copy of that branch in your working directory in one of the following ways:

- Create a local branch with the same name as the remote branch. For more information, see “Creating a Branch” on page 234.
- Right-click the remote branch in the commit history and select **Checkout branch** ⇨ ***branch-name***. A local version of the branch is created and checked out.

Note: If you check out a remote branch that already has a corresponding local branch, the remote branch is merged into the local branch.

Understanding Merging and Rebasing

Both merging and rebasing are used to integrate changes from one branch into another branch.

When you merge one branch into another, Git combines the changes from the commit sequences in each branch. A merge conflict occurs when Git encounters something in a file that has been changed in both branches. You must resolve any conflicts manually before the merge can be completed. After the merge is complete, you can still view the history of both branches. The existing branches are not changed in any way.

When you rebase a branch, you are moving an entire branch so that it begins on the tip of another branch. Rebasing enables you to incorporate all of the commits in one branch into another branch. You must resolve any conflicts manually before the rebase can be completed. Unlike merging, rebasing rewrites the project history and creates a linear project history. You can no longer view the original branch histories separately.

TIP You should never rebase a public branch, such as main, onto a new branch because this action can create problems for other users of the public branch.

Merging Branches

When you have completed work in a branch, you can merge the work in that branch into another branch.

To merge a branch:

1. Make sure the commit history is open for your local repository. In the **Current branch** drop-down list, select the branch into which you want to merge the new branch. For example, to merge *newbranch* with the main branch, you would need to check out the main branch.

2 Right-click the branch that you are merging, and select **Merge into *current-branch-name* ⇨ *new-branch-name***. For example, to merge *newbranch* with the main branch, you would right-click *newbranch* and select **Merge into main ⇨ *newbranch***. The current branch in the commit history list is updated with the latest commit from the new branch.

Note: If you are merging a branch into a branch that is not checked out, right-click the latest commit in the current branch, and select **Merge with branch ⇨ *branch-name***.

Note: SAS Studio displays a message if there is a conflict between files when you merge the branches. If a conflict occurs, the conflicted files are moved to the Unstaged changes area on the Commit subtab. After you resolve the conflicts, you can complete the merge by staging and committing the files.

Rebasing the Current Branch

You can use the rebase feature to move a branch to a new base commit. When you rebase a branch, the branch appears to have been created from a different commit. Rebasing is a powerful feature that enables you to change the history of your repository.

To rebase a branch:

1 Make sure the commit history is open for your local repository. From the **Current branch** drop-down list, select the branch that you want to rebase. For example, to rebase *newbranch* onto the main branch, you would need to check out the *newbranch* branch.

2 Right-click the branch that you want to use as the new base commit, and select **Rebase *branch-name* on ⇨ Selected commit**.

Note: SAS Studio displays a message if there is a conflict between files when you rebase the branches. If a conflict occurs, the conflicted files are moved to the Unstaged changes area on the Commit subtab. After you resolve the conflicts, stage and commit the files and continue the rebase.

Stashing Changes

You can use the stash feature to save any uncommitted changes (both staged and unstaged) that you are not yet ready to push to the remote repository. When you

stash changes that you have made to one or more files, the changes are removed from the working copy of the files and saved until you are ready to reapply them.

When you reapply the changes to your files, you can choose to delete or save the changes in your stash.

Note: When you stash changes, all updated and uncommitted files in the repository are affected.

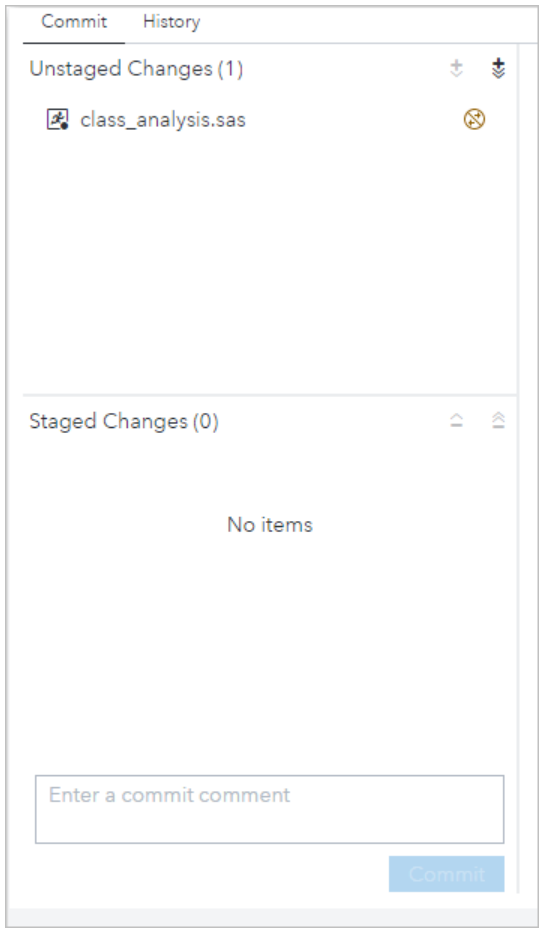
To use the stash feature, make sure the tab for your local Git repository is open in the work area. Click on the toolbar and choose from the following options:

- **Stash** – saves all uncommitted changes to the files in your local repository and removes the changes from the local files.
- **Pop stash** – reapplies all saved changes to the files in your local repository and deletes the changes from your stash.
- **Delete stash** – deletes the saved changes in your stash. When you delete the changes in your stash, you cannot reapply them to your local files.
- **Apply stash** – reapplies all saved changes to the files in your local repository and saves the changes in your stash.

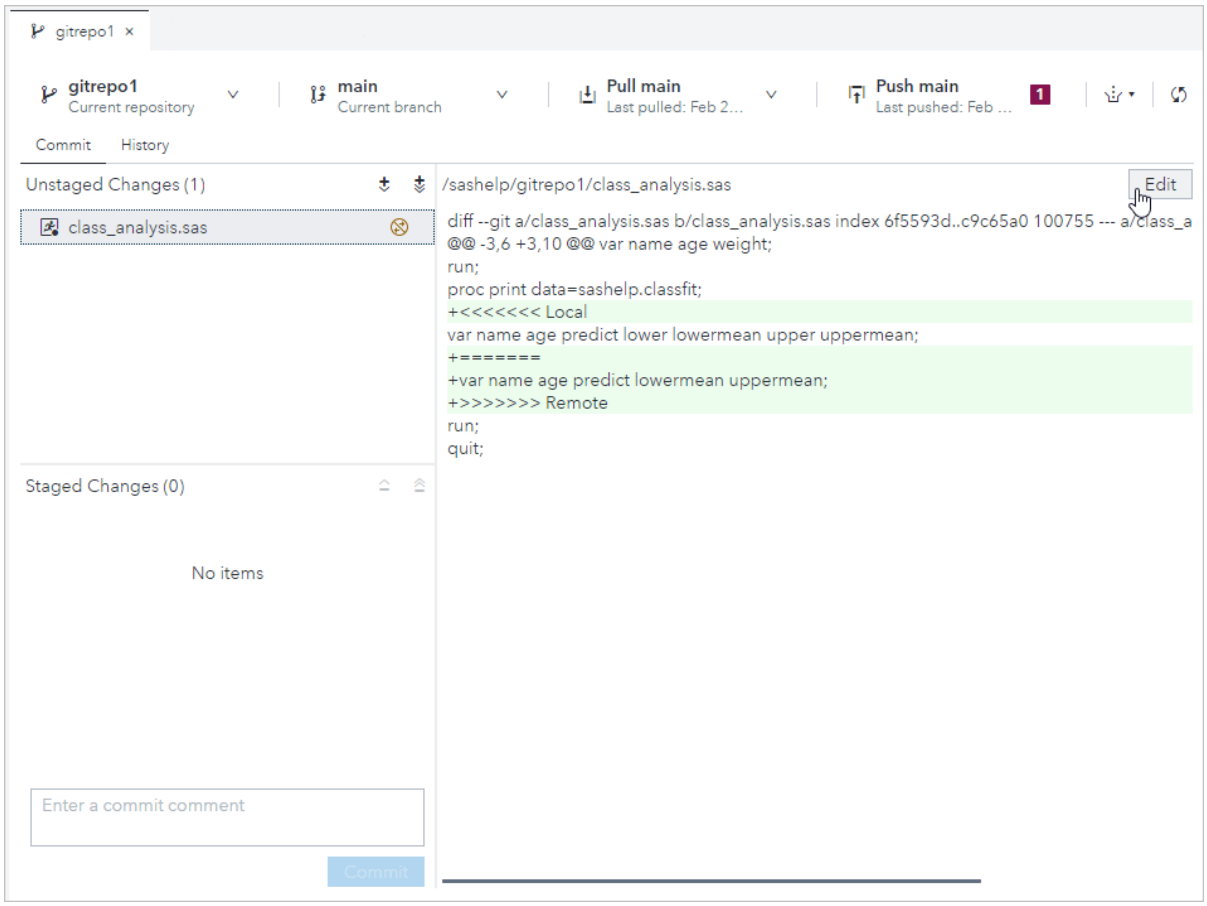
Resolving Merge Conflicts

Sometimes when you are working in a shared repository, you can encounter an error because you and a coworker have made changes to the same lines in a file. SAS Studio identifies the conflict in the file when you attempt to merge your changes from your local branch. In SAS Studio, merging occurs when you pull files from a remote repository, merge work from one branch into another branch, and use the Pop stash and Apply stash options.

For example, an error can occur when a coworker pushes a change to a file and you attempt to push a change to the same file without first pulling your coworker's updates to the file. When a file has a conflict and you attempt to pull from the remote repository or perform another merge operation, SAS Studio displays an error message and provides information to help you resolve the conflict. SAS Studio automatically documents the conflicts by adding information to the file to identify the conflicts that exist between the local and remote versions of the file. The file is moved back to the Unstaged Changes area of the Commit tab so that you can resolve the conflict.



To view the conflicts in the file, click the file in the Unstaged Changes area. The file is displayed in the area to the right of the Commit tab. View the documentation about the conflict that SAS Studio inserted in the file: the Local section contains the changes from your local commit, and the Remote section contains the conflicting lines from the commits that you pulled from the remote repository.



To resolve the conflicts in the file, click **Edit**. The file opens in a new tab in the work area. Edit the file as needed (considering both sets of changes), and then save the file. Click the Git repository tab, and then stage and commit the updated file.

Deleting a Repository

You can delete the local repository on your server file system. You can choose whether you want to delete all the files in the working directory when you delete the repository.

To delete a local repository:

1. Select **Options** ⇨ **Manage Git Connections**. Click **Repositories** and select the repository that you want to delete. Click 🗑.
2. Select the type of delete that you want to perform:
   - **Soft** – deletes only the Git repository definition.

     Note: After a soft delete, you can still view the folder using the **Explorer** section of the navigation pane. If you want to use the folder as a local Git repository again, you must right-click the folder and select **Git initialize**.

- **Mixed** – deletes the Git repository definition and removes any Git configuration information from the local repository directory. The files in your working directory are not deleted.

  Note: After a mixed delete, you can still view the folder using the **Explorer** section of the navigation pane. If you want to use the folder as a local Git repository again, you must right-click the folder and select **Git initialize**.

- **Hard** – deletes the Git repository definition, the local repository directory, and all files in the local working directory.

3 Click **Delete** to delete the repository.

Note: You can also delete a repository from the **Explorer** section of the navigation pane. Right-click the working directory and select **Delete**. Click **Delete** in the confirmation window, and then select the type of delete that you want to perform.

Sample Git Workflow Scenario

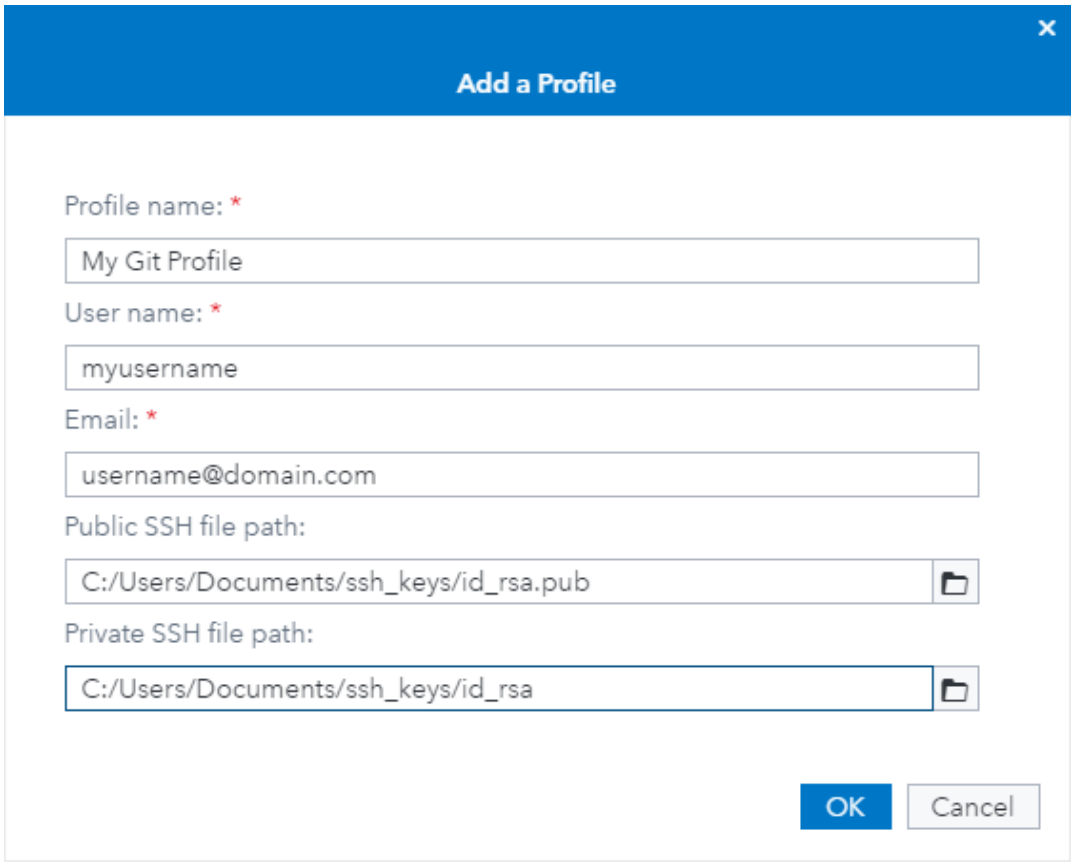
Suppose you need to help out on a project that stores its SAS programs in a Git repository. You need to make some minor changes to one program and add a new program to the project.

In this sample Git scenario, you clone a repository to your server file system, commit and push some changes, and create a branch. Then you merge the new branch with the main branch and push the changes.

1 Create a profile so that you can access Git repositories in SAS Studio.

   a Select **Options** ⇨ **Manage Git Connections**. Make sure **Profiles** is selected, and then click + to open the Add a Profile window.

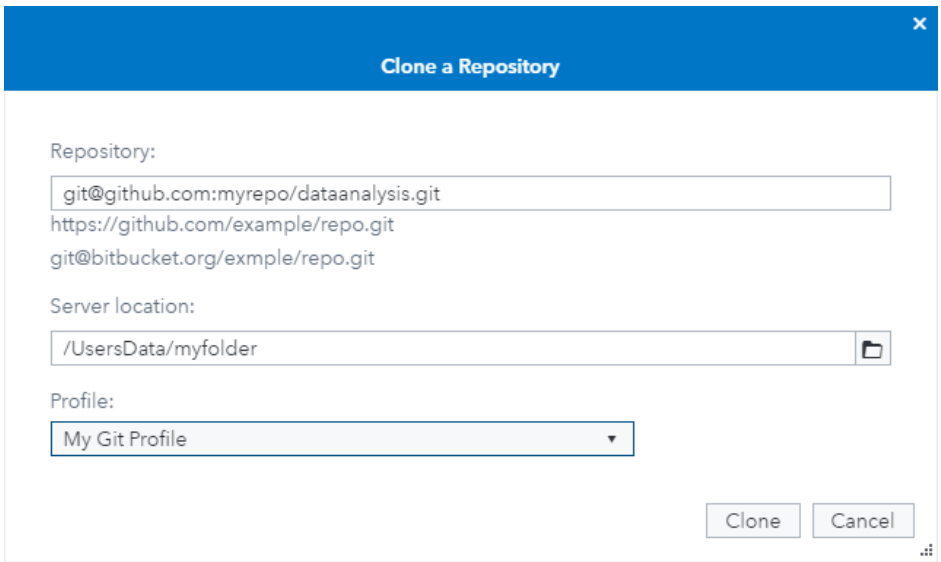
   b Enter the credentials necessary to access your Git repository and click **OK**.



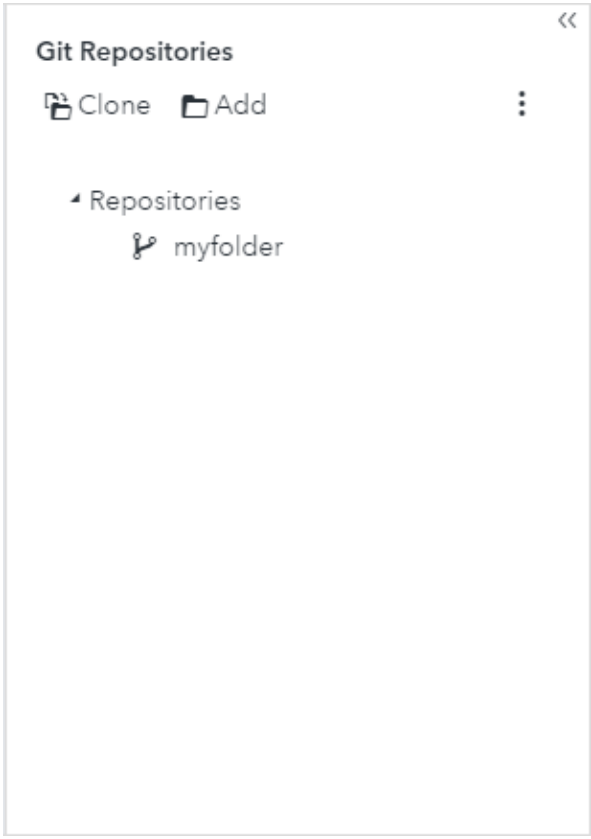
2 Clone your team's repository to your server file system so that you can access the files in the project.

   a In the **Git Repositories** section of the navigation pane, click **Clone a Repository**.

   b In the Clone a Repository window, specify the remote repository and the location on your server file system that you want to use.



   c Click **Clone** to create the repository. The repository is added to the **Git Repositories** section of the navigation pane.



3 Open the program that you need to update from your local repository, make your changes, and save the program.

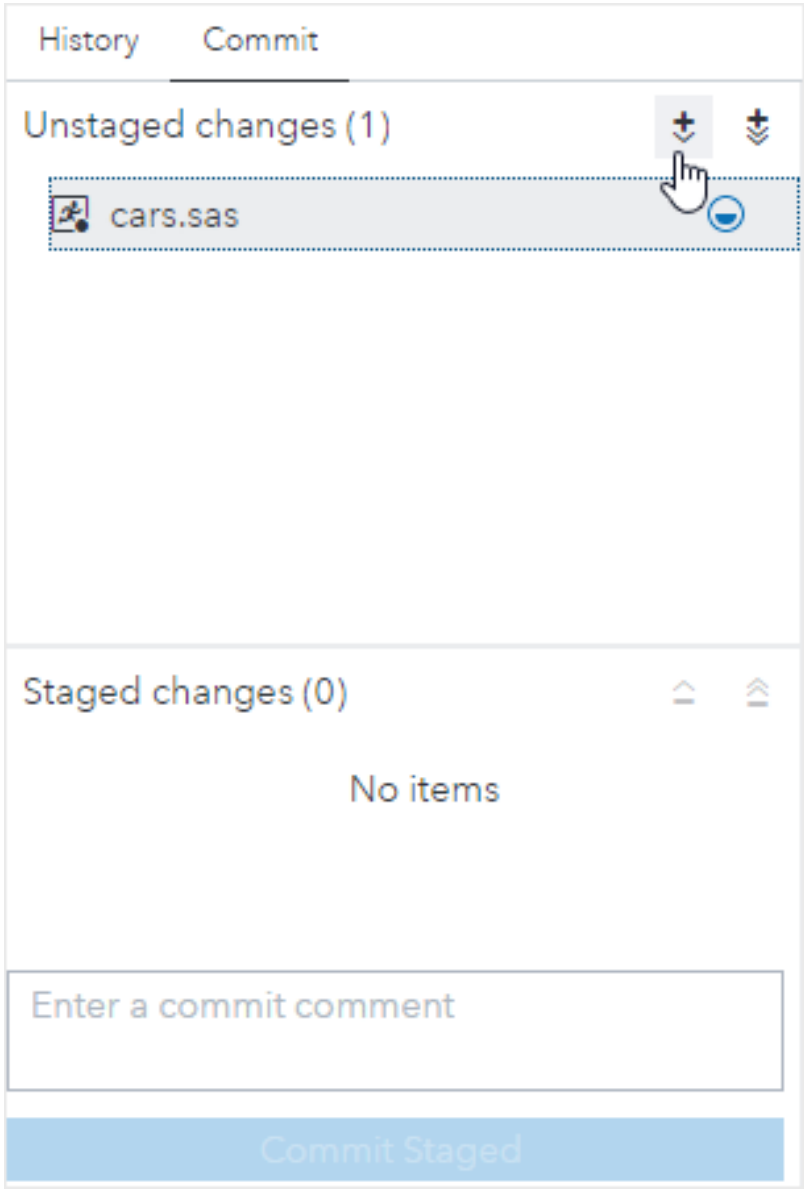
   a Click in the navigation pane to open the **Explorer** section and navigate to your working directory.

   b Double-click the SAS program that you want to edit and make your changes. Click to save your changes.

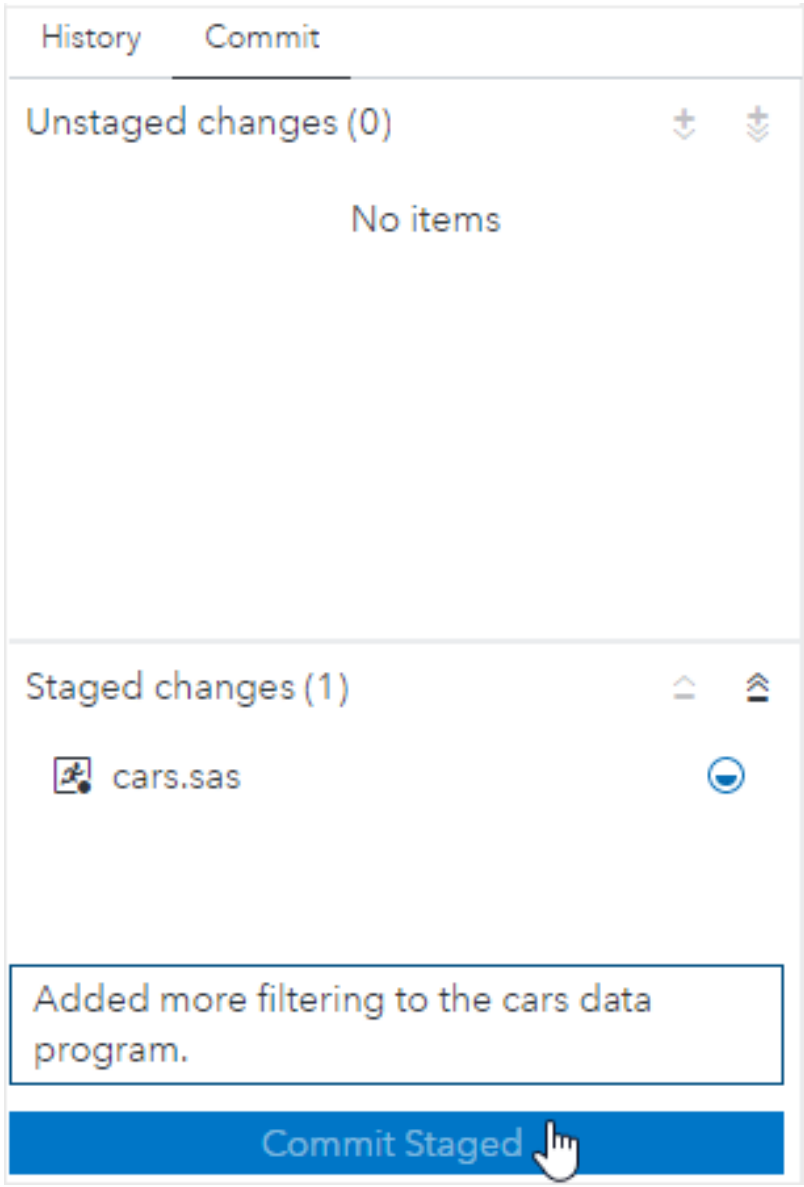
4 Stage your changed files, enter a commit message, and commit your files to your local repository.

   a Click in the navigation pane, and then double-click the repository to open it in the work area. Make sure the **Commit** tab is selected. If your changed file is not listed in the Unstaged changes area, click to refresh the list.

   b Select the updated file and click to move the file to the Staged changes area.



c Enter your commit comments in the commit box and click **Commit Staged**.

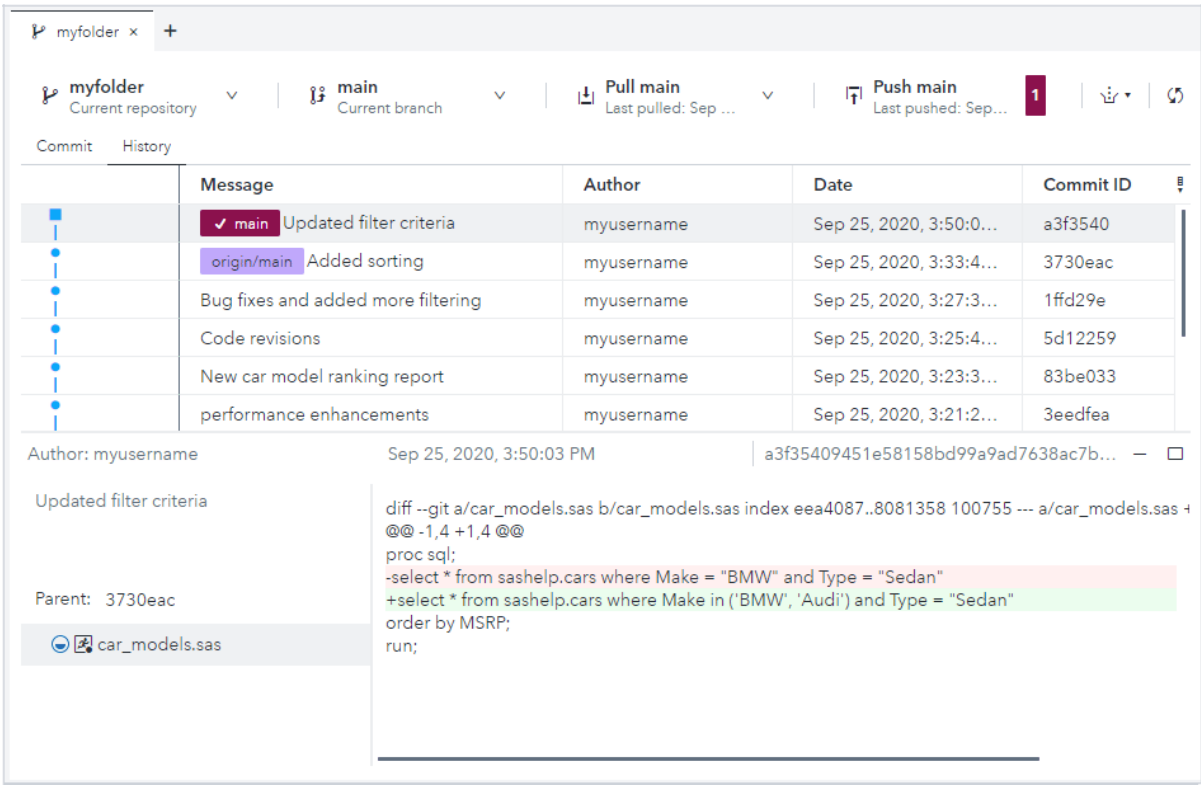


Your changed file is committed to the local repository.

d View your commit history by clicking **History** on your Git repository tab. Notice that your commit is listed at the top of the commit history list. Your commit currently exists only in the main branch of your local repository. The main branch of your local repository is identified by ✓ main.

The second commit in the list represents the state of the main branch on the remote repository. The main branch on the remote repository is identified by origin/main.

When you push your changes to the remote repository, your commit is then part of the origin/main branch on the remote repository.

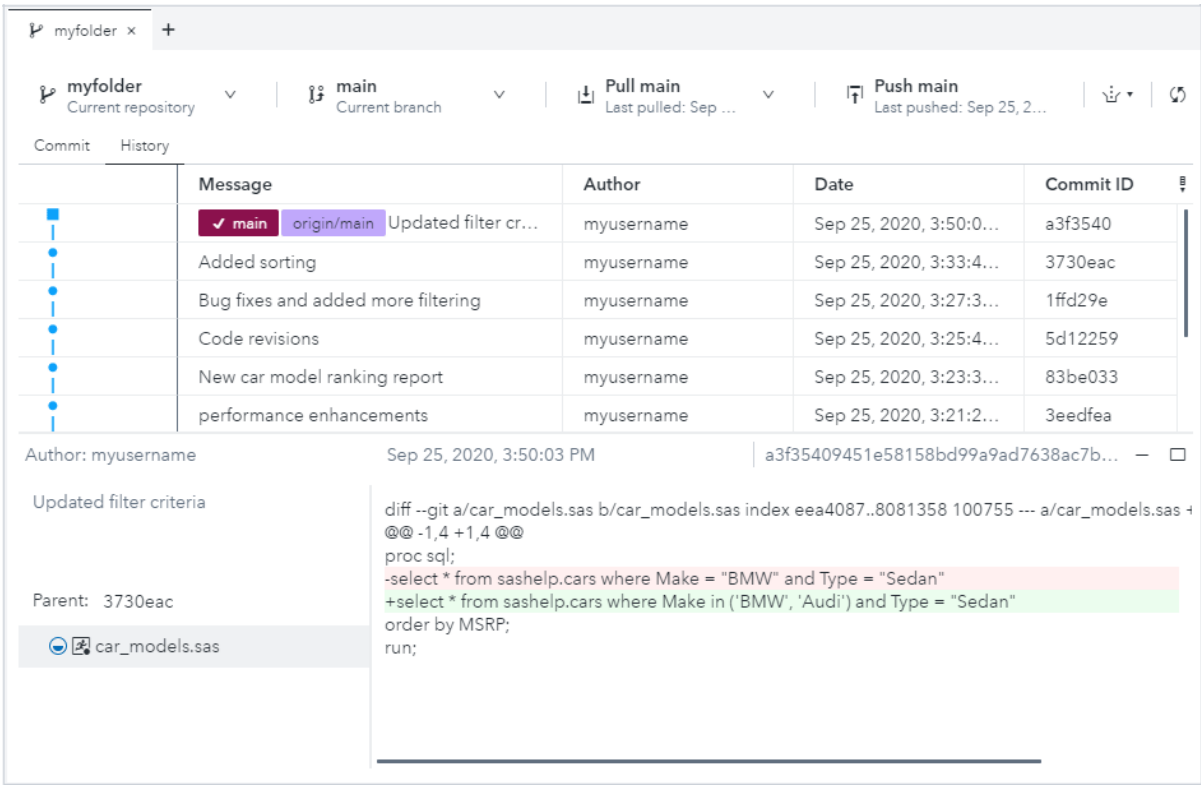


5 Pull the files from the remote repository and push your committed files.

> **TIP** It is good practice to pull the files from the remote repository before you push to make sure you do not have any conflicts.

a On your Git repository tab, select **Pull *branch-name*** ⇨ **Pull**. Your local repository is updated with the latest content from the remote repository. Any changes in the remote repository are merged with your local files. If there is a conflict between files in the remote and local repositories, SAS Studio displays a message so that you can resolve the conflict before you proceed.

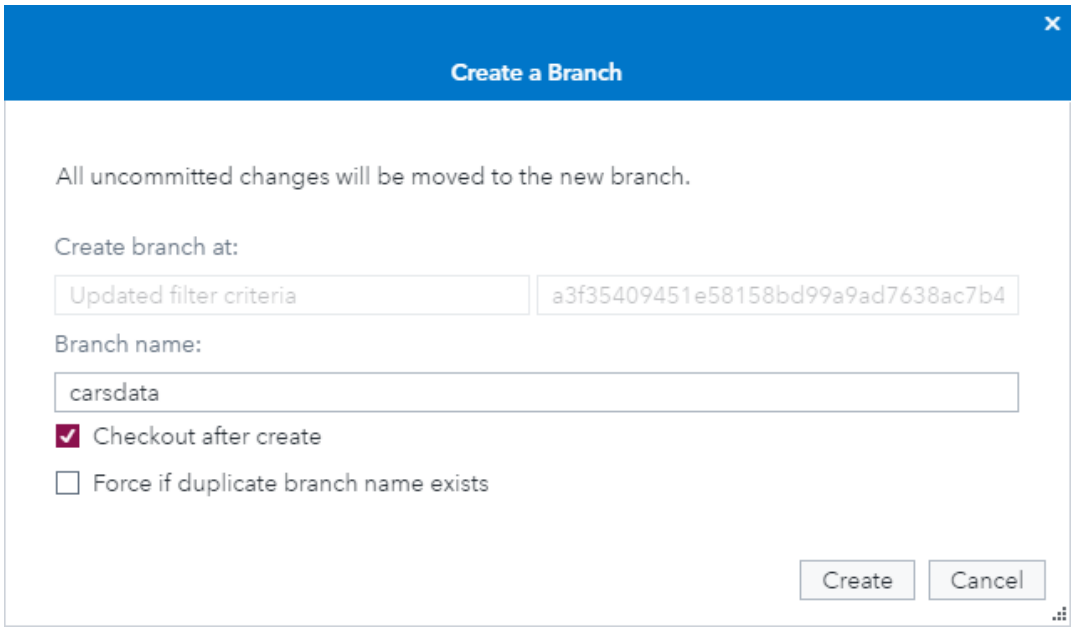
b Click **Push *branch-name*** to push your changed file to the remote repository. Your commit history is updated. Notice that your commit is now part of the remote origin/main branch.



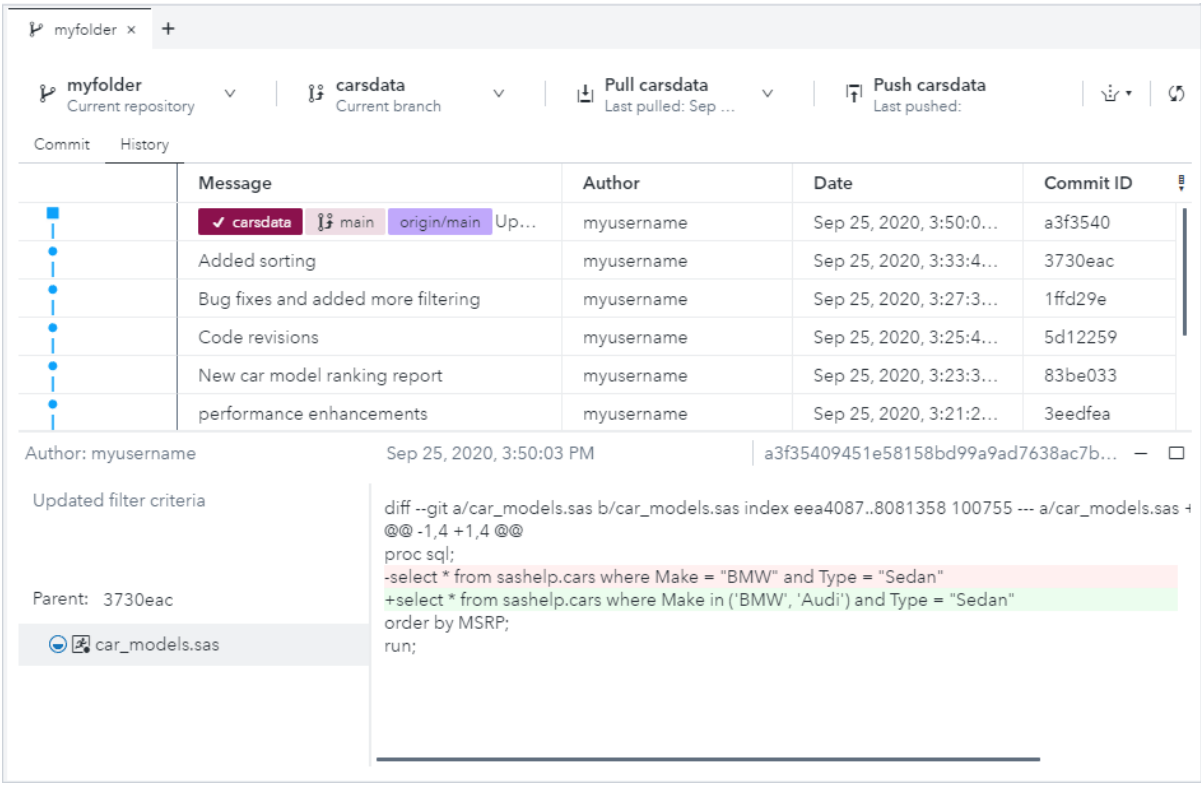
6 Create a branch to keep your work separate from the main branch. Add a new program, test it, and commit your changes to that branch.

   a In the commit history list, right-click your commit and select **Create new branch**. The Create a Branch window appears.

   b Enter a name for the branch and click **Create**. Notice that the **Checkout after create** option is selected so that the new branch is checked out and is the active branch that you are working in.

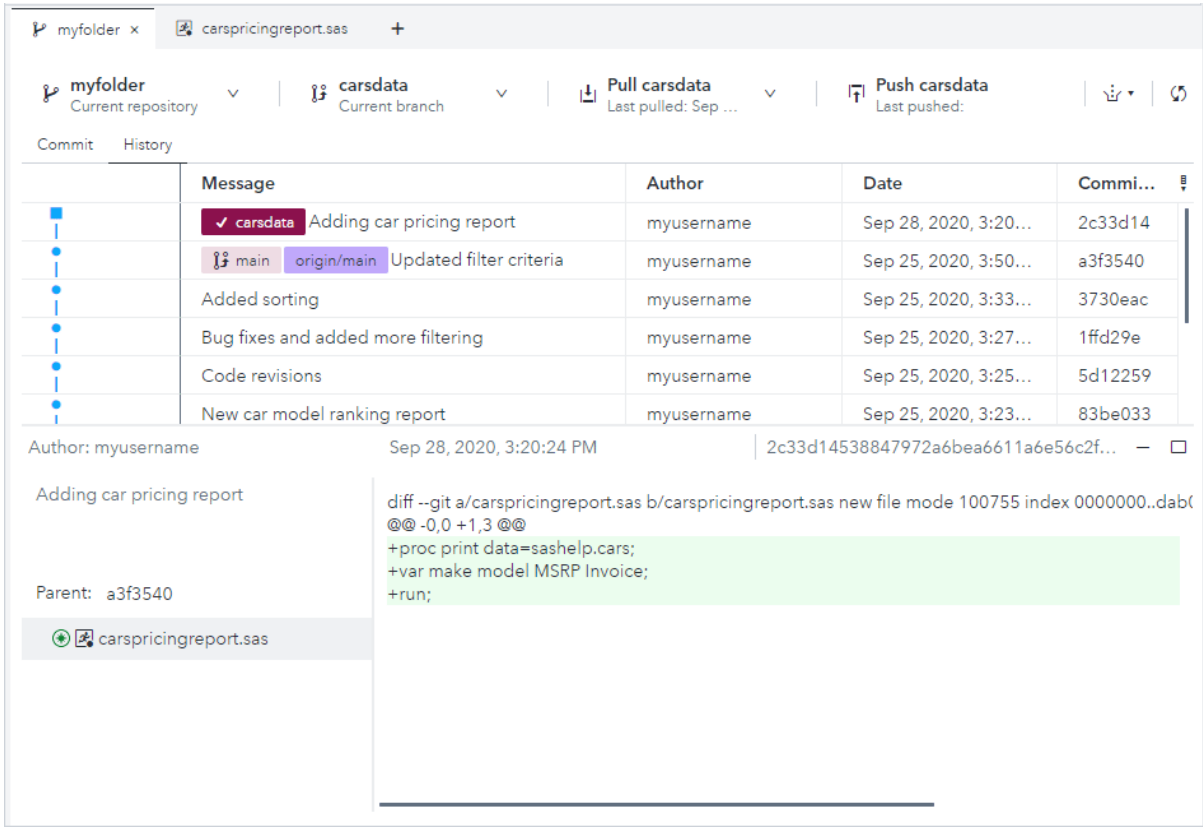


In the commit history list, notice that the new carsdata branch, the remote origin/main branch, and your local repository main branch are all synchronized and displayed on the same line.



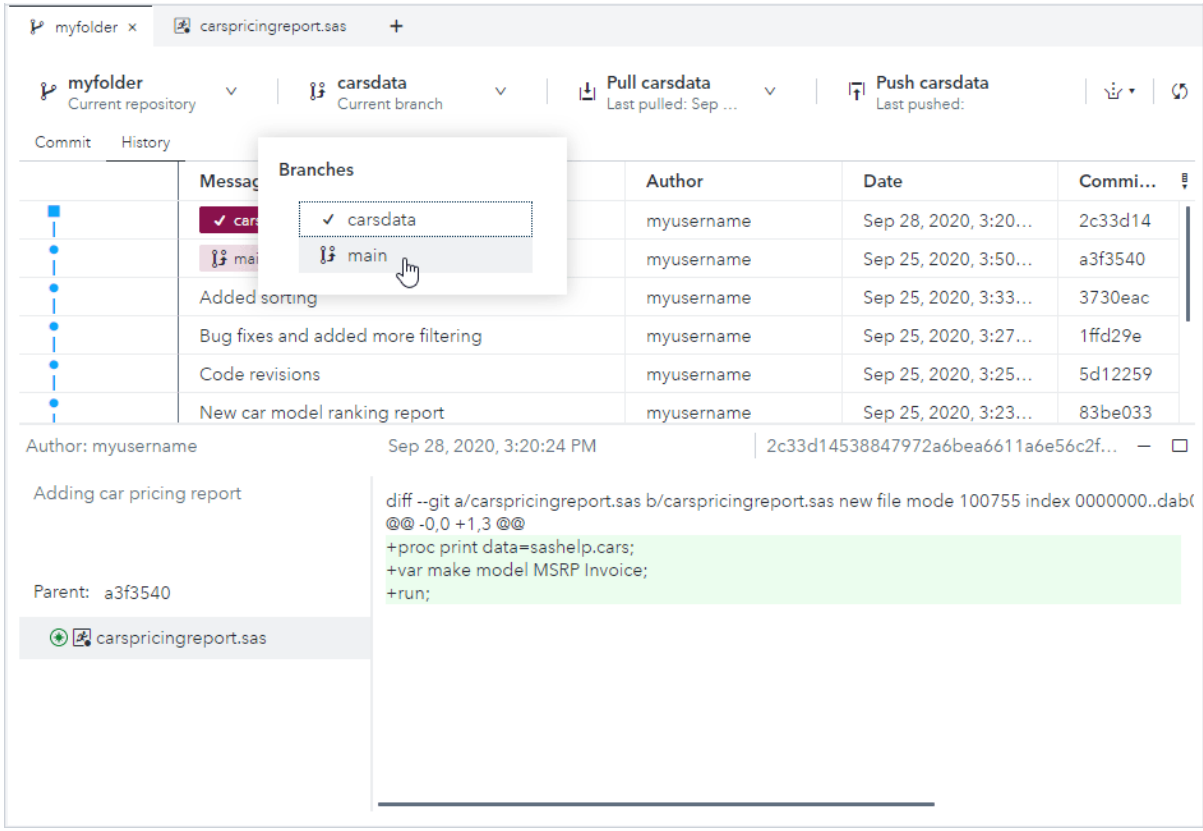
c Use the **Explorer** section of the navigation pane to create a program and save it in your working directory.

d Stage the new program and commit it to the new carsdata branch in your local repository. Notice in the commit history list that your commit in the carsdata branch is not synchronized with the remote origin/main branch or the local main branch.

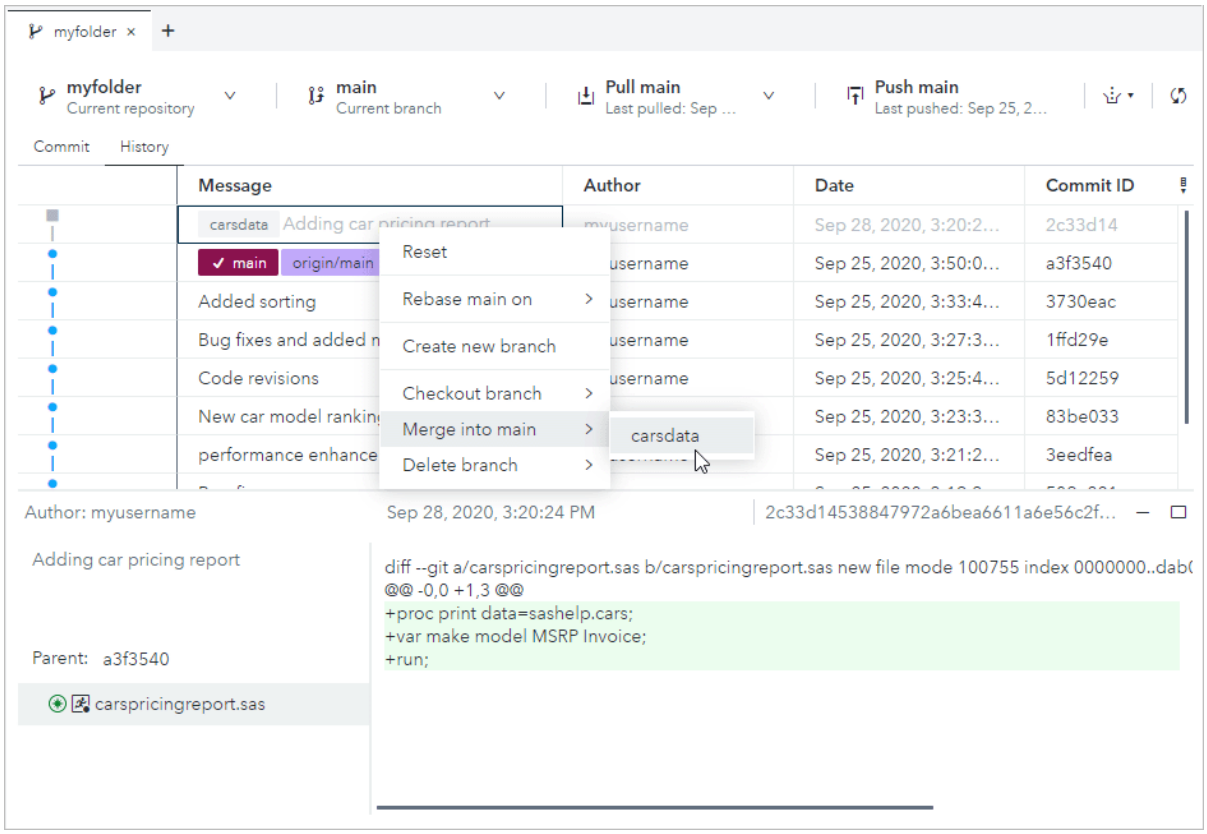


7 When your new program is complete, merge the new branch back into the main branch and perform a final pull from and push to the remote repository.

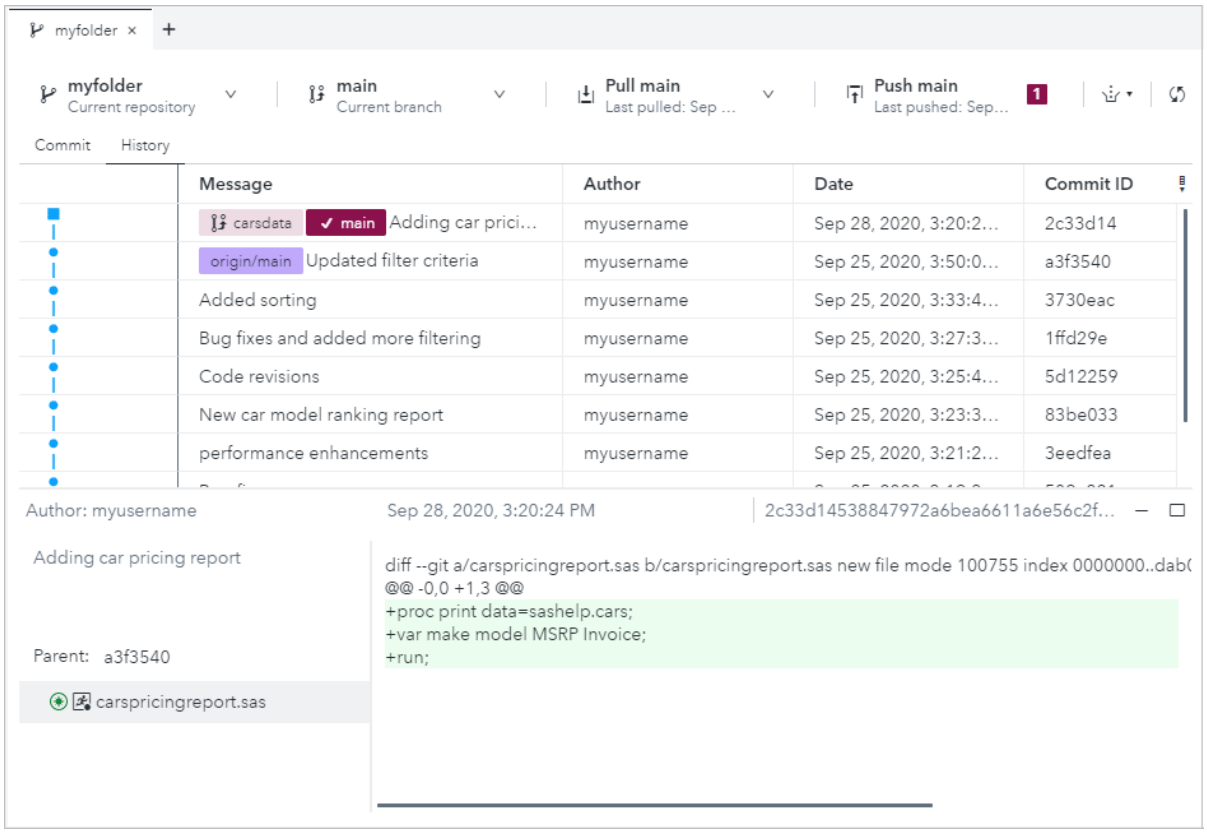
   a When you are ready to merge the carsdata branch into the main branch, you must first check out the main branch. Select the main branch from the Current branch drop-down list. In the confirmation window, click **Reload** to make sure that all of your changes are saved.



In the commit history list, right-click the carsdata branch and select **Merge into main ⇨ carsdata**.

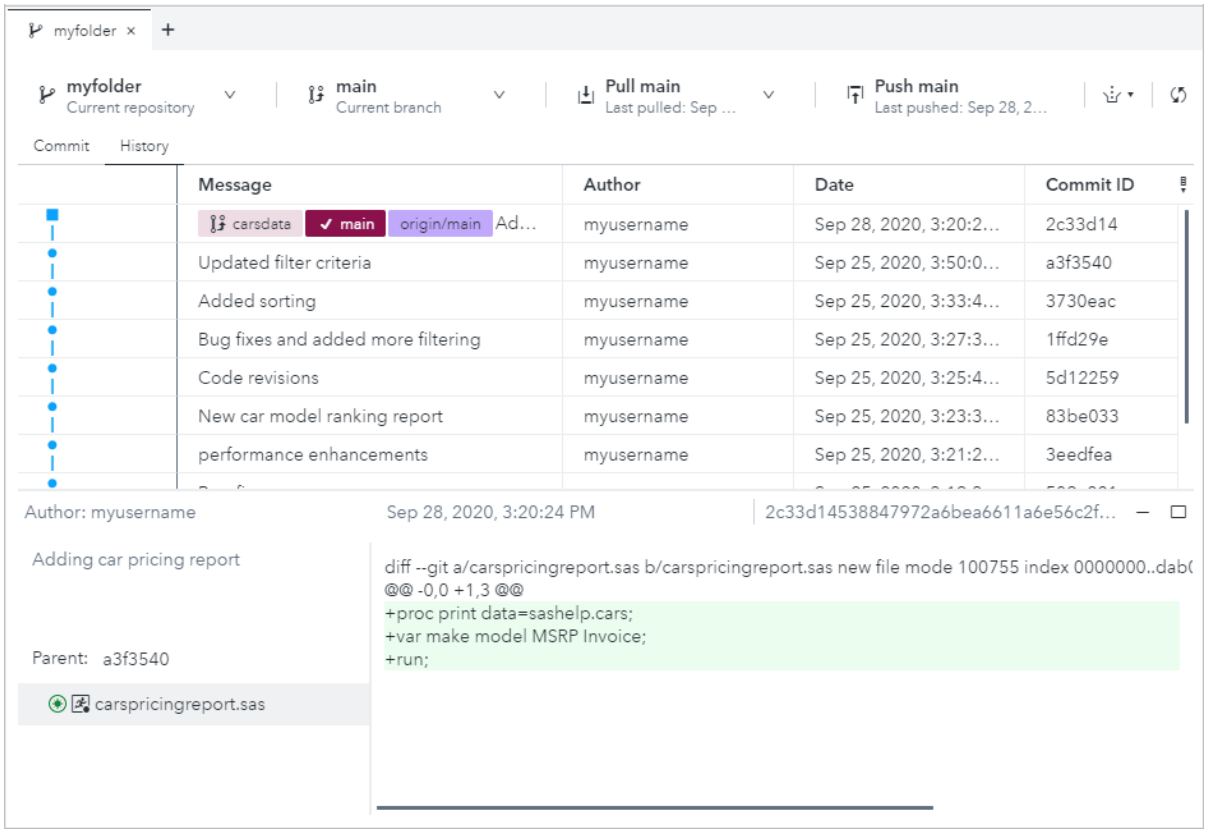


Notice in the commit history list that the main branch has been updated with the new file from the carsdata branch.



b On the **Git** tab, select **Pull main** ⇨ **Pull**.

c Click **Push** to push your changes to the remote repository. Your commit history is updated. Notice that the local main branch, which was merged with the carsdata branch, and the remote origin/main branch are now synchronized.


myfolder
myfolder
Current repository
main
Current branch
Pull main
Last pulled: Sep ...
Push main
Last pushed: Sep 28, 2...
Commit
History
Message
Author
Date
Commit ID
carsdata
main
origin/main
Ad...
myusername
Sep 28, 2020, 3:20:2...
2c33d14
Updated filter criteria
myusername
Sep 25, 2020, 3:50:0...
a3f3540
Added sorting
myusername
Sep 25, 2020, 3:33:4...
3730eac
Bug fixes and added more filtering
myusername
Sep 25, 2020, 3:27:3...
1ffd29e
Code revisions
myusername
Sep 25, 2020, 3:25:4...
5d12259
New car model ranking report
myusername
Sep 25, 2020, 3:23:3...
83be033
performance enhancements
myusername
Sep 25, 2020, 3:21:2...
3eedfea
Author: myusername
Sep 28, 2020, 3:20:24 PM
2c33d14538847972a6bea6611a6e56c2f...
Adding car pricing report
Parent: a3f3540
carspricingreport.sas
diff --git a/carspricingreport.sas b/carspricingreport.sas new file mode 100755 index 0000000..dab(
@@ -0,0 +1,3 @@
+proc print data=sashelp.cars;
+var make model MSRP Invoice;
+run;

9

Understanding Tasks in SAS Studio

What Is a Task?

A task is an XML and Apache Velocity code file that generates SAS code and formats results for you. Tasks include SAS procedures from simple data listings to complex analytical procedures. SAS Studio is shipped with several predefined tasks that are available in the Standard perspective. The tasks are organized into categories. Some categories and their tasks might not be available at your site because you do not have the required SAS product.

For more information about each of these tasks, see *SAS Studio: Task Reference Guide*.

How to Run a Task

To run a predefined task:

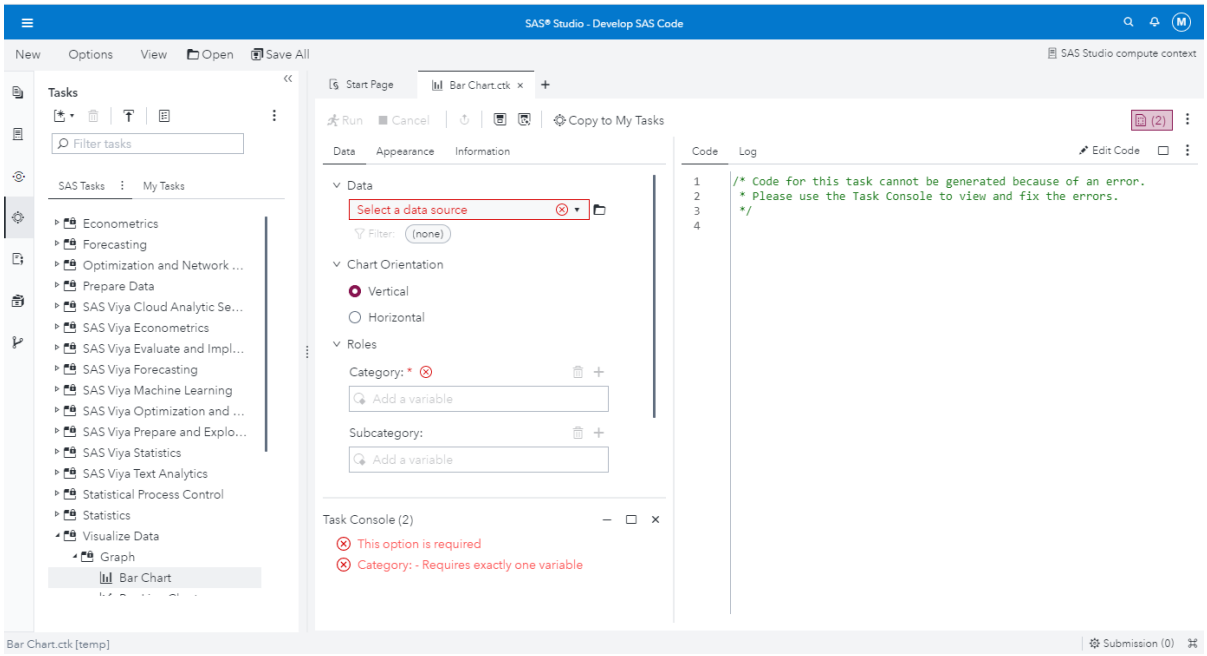
1 In the navigation pane, click to open the **Tasks** pane.

2 Expand the folder that contains the task.

Note: You can hide one or more tasks and task folders by right-clicking the selected tasks or task folders in the navigation pane and selecting **Hide**. Hidden tasks and task folders are displayed with an updated icon in the navigation pane and cannot be run. To show hidden tasks and task folders, right-click the tasks or task folders and select **Show**.

3 Right-click the task name and select **Open task**. Alternatively, you can double-click the task to open it.

The task opens to the right of the work area.



4 If the **Data** tab is available, specify an input data source and select columns for the roles in the data source. A role is a description of a variable's purpose in the task. To add a column to a role, click **+**. A list of available columns for that role appears. If multiple columns can be assigned, you can press Ctrl or Shift to select multiple columns from the list and click **OK**.

5 On the remaining tabs, specify any other required options, which are denoted with a red asterisk. As you assign values to the task, the relevant SAS code is generated. For more information about the options available for each task, see *SAS Studio: Task Reference Guide*.

> **TIP** The **Task Console** displays any information required to run the task.

6 To run the task, click **Run**.

Edit a Predefined Task

To customize the predefined tasks for your site, you can edit the XML code that is used to create the task.

To edit a predefined task:

1. In the navigation pane, click to open the **Tasks** pane.
2. Expand the folder that contains the task.
3. Right-click the name of the task that you want to edit and select **Copy as task template**. You can choose to save the task in My Tasks or the Explorer.
4. Specify a name and description for the task. By default, the name and description from the predefined task is used.

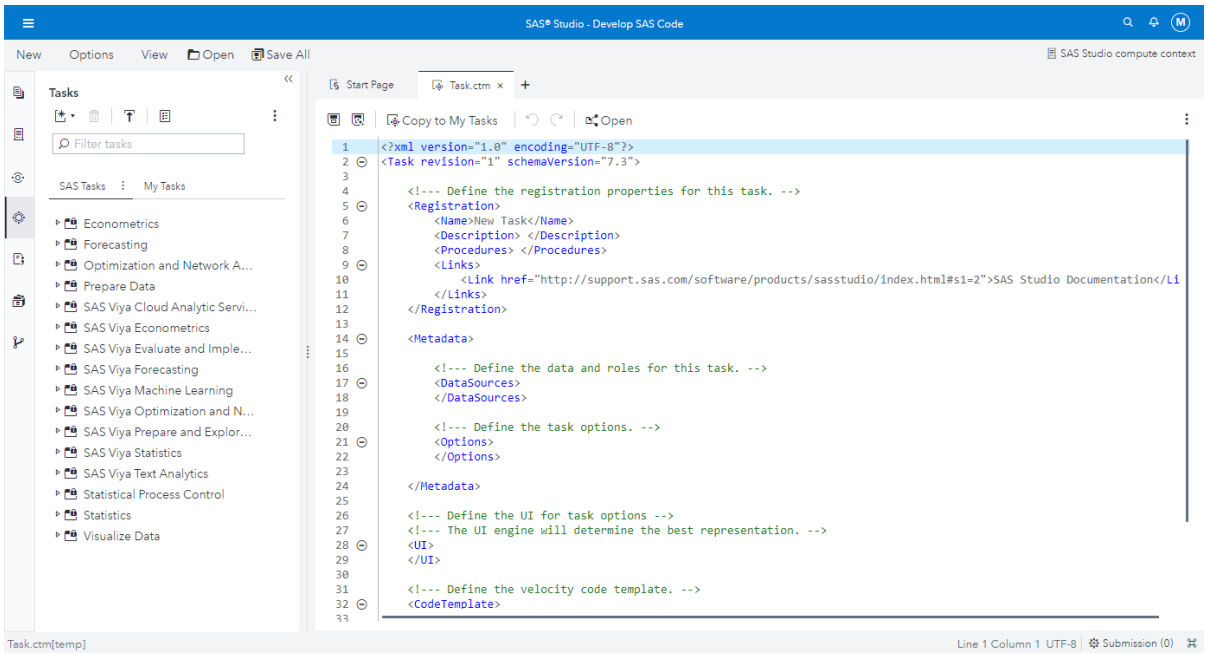
   Click **OK**.
5. Right-click the copied task and select **Edit task template**. The CTM file for the task appears in a new tab.
6. Edit the XML file and save your changes. To preview your changes, click **Open**.

Create a Custom Task

SAS Studio provides a template that you can use to create custom tasks for your site.

To create a custom task:

1. In the navigation pane, click to open the **Tasks** pane.
2. Click and select **Task**. A blank task template opens.



3 Edit the code in the task template to create your task. To view the user interface for the task template, click **Open**. For more information about this file, see *SAS Studio: Developer's Guide to Writing Custom Tasks*.

4 Click .

Specifying the Options for the Task Code

The Preferences window enables you to change several options that affect what and how the task code is displayed.

To access these options, select **Options** ⇨ **Preferences**. In the Preferences window, click **Tasks**.

For more information, see "Setting Task Preferences" on page 270.

Appendix 1

Customizing SAS Studio

About Customizing SAS Studio

You can customize SAS Studio by changing the global settings, such as theme and locale, and by specifying application preferences, such as code editor and table viewer options.

Creating Global Settings

About the Settings Window

The Settings window enables you to specify settings for the appearance theme, locale, and accessibility. To change the global settings, click your initial on the toolbar and select **Settings**.

Specifying the General Settings

From the **General** page, you can specify these options for SAS Studio.

Options	Description
Theme	Enables you to change the appearance of the SAS Studio web application. The theme specifies the collection of colors, graphics, and fonts that appear in the application. You can choose from SAS themes or custom themes, if you have any available. ■ **Use the default theme (*theme-name*)** — uses the default theme as specified by your system administrator. ■ **Choose a theme** — enables you to select a theme. You can choose from these SAS themes: High Contrast Presents a dark background with high-contrast foreground elements to meet the needs of users with low vision. Ignite Presents a dark user interface that enables graphs, visualizations, and other elements to stand out. Illuminate Includes a clean and uncomplicated color palette that is easy to use. This is the default theme. Inspire Consists of vibrant and cohesive colors that shift the emphasis from the application to the content. The theme change takes effect after you close the Settings window.
Reset to show all warning and information messages	Displays warnings and messages that you previously chose to hide.

Options	Description
Profile picture	Displays your profile picture.

Specifying the Region and Language Settings

From the **Region and Language** page, you can specify these options.

Options	Description
Locale for regional formats and sorting	Specifies the locale that is used for sorting data and formatting values such as dates, times, numbers, and currency. The default setting is the browser locale. Changes take effect after you sign out and sign back in.
Locale for offline processes	Specifies the locale that is used for offline jobs or background processes such as report distributions or notifications. The default setting is the locale of the Java Runtime Environment.

Specifying the Accessibility Settings

From the **Accessibility** page, you can specify these options.

Options	Description
Enable sounds	Enables audio indicators for events that occur within the user interface.
Enable visual effects	Adds visual effects that indicate state changes. For example, when this setting is enabled, you see a subtle movement in the user interface if you delete an item.
Adjust the display duration for pop-up notifications	Enables you to specify the length of time in seconds that pop-up notifications are displayed. The default value is 5 seconds.
Invert application colors	Displays SAS Studio in inverted colors.
Display tooltips when using the keyboard to navigate	Displays the tooltips for each item in the application in a pop-up window as you navigate with the keyboard. You can specify where in the browser window you want to display the tooltips.

Options	Description
Customize the focus indicator settings	Enables you to customize the attributes of the outline that indicates which user interface component is active.

Setting Your Preferences

About the Preferences Window

The Preferences window enables you to customize several options in SAS Studio.

To change your preferences, select **Options** ⇨ **Preferences**.

Note: Preferences do not apply to nodes in a flow.

Setting the Start-Up Preferences

From the **Start Up** page, you can specify these options for starting SAS Studio.

Option	Description
Tabs to Open	Specifies the tabs to be displayed when you start the SAS Studio application. You can choose from these options: ■ **None** starts SAS Studio with no tabs open in the work area. This option is selected by default. ■ **Restore tabs listed in Open Files pane** restores the tabs that were open in your prior session of SAS Studio. When this option is selected, the tabs are also restored if you switch between the Standard and Interactive perspectives in a SAS Studio session. Any temporary files that were open are not restored. **Note:** The tabs are not necessarily restored in the same order they were in when you closed SAS Studio. Tabs that display a large amount of data can take longer to open and might be opened last. ■ **Open a new file** enables you to choose whether to open a new program or flow tab. If you are using the Interactive perspective, this option opens a program tab.

Option	Description
Show Start page	Opens the **Start Page** tab when you start the SAS Studio application. To automatically close the **Start Page** tab when you open another tab in the work area, select **Automatically dismiss**.
Restore open folder state in Explorer pane	Automatically reopens the folders that were expanded in the **Explorer** section of the navigation pane in your prior session of SAS Studio.

Setting General Preferences

From the **General** page, you can set these options.

Option	Description
Display tab after submission	Specifies the tab to display after you run a program or a task. You can choose to display the code, log, or results tabs. If you are running the Standard perspective, you can also choose to display output data tab. The **Results** tab is displayed by default.
When activating output data tabs, automatically open	Specifies which output data tab to display when you run a program or task that generates data.
View output data in a pane	Displays the list of output data for programs, tasks, and flows in a collapsible pane. If this option is not selected, then the list of output data is displayed in a compact menu.
Display warning if text files are larger than specified size	Displays a warning when you open a text file that is larger than the specified size in megabytes (MB). Text files include file types such as *.sas, *.txt, and *.csv. If you open a file that is larger than the specified size, the file is truncated and opened in Read-Only mode. The default size is 10 MB.
Default text encoding	Specifies the character-set encoding that is used when text files, such as *.sas programs, are read or written. The default value is UTF-8. For a list of some of the encoding options and the languages that they are associated with, see Appendix 3, “Text Encoding Options and Language Mappings,” on page 275.

Option	Description
Display hidden folders and items on the file system	Displays hidden files and folders in the **Explorer** section of the navigation pane.
Enable document recovery	Enables you to recover the most recently saved version of your open files, such as SAS programs, flows, queries, and tasks, if SAS Studio times out while you are away or closes unexpectedly due to a situation such as a power loss. If you select this option, you can use the Document Recovery window to choose the files that you want to recover. For more information, see "Using the Document Recovery Window" on page 22. This option is selected by default. You can use the **Document save interval** option to specify how often you want to save your files. The default interval is 30 seconds.
Automatically refresh libraries after each submission	Refreshes the Libraries section of the navigation pane after you run a program, task, or query so that any changes to the libraries are displayed immediately. However, selecting this option can cause you to lose your place in the list of libraries and affect performance. This option is selected by default. **Note:** If you do not select this option and you run code that creates a new file or library, you must click ⋮ > **Refresh** to display the changes. To refresh an individual table or library, right-click the table or library and select **Refresh**. If you create a new file or library using the SAS Studio user interface, those changes are displayed automatically even if this option is not selected.
Automatically refresh files and folders after each submission	Refreshes the Explorer section of the navigation pane after you run a program, task, or query so that any changes to the files and folders are displayed immediately. However, selecting this option can cause you to lose your place in the list of files and folders and affect performance. This option is selected by default. **Note:** If you do not select this option and you run code that creates a new file or folder, you must click ⋮ > **Refresh** to display the changes. To refresh an individual folder, right-click the folder and select **Refresh**. If you create a new file or folder using the SAS Studio user interface, those changes are displayed automatically even if this option is not selected.
Show suggested commands	Displays the possible command options when you type in the command line.

Setting the Code and Log Preferences

From the **SAS Programs** ⇨ **Code and Log** page, you can specify these options for the code and log tabs.

Option	Description
Show generated code in the SAS log	Displays the ODS statements, %LET statements, and any other code that is automatically generated by SAS in the log file. This option applies to both SAS tasks and SAS program files.
Show custom code in the SAS log	Includes custom code in the log file.
Use enhanced log viewer	Automatically generates the log each time the program is run. This option is selected by default. If you do not select this option, the log is loaded only when the Log tab is active. Using the enhanced log viewer can affect your performance. You can choose from these options when you use the enhanced log viewer: ■ **Generate errors, warnings and notes index** creates a section at the top of the log that provides easy access to all of the errors, warnings, and notes in the log. When you click a message, SAS Studio highlights it for you in the log so that you can see exactly where the message occurs in the log. ■ **Display error and warning icons in the Code tab gutter** displays error and warning icons in the gutter of the Code tab. When you click an icon, SAS Studio highlights the message in the log. Icons are displayed in the gutter only when you run the entire program and when **Automatically clear log** is selected. Icons cannot be displayed if **Append log** is selected. ■ **Show only errors and warnings in the SAS log** displays only error and warning messages in the log. Informational notes are not displayed.
Stream log updates while a procedure is running	Displays log updates as a procedure is processed. If you do not select this option, the log is displayed when the procedure has finished running. This option is selected by default and might have a slight effect on performance. This option is available only if you are running the Standard perspective.
Display warning if log is larger than specified size	Displays a warning when the number of lines in the log exceeds the number that is specified in the **Maximum number of log lines** option. The value can range from 100 to 10,000.

Option	Description
Automatically clear log	Clears the log each time you run code. This option is selected by default. If you do not select this option, then a **Clear log** button is available in the Program Editor. This option is available only if you are running the Standard perspective.
Append log	Appends new log information to the existing log. You can choose to append the logs for programs, tasks, or both programs and tasks. This option is available only if you are running the Standard perspective.
Program tab layout	Specifies the arrangement of the **Code**, **Log**, **Results**, and **Output Data** tabs in the work area.
Display tabs in the Preview section	Specifies which tabs to include in the preview section of the program tab after you run code. **Note:** This option is applied only to programs that you open after you have selected the option.

Setting the Results Preferences

From the **Results** page, you can specify these options.

Option	Description
Display warning if results are larger than specific size	Displays a warning message when you attempt to open a results file that is larger than the size that is specified in the **Maximum size** option. You can enter a value between 0.5 and 10 MB. The default value is 3 MB.
Generate table of contents	Creates a table of contents for the output.
Produce HTML output	Generates results in HTML format. This is the default results format. If you clear this option, then the **Results** tab is not displayed when you run a program.
Specify an HTML output style	Enables you to specify the style that is applied to results in HTML. If this option is not selected, SAS Studio uses the style that is associated with the application theme. If this option is selected, you can use the **Output style** drop-down list to select the style that is applied to the results. **Note:** This option is available only in SAS Studio 5.2 (Enterprise).

Option	Description
Generate HTML graphs as SVG	Creates SVG graphs instead of PNG graphs in HTML output. SVG graphs maintain clarity when you zoom in and out.
Enable accessible graph option	Adds accessibility metadata to graphs that are created by ODS Graphics. Users with disabilities access the accessibility metadata using SAS Graphics Accelerator. This option is available only if you are running SAS 9.4M4 or later. For more information, see the ODS HTML5 Statement in *SAS Output Delivery System: User's Guide*. **Note:** The SAS code that is associated with this option is displayed in the log file only if you have selected the **Show generated code in the SAS log** option in the SAS Programs > Code and Log preferences.
Produce PDF output	Generates results in PDF format. This option is not selected by default and is available only if you are running the Standard perspective.
Output style	Displays the style that is applied to results in PDF. To change the style that is applied to the results, select another style from the drop-down list. This option is available only if you are running the Standard perspective.
Generate table of contents	Creates a table of contents in the PDF file. This option is available only if you are running the Standard perspective.
Enable accessible PDF option	Adds accessibility metadata to the PDF file that enables the file to be accessed by assistive technology such as a screen reader. When metadata is added, the file is often called a "tagged PDF" and follows the PDF/Universal Accessibility (PDF/UA) format.
Produce WORD output	Generates results in Microsoft Word format. This option is available only if you are running the Standard perspective.
Output style	Displays the style that is applied to results in Word format. To change the style that is applied to the results, select another style from the drop-down list. This option is available only if you are running the Standard perspective.
Produce RTF output	Generates results in RTF format. This option is not selected by default and is available only if you are running the Standard perspective. **Note:** Microsoft has discontinued enhancements to the RTF specification. It is recommended that you use the **Produce WORD output** option to generate your output. Using the **Produce WORD output** option also ensures that your output is accessible.

Option	Description
Output style	Displays the style that is applied to results in RTF. To change the style that is applied to the results, select another style from the drop-down list. This option is available only if you are running the Standard perspective.
Produce EXCEL output	Generates results in Microsoft Excel format. This option is available only if you are running the Standard perspective.
Output style	Displays the style that is applied to results in Excel format. To change the style that is applied to the results, select another style from the drop-down list. This option is available only if you are running the Standard perspective.
Produce PPT output	Generates results in Microsoft PowerPoint format. This option is available only if you are running the Standard perspective.
Output style	Displays the style that is applied to results in PowerPoint format. To change the style that is applied to the results, select another style from the drop-down list. This option is available only if you are running the Standard perspective.

Note: If you want to use a custom style, you must customize the SAS Studio output environment. For more information, see Appendix 4, "Customized Output Environment," on page 277.

Setting Editors Preferences

From the **Editors** page, you can set the options for the SAS and XML language editors.

Option	Description
Enable snippet abbreviations	Enables you to insert snippets into a program by entering @*snippet-name* or entering @ and using the pop-up window to select from a list of snippets in your My Snippets folder. This option is selected by default. For more information, see "Create a Snippet" on page 66.
Editor Options	Enables you to specify editing and display options as well as appearance options for the code editor. For more information, see "Setting General Code Editor Preferences" on page 267 and "Setting Code Editor Appearance Preferences" on page 268.

Option	Description
Enable syntax highlighting	Displays the text in the code editor in different colors to help you identify different elements in the syntax. This option is selected by default.
Enable hint when you hover over keywords	Displays the syntax help window when you position the mouse pointer over a valid SAS keyword in your program. If this option is not selected, then you can view the syntax help by right-clicking a keyword and selecting **Syntax help**. This option is not selected by default.

Setting General Code Editor Preferences

From the **Editors** page, click **Editor Options** and then click **Behavior and Appearance** to set editing and display options for the code editor.

Option	Description
Show autocomplete list	Turns on the autocomplete feature of the code editor. This feature can predict the next keyword that you want to type before you actually type it completely. For more information, see "Using the Autocomplete Feature" on page 35.
Auto-indent	Automatically indents a new line by the amount of space that the previous line is indented.
Auto-insert block comments	Automatically inserts the closing comment tag when you enter /* and press Enter so that you can insert comments that span multiple lines.
Use overtype mode	Replaces existing characters with new characters when you type in the code editor.
Auto-pair quotation marks	Automatically inserts a closing quotation mark when you type a quotation mark and positions the cursor in between them.
Highlight matching brackets	Highlights the matching bracket when you position the cursor on a bracket.
Auto-pair brackets	Automatically inserts a closing bracket when you type a left bracket and positions the cursor in between them.
Enable code folding	Enables you to collapse and expand sections of code in the code editor.

Option	Description
Enable line highlighting	Automatically highlights the line of code that you specify when you use the **Go to line** feature.
Show indent guides	Displays vertical lines that indicate where an indented section of code begins and ends.
Show line-length guide	Displays a vertical line in the Program Editor at the column position that you specify. The default column position is 80.
Show invisible characters	Displays characters such as spaces and carriage returns in the Program Editor.
Show line numbers	Displays line numbers in the leftmost column of the program tab.
Print line numbers	Includes line numbers in the leftmost column when you print a program.
Wrap text	Displays text that continues past the right edge of the page on the next line.
Tab width	Specifies the number of spaces that are inserted into your text when you insert a tab character. The default value is four spaces for each tab character.
Substitute spaces for tabs	Inserts the number of spaces specified in the **Tab width** option instead of a single tab character.

Setting Code Editor Appearance Preferences

You can specify appearance options for both SAS code and XML code in the code editor. From the **Editors** page, click **Editor Options** and then click **Colors and Fonts for SAS** or **Colors and Fonts for XML** to set appearance options for the code editor.

Option	Description
Syntax theme	Specifies whether to use a light or dark background in the code editor.
Font	Specifies the font that is used in the code editor.
Editor background	Specifies the background color that is used in the code editor.

Option	Description
Enable syntax highlighting	Enables you to specify the text style and background color of specific file elements in the code editor.

Setting Tables Preferences

From the **Tables** page, you can set the options for the table viewer.

Option	Description
Restrict number of tables displayed when expanding libraries	Limits the number of tables that are displayed when you expand a library in the **Libraries** section of the navigation pane. Use the **Tables displayed** box to specify the maximum number of tables to display. You can view any additional tables that are not displayed by clicking **More** at the end of the list of tables. Any changes that you make to this option take effect when you refresh the Libraries tree or restart SAS Studio.
Column headers	Specifies whether column name or labels are displayed in the table viewer. Column names are displayed by default.
Rows per page	Specifies the number of rows to display per page of data. The default value is `200`. **Note:** You can override this option for an open table in the table viewer by clicking ⋮ on the table viewer toolbar and selecting **Row paging**.
Columns displayed	Specifies the maximum number of columns to display in the table viewer. The default value is `100`. Increasing the number of columns that are displayed might affect your performance.
SAS variable name policy	Enables you to specify one of the following sets of rules to apply to SAS variable names. ■ **ANY** specifies that the variable names can begin with or contain any characters, including blanks, must contain at least one character, and cannot contain any null bytes. Variable names can contain mixed-case letters as well as special and multi-byte characters. Names can be up to 32 bytes in length. This option is selected by default. Leading blanks are preserved, but trailing blanks are ignored. ■ **V7** specifies that the variable names must begin with a letter of the Latin alphabet (A-Z, a-z) or the underscore character. They cannot contain blanks or special characters except for the underscore and cannot be

Option	Description
	assigned the names of special SAS automatic variables or variable list names. Variable names can contain mixed-case letters and can be up to 32 bytes in length. ■ **UPCASE** specifies that the variable name follows the same rules as V7, except that the variable name is uppercase, as in earlier versions of SAS.
SAS member name policy	Enables you to specify one of the following sets of rules to apply to SAS data set names, SAS data view names, and item store names. Any changes you make to this option take effect when you start SAS Studio or reset your SAS session. ■ **EXTEND** (Extended member names) specifies that names must contain at least one character (letters, numbers, valid special characters, and national characters), cannot begin with a blank or a period, and cannot include a null byte. Names can contain mixed-case letters and can be up to 32 bytes in length. Leading blanks are preserved, but trailing blanks are ignored. Special characters cannot include the / \ * ? " < > |: - characters. This option is selected by default. ■ **COMPATIBLE** (Basic member names) specifies that the names must begin with a letter of the Latin alphabet (A–Z, a–z) or the underscore character and cannot contain blanks or special characters except for the underscore. Names can contain mixed-case letters and can be up to 32 characters in length.

Setting Task Preferences

From the **Tasks** page, you can set the options for the generated SAS code and the task layout in the SAS Studio workspace. These options are available only if you are running the Standard perspective.

Option	Description
Generate header comments for task code	Adds comments before the generated code for a SAS task.
Automatically format generated code	Automatically formats any code that is generated by a task.
Show Preview section	Specifies whether to display the preview section of the task tab after you run a task and which tabs to include in the preview.

Setting Flow Preferences

From the **Flows** page, you can set the options for flows. These options are available only if you are running the Standard perspective.

Preview tab layout	Specifies how the preview tabs are displayed in relation to the flow canvas. By default, the preview tabs are displayed horizontally, or below the flow canvas.

Setting Background Submit Preferences

From the **Background Submit** page, you can set the options for background submissions. These options are available only if you are running the Standard perspective.

Location of log and output files	Specifies where to save the output and log files. You can choose from these options: ■ **Use same name and location as file** uses the name and location of the program file for the log and output files. ■ **Prompt for output and log file names** prompts you to specify a location in which to save the log and output files. ■ **Specify names and location** enables you to specify a name and location for the log and output files.
Log or output files exist	Specifies how to handle the background submission if a log and output file already exist. You can choose from these options: ■ **Replace existing files** automatically replaces existing log and output files of the same name. ■ **Cancel the submission** cancels the background submission when there are existing log and output files of the same name. ■ **Automatically add a timestamp** saves all log and output files by creating a unique file name for each file. The log and output files are saved as `program-name(YYYY-MM-DD HH:MM:SS)`. ■ **Prompt** displays a message window to confirm that you want to replace or specify a new location for the existing log and output files before submitting the background submission. If you select `Cancel`, the background submission is canceled.

Setting Query Preferences

From the **Query** page, you can set the options for queries. These options are available only if you are running the Standard perspective.

Generate code using	Specifies whether to use PROC SQL or PROC FEDSQL to generate the query code. The default value is PROC SQL. For more information, see “Generating a FedSQL Query” on page 98.
Columns to display on Select tab for PROC SQL	Specifies the columns that are displayed on the **Select** tab in addition to the table and column name. This option applies only to queries that are using PROC SQL to generate the query code.
Show Preview section	Specifies whether to display the preview section of the query tab after you run a query and which tabs to include in the preview.

Setting Import Preferences

From the **Import** page, you can set the options for importing data. These options are available only if you are running the Standard perspective.

Show Preview section	Specifies whether to display the preview section of the import tab after you import data and which tabs to include in the preview.

Appendix 2

Using the Expression Builder

Building an Expression

You can use the Expression Builder when you are creating a filter on your data or when you are creating a filter, join, or calculated column in a query.

Note: The following features are not available in the Expression Builder if you are creating your expression for a node in a flow:

- Values and Log area in the Expression Builder window
- **Get Values** button that enables you to choose from a list of values and search for a specific value
- **Validate** button to validate the syntax of your expression

The Expression Builder window contains three main areas:

Data and Functions area	provides access to the columns in your data and the functions that you can use.
expression area	displays the expression as you build it. You can also type an expression in this area.
Values and Log area	the Values tab displays distinct values for columns. The Log tab displays the validation log. If you are using the Expression Builder to create a calculated column, this area also includes a Properties tab.

Note: The Values and Log area is not available if you are creating an expression for a node in a flow.

You can add columns and functions to the expression by double-clicking the name. You can also add column names by right-clicking the column and selecting **Add to expression** or by dragging the column to the expression box.

To populate a list of distinct values for the column, select the column on the Data tab, and then click **Get Values** on the Values tab. You can add values to the expression by double-clicking the value, clicking **Add to Expression** on the toolbar, or dragging the value to the expression box.

Note: You can also type the expression directly in the expression box located above the table viewer.

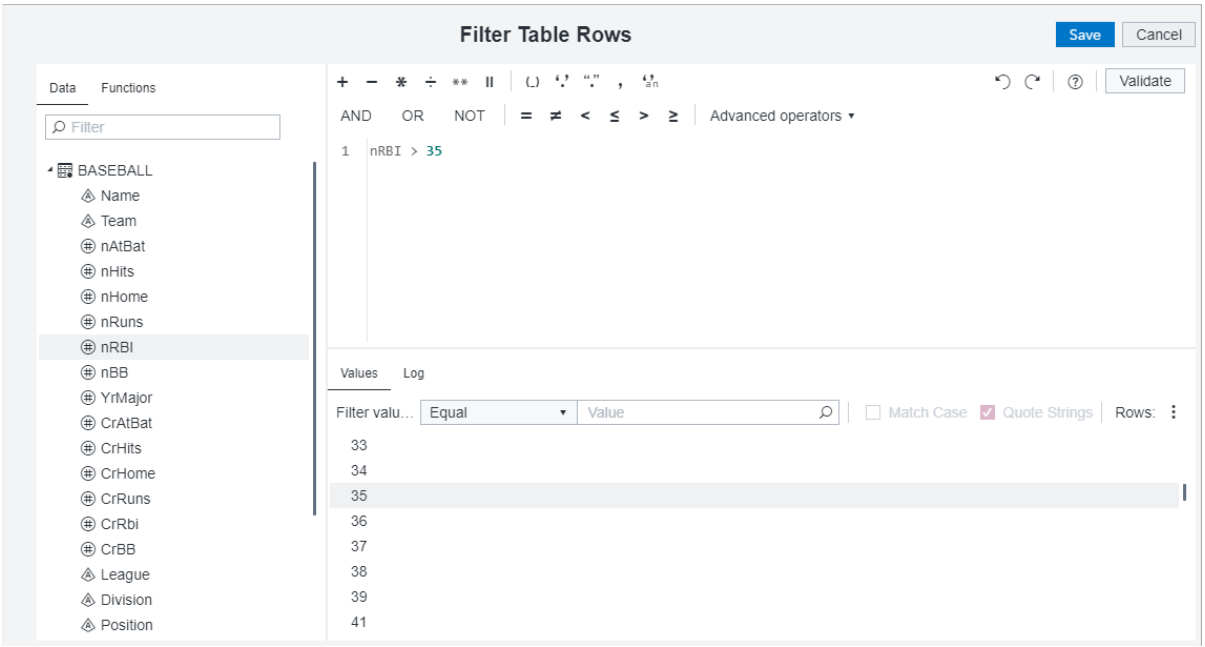
Note: The option to choose from a list of values and search for a specific value is not available if you are creating the expression for a node in a flow.

To validate the syntax of your expression, click **Validate**. The Log tab displays any errors in your expression syntax.

Note: The option to validate your syntax is not available in the following situations:

- You are using the Interactive perspective. For more information, see "Understanding Perspectives" on page 25.
- You are creating an expression for a node in a flow.

Note: The = operator does not give expected results with decimal values. To create a filter with decimal values, use the Between operator.



Appendix 3

Text Encoding Options and Language Mappings

About the Text Encoding to Language Mappings

The following table lists some of the text encoding options and the languages they are associated with. For more information, see “Setting General Preferences” on page 261.

Text Encoding Options and Language Mappings

Text Encoding Option	Language
Windows-1250	(Central European languages): Polish, Czech, Slovak, Hungarian, Slovenian, Serbian Latin, Croatian, Bosnian, Romanian, Albanian

Text Encoding Option	Language
Windows-1251	(Cyrillic languages): Russian, Byelorussian, Bulgarian, Serbian Cyrillic, Macedonian, Ukrainian
Windows-1252	(Western European languages): Afrikaans, Basque, Catalan, Valencian, Welsh, Danish, German, English, Spanish, Basque, Finnish, Faroese, French, Western Frisian, Irish, Galician, Indonesian, Icelandic, Italian, Inuktitut, Luxembourgish, Malay, Norwegian Bokmål, Dutch, Norwegian Nynorsk, Portuguese, Quechua, Romansh, Northern Sami, Swedish, Swahili, Tswana, Xhosa, Zulu
Windows-1253	Greek
Windows-1254	Turkish
Windows-1255	Hebrew
Windows-1256	Arabic
Windows-1257	(Baltic languages): Estonian, Latvian, Lithuanian
Windows-1258	Vietnamese

Appendix 4

Customized Output Environment

Overview

You must customize the SAS Studio output environment to perform any of these tasks:

- generate output for other output destinations
- send your results to another location
- use a custom style for your output
- use an image format other than the default
- create a drill-down graph
- create an animated GIF or SVG image

To customize the SAS Studio output environment, first disable the default output environment in order to conserve system resources. Next, establish your own output environment, and then execute the SAS statements that are required to generate your output. Use ODS statements, ODS procedures, or ODS options in your SAS program to define the environment that you need.

As a best practice, if your SAS program requires a customized output environment in SAS Studio, your program should always perform these steps:

1. Create a file reference for your ODS output. You can use the &_SASWS_ macro variable that is defined in SAS Studio to reference your home directory as shown in the following statement:

   ```
   filename odsout "&_SASWS_/charts";
   ```

   If you want to store your image files in a separate directory, create a second file reference for your image files as shown in the following statement:

   ```
   filename ods1out "&_SASWS_/charts/images";
   ```

   Note: The directories that you specify must already exist, and you must have Write access to the directories.

2. To conserve system resources, disable the default output environment by using the following statement:

   ```
   ods _all_ close;
   ```

3. Open the desired ODS destination. Use the PATH= option to specify the file reference that you created for your ODS output. If you created a separate file reference for your image files, use the GPATH= option to specify the image output file reference. Here is an example:

   ```
   ods html path=odsout gpath=ods1out file="saleschart.html";
   ```

4. Execute the SAS statements that are required to generate your output.

5. Close your ODS destination.

When you disable the default SAS Studio output environment, results are no longer displayed on the **Results** tab for the duration of your program. The results are generated only by the ODS destination that you open.

Generate Output for Other Output Destinations

If you need to generate output other than the default HTML5, PDF, or RTF output, you must open your own ODS destination. Examples of output destinations include HTML, PowerPoint, and LISTING. After you disable the default output environment, use an ODS statement to open your own output destination. Here is an example:

```
filename odsout "&_SASWS_/charts";
ods _all_ close;
ods powerpoint path=odsout file="filename";
```

To access the dictionary of ODS statements, see *SAS Output Delivery System: User's Guide*.

Send Your Results to Another Location

When you execute a program in SAS Studio, you can download the output from the **Results** tab to your local machine. If you want to send your output directly to another location, you must open your own ODS destination. By default, output files that are generated by the ODS destinations that you open are written to your home directory.

If you want to send the results to a specific location, use a FILENAME statement to define a file reference to the desired location. You can use the &_SASWS_ macro variable to reference your home directory. After you create the file reference, use the PATH=*file-reference* option in your ODS statement. Here is an example:

```
filename odsout "&_SASWS_/charts";
ods _all_ close;
ods html path=odsout file="sales.htm";
```

In this case, file sales.htm and any image files that are generated are written to subdirectory **charts** in your home directory.

Use a Custom Style for Your Output

When you need to use a custom ODS style such as a corporate style for your results in SAS Studio, you must open your own ODS destination. You cannot specify a custom style for the default results. Use the STYLE= option in your ODS statement to specify your custom style. Here is an example:

```
filename odsout ="&_SASWS_/charts";
ods _all_ close;
ods html path=odsout file="filename.htm" style=style-name;
```

To create a custom style, use the ODS TEMPLATE procedure, CSSStyles, or the STYLE= option. For more information, see *SAS Output Delivery System: User's Guide*.

Use an Image Format Other Than the Default

When you need to use an image format other than the default, you must specify the desired output format, and then open your own ODS destination. To specify the image format:

- If you are using SAS/GRAPH to create your graphs, specify the DEVICE= option in an OPTIONS or GOPTIONS statement. For more information, see *SAS/GRAPH: Reference*.

  Note: The ACTIVEX and JAVA graphics output devices are not recommended in SAS Studio 5.2 or later. Your graphs might not work as expected if you use these output devices.

- If you are using ODS Graphics to create your graphs, specify the OUTPUTFMT= option in an ODS GRAPHICS statement. For more information, see *SAS Output Delivery System: User's Guide*.

Create a Drill-down Graph

When you need to create a drill-down graph in SAS Studio, you must open your own ODS destination. Drill-down graphs provide a convenient means for users to explore complex data. In a drill-down graph, certain elements of the graph contain active links. When a user clicks a linked element, the linked resource appears in a new browser window by default.

For more information, see the following documents:

- If you are using SAS/GRAPH to create the graph, see *SAS/GRAPH: Reference*.
- If you are using the Graph Template Language to create the graph, see *SAS Graph Template Language: User's Guide*.

Create an Animated GIF or SVG Image

When you need to create an animated graph in SAS Studio, you must open your own ODS destination. An animated graph displays a series of charts automatically when the graph is viewed in a web browser or other viewer that supports animation. The animation plays as a sequence of graphs in a slide-show fashion with a delay between each graph. The sequence can play only one time, loop a fixed number of times and then stop, or loop indefinitely.

For more information, see the following documents:

- If you are using SAS/GRAPH to create the graph, see *SAS/GRAPH: Reference*.
- If you are using the Graph Template Language to create the graph, see *SAS Graph Template Language: User's Guide*.

Appendix 5

SAS Studio Command Line

About the Command Line Interface

You can use the command line interface to access SAS Studio using the keyboard to control the application. To open the command line, select **View** ⇨ **Command** or press Alt+M in Windows environments and Option+M in MacOS environments.

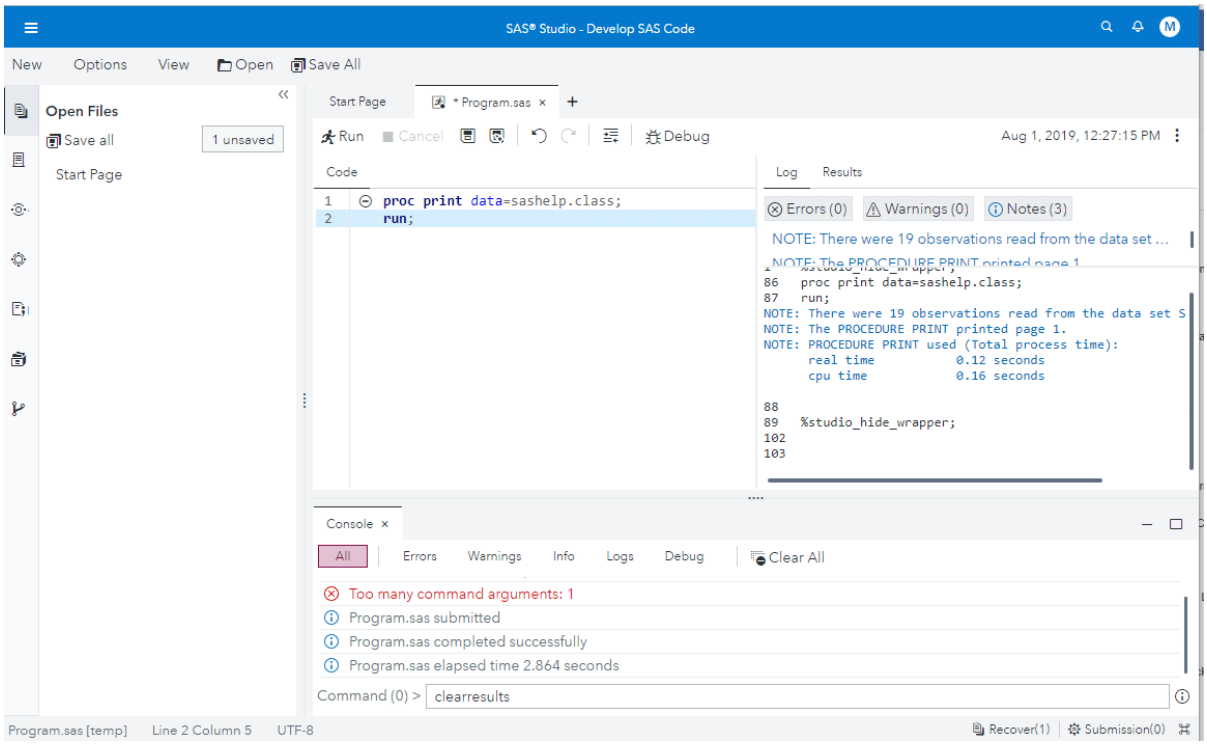
A command can have options and arguments. Options have a dash at the beginning. Arguments that contain white space characters must be enclosed in quotes. Arguments for a command must be in a specified order, but options can be mixed into the argument list in any order.

Enter the command you want to run in the command line. When you are entering text in the command line, SAS Studio suggests possible commands. To select a command from the list of suggestions, use the up and down arrow keys and press Enter. To run the command, press Enter again. To run multiple commands, enter the commands separated by semicolons.

Note: The availability of a command depends where you are in the user interface. For example, only a subset of commands are available when working with flows.

Commands are executed in order and the next command starts when the previous one completes. If a command does not complete and processing halts, use Ctrl +Shift+X to clear the command queue.

When a command runs, information is added to the SAS Studio console. If the command fails, an error message appears in the console. To open the console, select **View** ⇨ **Console** or press Alt+C.



To view previous commands, press Ctrl+Arrow Up to go up one step in the command history. Press Ctrl+Arrow Down to go down one step in the command history.

To abort the execution of a list of commands:

- In Windows environments, press Ctrl+Shift+X.
- In OS X environments, press Command+Shift+X.

Commands in Standard Perspective

These commands are available in the standard perspective.

Table A5.1 *Commands in Standard Perspective*

Command Name	Description
BGSUBMIT	Starts a background submission and runs the code currently selected in the code editor. If no code is selected, all the code is run.
CANCEL	Stops running the current submission.
CLEAR	Deletes the content in the **Log** and **Results** tabs.
CLEARCODE	Deletes all code from the **Code** tab.

Command Name	Description
CLEARHISTORY	Deletes submission history.
CLEARLOG	Deletes the content in the **Log** tab.
CLEARRESULTS	Deletes the content in the **Results** tab.
CLOSE	Closes the selected tab. **Note:** This command is available when working with flows.
CLOSEALL	Closes all tabs. **Note:** This command is available when working with flows.
CONSOLE	Opens the console for SAS Studio. **Note:** This command is available when working with flows.
DEBUG	Starts the DATA Step debugger. **Note:** The cursor must in the DATA step code for this command to work.
EXPORTTABLE <*path-to-output-folder*> <*filename.ext*> <-csv | -dlm | -tab><-force>	Opens the Export Table window, when the data view is open and if output path is specified. If you specify the output path and filename of the table view, the Export Window does not open. For more information, see "Specifying the Path to the Content: SAS Content or SAS Compute" on page 291. These options are available: ■ `-csv` | `-dlm` | `-tab` specifies the export type for the file. If this option is not specified, the export type is determined from the extension on the filename. ■ `-force` overwrites an existing file with the same name.
FIND *target-string* <-case | -c> <-up | -u><-word | -w>	Searches for the specified target string in the code. These options are available. ■ `-case` | `-c` specifies that the search is case sensitive. By default, the case is ignored. ■ `-up` | `-u` searches up. By default, the search goes down the file. ■ `-word` | `-w` searches for whole word matches. By default, the search does not check for word boundaries. **Note:** Use the NEX command to move to the next instance of the target string. Use the PREV commands to move to the previous instance of the target string.

Command Name	Description
GOTOLINE <*line-number*>	Moves the cursor to the specified line number. If you do not specify a line number, SAS Studio prompts you for this information.
KEYS <edit>	Opens the Keyboard Shortcuts window. If you specify the `edit` option, the Manage Keyboard Shortcuts window opens enabling you to edit an existing shortcut. From this window, you can also import and export shortcuts and reset all shortcuts. **Note:** This command is available when working with flows.
NEWIMPORT	Opens an import data window on a new tab in the work area. **Note:** This command is available when working with flows.
NEWFLOW	Opens a new flow. **Note:** This command is available when working with flows.
NEWPROGRAM	Adds a new **Program** tab to the work area. **Note:** This command is available when working with flows.
NEWQUERY	Opens a query window on a new tab in the work area. **Note:** This command is available when working with flows.
NEWTASKTEMPLATE	Opens a blank task template on a new tab in the work area. **Note:** This command is available when working with flows.
NEXT	Finds the next occurrence of the pattern specified in the FIND command.
OPENENCODED <*path-to-file/filename.ext*> <-encoding>	Opens the Open File window, if no options are specified. This window does not open if you specify a filename. To open a specific file, specify the path and filename. For more information, see "Specifying the Path to the Content: SAS Content or SAS Compute" on page 291. Use the `-encoding` option to specify the encoding of the file (for example, `-UTF8`). **Note:** This command is available when working with flows.

Command Name	Description
OPENFILE *<path-to-file/filename.ext>*	Opens the Open File window, if no filename is specified. This window does not open if you specify a filename. To open a specific file, enter the path and filename.For more information, see "Specifying the Path to the Content: SAS Content or SAS Compute" on page 291. **Note:** This command is available when working with flows.
PREFERENCES	Opens the Preferences window. **Note:** This command is available when working with flows.
PREV	Finds the previous occurrence of the pattern specified in the FIND command.
REPLACE *target-string replacement-string*<-case | -c><-up | -u><-word | -w><-all | -a>	Searches for the specified target string in the code and replaces it with the specified content. These arguments are available. ■ `-case` | `-c` specifies that the search is case sensitive. By default, the case is ignored. ■ `-up` | `-u` searches up. By default, the search goes down the file. ■ `-word` | `-w` searches for whole word matches. By default, the search does not check for word boundaries. ■ `-all` | `-a` replaces all occurrence of the target string.
RESET	Resets the SAS Studio session. **Note:** This command is available when working with flows.
SAVE *<output-path> <filename.ext>* <-force> <-encoding>	Opens the Save As window, if no output path is specified.. This window does not open if you specify an output path. For more information, see "Specifying the Path to the Content: SAS Content or SAS Compute" on page 291. If no filename is specified, the file keeps its original filename. To change the filename, you must specify the `filename.ext` argument. You can also specify these options: ■ `-force` overwrites an existing file with the same name. ■ `-encoding` specifies the encoding of the file, such as `-UTF8`.
SAVEALL	Saves all open files.

Command Name	Description
SAVECODE <*output-path*> <*filename.ext*> <-force> <-encoding>	Opens the Save As window, if no output path is specified. This window does not open if you specify an output path. For more information, see "Specifying the Path to the Content: SAS Content or SAS Compute" on page 291. If no filename is specified, the original filename is used. To change the filename, you must specify the `filename.ext` argument. You can also specify these options: ■ `-force` overwrites an existing file with the same name. ■ `-encoding` specifies the encoding of the file, such as `-UTF8`.
SAVELOG <*output-path*> <*filename.ext*> <-force> <-encoding>	Opens the Save As window, if no output path is specified. This window does not open if you specify an output pathFor more information, see "Specifying the Path to the Content: SAS Content or SAS Compute" on page 291. If no filename is specified, the original filename is used. To change the filename, you must specify the `filename.ext` argument. You can also specify these options: ■ `-force` overwrites an existing file with the same name. ■ `-encoding` specifies the encoding of the file, such as `-UTF8`.
SAVEPACKAGE <*output-path*> <*filename.ext*> <-force> <-encoding>	Opens the Save As window, if no output path is specified. This window does not open if you specify an output path.For more information, see "Specifying the Path to the Content: SAS Content or SAS Compute" on page 291. If no filename is specified, the original filename is used. To change the filename, you must specify the `filename.ext` argument. You can also specify these options: ■ `-force` overwrites an existing file with the same name. ■ `-encoding` specifies the encoding of the file, such as `-UTF8`.
SAVERESULTS <*output-path*> <*filename.ext*> <-force> <-encoding>	Opens the Save As window, if no output path is specified.. This window does not open if you specify an output path.For more information, see "Specifying the Path to the Content: SAS Content or SAS Compute" on page 291. If no filename is specified, the orignal filename is used. To change the filename, you must specify the `filename.ext` argument. You can also specify these options: ■ `-force` overwrites an existing file with the same name.

Command Name	Description
	■ `-encoding` specifies the encoding of the file, such as `-UTF8`.
SAVESUMMARY <*output-path*> <*filename.ext*> <-force> <-encoding>	Opens the Save As window, if no output path is specified. This window does not open if you specify an output path. For more information, see "Specifying the Path to the Content: SAS Content or SAS Compute" on page 291. If no filename is specified, the file keeps its original filename. To change the filename, you must specify the `filename.ext` argument. You can also specify these options: ■ `-force` overwrites an existing file with the same name. ■ `-encoding` specifies the encoding of the file, such as `-UTF8`.
SEARCH <-max | -restore | -off>	Displays the **Search** tab. These options are available: ■ `-max` maximizes the preview area. ■ `-restore` restores the preview area. ■ `-off` closes the **Search** tab. **Note:** This command is available when working with flows.
SELECTCODE	Selects the **Code** tab.
SELECTLOG	Selects the **Log** tab.
SELECTOUTPUT	Selects the **Output** tab.
SELECTRESULTS	Selects the **Results** tab.
SUBMISSIONS <-max | -restore | -off>	Displays the **Submissions** tab. These options are available: ■ `-max` maximizes the preview area. ■ `-restore` restores the preview area. ■ `-off` closes the **Submissions** tab. **Note:** This command is available when working with flows.
SUBMIT	Runs the currently selected code or all the code in the editor if no code is selected.
VIEWTABLE <*library.table*> <-last>	Opens the Select Table window if no filename is specified. Use the `library.table` argument to specify the table to view. The Select Table window does not open. If no table is specified, use the `-last` option to open the last table that was generated in the current session.

Command Name	Description
	Note: This command is available when working with flows.
ZOOM <-max> <-restore>	Specifies the size of the tab area. These arguments are available. ■ `-max` specifies to maximize the size of the tab area. ■ `-restore` returns the tab area to the default size. **Note:** This command is available when working with flows.

Commands in Interactive Perspective

These commands are available in the interactive perspective.

Table A5.2 *Commands in Interactive Perspective*

Command Name	Description
CANCEL	Stops running the current submission.
CLEAR	Deletes the content in the **Log** and **Results** tabs.
CLEARCODE	Deletes all code from the **Code** tab..
CLEARLOG	Deletes the content in the log.
CLEARRESULTS	Deletes the content in the **Results** tab.
CLOSE	Closes the selected tab.
CLOSEALL	Closes all tabs.
CONSOLE	Opens the console for the command line.
FIND *target-string* <-case | -c> <-up | -u> <-word | -w>	Searches for the specified target string in the code. These options are available. ■ `-case` | `-c` specifies that the search is case sensitive. By default, the case is ignored. ■ `-up` | `-u` searches up. By default, the search goes down the file.

Command Name	Description
	■ `-word` | `-w` searches for whole word matches. By default, the search does not check for word boundaries.
GOTOLINE <*line-number*>	Moves the cursor to the specified line number. If you do not specify a line number, SAS Studio prompts you for this information.
KEYS <edit>	Opens the Keyboard Shortcuts window. If you specify the `edit` option, the Manage Keyboard Shortcuts window opens enabling you to edit an existing shortcut. From this window, you can also import and export shortcuts and reset all shortcuts.
NEWPROGRAM	Adds a new **Program** tab to the work area.
NEXT	Finds the next occurrence of the pattern specified in the FIND command.
OPENENCODED <*path-to-file/filename.ext*> <-encoding>	Opens the Open File window, if no options are specified. This window does not open if you specify a filename. To open a specific file, specify the path and filename. For more information, see "Specifying the Path to the Content: SAS Content or SAS Compute" on page 291. Use the `-encoding` option to specify the encoding of the file (for example, `-UTF8`).
OPENFILE <*path-to-file/filename.ext*>	Opens the Open File window, if no filename is specified. This window does not open if you specify a filename. To open a specific file, enter the path and filename. For more information, see "Specifying the Path to the Content: SAS Content or SAS Compute" on page 291.
PREFERENCES	Opens the Preferences window.
PREV	Finds the previous occurrence of the pattern specified in the FIND command.
REPLACE *target-string replacement-string* <-case | -c> <-up | -u> <-word | -w> <-all | -a>	Searches for the specified target string in the code and replaces it with the specified content. These arguments are available. ■ `-case` | `-c` specifies that the search is case sensitive. By default, the case is ignored. ■ `-up` | `-u` searches up. By default, the search goes down the file. ■ `-word` | `-w` searches for whole word matches. By default, the search does not check for word boundaries. ■ `-all` | `-a` replaces all occurrence of the target string.

Command Name	Description
RESET	Resets the SAS Studio session.
SAVE <*path-to-parent-folder*> <*filename.ext*> <-force> <-encoding>	Opens the Save As window, if no options are specified.. This window does not open if you specify a filename. To save a specific file, enter the folder name.For more information, see "Specifying the Path to the Content: SAS Content or SAS Compute" on page 291. If no filename is specified, the file keeps its original filename. To change the filename, you must specify the `filename.ext` argument. You can also specify these options: ■ `-force` overwrites an existing file with the same name. ■ `-encoding` specifies the encoding of the file, such as `-UTF8`.
SAVEALL	Saves all open files.
SEARCH <-max | -restore | -off>	Displays the **Search** tab. These options are available: ■ `-max` maximizes the preview area. ■ `-restore` restores the preview area. ■ `-off` closes the **Search** tab.
SHOWLOG <-max | -restore | -off>	Displays the **Log** tab. These options are available: ■ `-max` maximizes the preview area. ■ `-restore` restores the preview area. ■ `-off` closes the **Log** tab.
SHOWRESULTS <-max | -restore | -off>	Displays the **Results** tab. These options are available: ■ `-max` maximizes the preview area. ■ `-restore` restores the preview area. ■ `-off` closes the **Results** tab.
SUBMISSIONS <-max | -restore | -off>	Displays the **Submissions** tab. These options are available: ■ `-max` maximizes the preview area. ■ `-restore` restores the preview area. ■ `-off` closes the **Submissions** tab.
SUBMIT	Runs the currently selected code or all the code in the editor if no code is selected.
VIEWTABLE <*library.table*> <-last>	Opens the Select Table window. Use the `library.table` argument to specify the table to view.

Command Name	Description
	If no table is specified, use the `-last` option to open the last table that was generated in the current session.
ZOOM <-max><-restore>	Specifies the size of the tab area. These arguments are available. ■ `-max` specifies to maximize the size of the tab area. ■ `-restore` returns the tab area to the default size.

Specifying the Path to the Content: SAS Content or SAS Compute

When specifying the file path for these commands, you must prefix the directory path argument with sascontent or sascompute to specify where the file or folder is located.

- EXPORTTABLE
- OPENFILE
- OPENENCODED
- SAVE
- SAVECODE
- SAVELOG
- SAVERESULTS
- SAVESUMMARY
- SAVEPACKAGE

Here are some examples using the EXPORTTABLE command:

```
exporttable "sascontent:/Users/userID/My Folder/Import" "air.csv"
exporttable "sascompute:/your-server.com/userID/SAS Studio Content/Import" "air.csv"
```

Appendix 6

Transitioning to SAS Studio

Tips for SAS Studio 3.x Users

Overview of Differences

SAS Studio 2020.1 includes new flow functionality and an updated and simplified user interface with many new and existing options now available from an overflow menu. In most cases, your tasks, queries, and jobs work the same in SAS Studio 2020.1.

However, default file storage has changed significantly. The default file storage location is SAS Content, and administrators must configure access to the file system.

Process Flow Updates

Process flows were available in SAS Studio 3.x and enabled you to create a repeatable process flow based on references to specific columns, tables, tasks, and programs. Starting with SAS Studio 2020.1, a flow is a metadata-driven design tool that supports a sequence of operations on data by using nodes to represent the data and operations. Nodes are created by adding "steps" to your flow.

Flows are designed to enable you to manage complex processes by creating a generalized flow of data without hard-coding references to tables, columns, and operations ahead of time. You can update the specific attributes of any node in the flow by using the node properties. Ports represent the input source and output target of an operation node and can be accessed programmatically by the macro variables that are associated with each port. Nodes and ports can be used to provide metadata-based definitions so that you have greater flexibility in defining the content of your flow.

The types of nodes that are available to use in a flow can be accessed from the **Steps** section of the navigation pane. Additional steps are added to the **Steps** section with each new release of SAS Studio. For more information, see “Understanding Nodes” on page 106.

You cannot currently convert saved SAS Studio 3.x process flows to SAS Studio 2020.1 flows, but you can create a SAS program from the nodes in a process flow. For more information, see “Generating Code from a Process Flow” in *SAS Studio 3.8: User’s Guide*.

File Storage

For SAS Studio 2020.1 users, the default file storage location is SAS Content. SAS Content locations are integrated with other SAS Viya applications, such as SAS

Drive, so that you can seamlessly share files among the applications and with other SAS users.

If you also need access to a file system to perform other tasks, such as working with a Git repository, your Kubernetes and SAS Studio administrators can create access to the file system during deployment. In SAS Studio 5.x, by default, you had access to both SAS Content and file system storage locations. For more information, see "Creating Persistent File Storage" in *SAS Studio: Administrator's Guide*.

Queries

In SAS Studio 2020.1, you can create both stand-alone and flow queries, and you can now add calculated columns that are based on other columns and values to a query. For more information, see "Creating a Query in a Flow" on page 141.

Saved queries from earlier versions of SAS Studio can be run in SAS Studio 2020.1 if the following criteria are met:

- The data that was used in the original query is available in SAS Studio 2020.1.
- The data that is available in SAS Studio 2020.1 is not missing any columns that were used in the original query.
- All tables that are included in the original query must have a defined join.

If a saved query cannot be migrated, SAS Studio converts the query to a program file and includes a note in the program that explains why the query could not be migrated.

Tasks

You can use all tasks, including any custom tasks, that were available in SAS Studio 3.x. You can also create custom tasks in SAS Studio 2020.1. For more information, see "Create a Custom Task" on page 255.

You cannot include a task as a step in a flow in SAS Studio 2020.1. However, you can use the **Code to Flow** button on the task menu to copy the code that is automatically generated by a task to a new SAS Program node in an open flow. For more information, see "Copying Code to a Flow" on page 117.

Importing Data

Your saved import task files (*.CTL) still work in SAS Studio 2020.1. If you are importing data as part of a flow, use the new Import step. For more information, see "Importing Data from an External File" on page 113.

Preferences

Between SAS Studio 3.x and SAS Studio on Viya, the content of the Preferences window was reorganized. Some preferences, such as the preferences for the code editor, are common options and are not specific to SAS Studio.

If you are upgrading from SAS Studio 3.7 or earlier, the default value for the **SAS variable name policy** has changed to **ANY**. As a result, by default, the variable names can begin with or contain any characters, including blanks, must contain at least one character, and cannot contain any null bytes.

Configuration Properties

Your SAS Studio administrator can use configuration properties to customize your software. In SAS Studio 3.x, your administrator might have specified these customizations in the `config.properties` file. When you re-install or reconfigure SAS Studio, any customizations in the `config.properties` file are lost. Before you upgrade to a new release of SAS Studio or reconfigure SAS Studio, record any changes that you made to the `config.properties` file. After you re-install or reconfigure SAS Studio, you must reset these configuration properties in SAS Environment Manager.

The configuration properties in SAS Studio 3.x were prefaced with `webdms`. Some of these configuration properties are not used in the SAS Viya environment. Also, in SAS Studio on Viya, configuration properties are prefaced with `sas.studio`. For a list of the available configuration properties, see "Configuration Properties for SAS Studio" in *SAS Studio: Administrator's Guide*.

Themes

SAS Studio 2020.1 uses the same default corporate theme as all SAS Viya 2020.1 products. For more information, see "Creating Global Settings" on page 258.

General User Interface and Feature Differences

Some of the options for SAS Studio features have been moved from the toolbar for that feature to an overflow menu (represented by ⋮ on the toolbar) in order to simplify the user interface.

Here are some of the new features that were added after SAS Studio 3.x:

- DATA Step debugger. For more information, see "Using the DATA Step Debugger" on page 45.

- Expression Builder. For more information, see “Building an Expression” on page 273.
- Jobs definitions and scheduling. For more information, see SAS Studio Developer’s Guide: Working with Jobs.
- Console window. For more information, see “Using the Console” on page 18.
- Command line interface. For more information, see “About the Command Line Interface” on page 281.
- Interactive perspective. For more information, see “Understanding Perspectives” on page 25.
- FedSQL queries. For more information, see “Generating a FedSQL Query” on page 98.
- Manage keyboard shortcuts. For more information, see “Managing Keyboard Shortcuts” on page 29.
- Open Files section of the navigation pane. For more information, see “Accessing Your Open Files” on page 5.
- Git branching and merging. For more information, see Chapter 8, “Understanding Git Integration in SAS Studio,” on page 219.

Tips for SAS Enterprise Guide Users

Overview of Differences

Both SAS Studio and SAS Enterprise Guide provide a point-and-click interface to SAS that enables you to create reports, graphs, and charts; access SAS servers and data; and analyze data. Both products also include ready-to-use tasks for analysis and reporting, a color-coded SAS language editor, and access to some of the most common features of Git.

The primary difference between SAS Studio and SAS Enterprise Guide is that SAS Studio is a tool that you can use to write and run SAS code through your web browser. SAS Enterprise Guide is a Microsoft Windows client application that you install on your machine.

Using the SAS 9 Content Assessment Tool

SAS 9 Content Assessment is a collection of applications that is designed to help you understand various characteristics of your SAS 9.4 system. Each application examines your SAS®9 system for relevant information, gathers key details, and produces results from each part of the assessment.

You should run the SAS 9 Content Assessment tool before you upgrade to a new release. This tool identifies any problems that you might encounter when you

upgrade. For more information, see "Executing the importEGProjects Application" in *SAS Content Assessment*.

Opening a SAS Enterprise Guide Project in SAS Studio

SAS Enterprise Guide projects cannot currently be directly opened in SAS Studio, but you can use the following options for transitioning your work in a SAS Enterprise Guide project to SAS Studio 2020.1:

- If you have administrator privileges, you can use SAS Environment Manager to import your SAS Enterprise Guide project and create a SAS Studio flow file (*.flw) for each process flow in your project. All executable nodes in the SAS Enterprise Guide process flow, except for externally referenced programs (*.sas), are converted to SAS Program nodes in a SAS Studio flow. Links between nodes are maintained. For more information, see "SAS Enterprise Guide" in *SAS Viya: Content Migration From SAS 9.4*.
- Export the code from your project and open the code in SAS Studio 2020.1. For more information, see "Export All Code" in *SAS Enterprise Guide: User's Guide*.
- Convert the project to a SAS Studio 3.8 process flow, and then create a SAS program from the process flow. For information about converting a project to a process flow, see "Generating Code from a Process Flow" in *SAS Studio 3.8: User's Guide*.

Note:

- You might need to update references to data in your SAS Enterprise Guide project that are not accessible in SAS Studio.
- When you begin working in SAS Studio with code that you developed or converted from SAS Enterprise Guide, you need to evaluate and update the code to make sure that it takes advantage of the new CAS programming language. For more information, see Programming Documentation for SAS Viya.

Process Flow Differences

Process flows are available in SAS Enterprise Guide and are automatically created when you create a project in SAS Enterprise Guide. You can also create multiple additional process flows in your project. Process flows in SAS Enterprise Guide enable you to show the relationship between two or more objects that are based on references to specific columns, tables, tasks, and programs.

SAS Studio 2020.1 includes flows, which are a metadata-driven design tool that supports a sequence of operations on data by using nodes to represent the data and operations. Nodes are created by adding "steps" to your flow. Flows are designed to enable you to manage complex processes by creating a generalized flow of data without hard-coding references to tables, columns, and operations ahead of time. You can update the specific attributes of any node in the flow by

using the node properties. Ports represent the input source and output target of an operation node and can be accessed programmatically by the macro variables that are associated with each port. Nodes and ports can be used to provide metadata-based definitions so that you have greater flexibility in defining the content of your flow.

The types of nodes that are available to use in a flow can be accessed from the **Steps** section of the navigation pane. Additional steps are added to the **Steps** section with each new release of SAS Studio. For more information, see "What Is a Flow?" on page 102.

Query Differences

Queries in SAS Enterprise Guide and SAS Studio offer much of the same functionality. Queries in both products can be based on one or more tables and can include computed, or calculated, columns that are based on other columns or values. In SAS Enterprise Guide, you use the Query Builder to create a query as part of a project. In SAS Studio 2020.1, you can create both stand-alone and flow queries, and you can choose to create your query by using PROC FEDSQL instead of PROC SQL. For more information, see "What Is a Query?" on page 70.

Importing Data

Both SAS Enterprise Guide and SAS Studio enable you to create SAS data sets by importing text, CSV, Microsoft Excel, and many other PC-based database files. SAS Enterprise Guide uses the Import Data wizard to import files. SAS Studio has two types of import functionality. The quick import enables you to quickly import data from the file system or from SAS Content by selecting a few options. You can also import data into a flow. This import functionality is more robust and provides additional options. For more information, see "About Importing Data to SAS Studio" on page 204.

Task Differences

Both SAS Enterprise Guide and SAS Studio contain SAS tasks, which are graphical interfaces to SAS procedures from simple data listings to complex analytical procedures. In SAS Enterprise Guide, you could have these types of tasks:

- predefined tasks that ship with SAS Enterprise Guide.
- SAS Studio tasks. If you have SAS Enterprise Guide and SAS Studio at your site, you can access SAS Studio tasks in SAS Enterprise Guide.
- custom tasks. You can develop custom tasks for your site and make these available in SAS Enterprise Guide.

The underlying technology for the SAS Enterprise Guide tasks and SAS Studio tasks differs, so you cannot open a SAS Enterprise Guide task by using the task interface in SAS Studio.

To transition a predefined task, a SAS Studio task, or a custom task from SAS Enterprise Guide to SAS Studio:

1 In SAS Enterprise Guide, export the SAS code.

2 In SAS Studio, open the exported SAS code.

Tips for SAS Data Integration Studio Users

SAS Data Integration Studio and SAS Studio

The data management features in SAS Studio are not currently a direct replacement for SAS Data Integration Studio. More data management features are made available with each new release of SAS Studio.

Creating Flows

SAS Studio flows are like SAS Data Integration Studio jobs. Flows are built from a sequence of data and operation nodes, which you can connect in order to specify the order of execution. For example, the next figure shows a flow that selects the number of home runs for each player in a table of baseball data.

Figure A6.1 *Flow That Selects Home Run Data*

- BASEBALL is a Table node that specifies an input table of data about baseball teams.
- Select Home Runs is a Query node that selects the number of home runs for each player in the BASEBALL table.
- Home Runs is a Table node that specifies the output table for the query.

For more information about working with flows, see the following topics:

- “Creating a Flow” on page 106
- “Opening a Flow” on page 106
- “Understanding the Flow Tab” on page 103

- “Running a Flow” on page 143

You cannot currently convert SAS Data Integration Studio jobs to SAS Studio flows.

Registering Libraries and Files

SAS data sets can be sources or targets in a flow. The **Libraries** section of the navigation pane enables you to access the SAS libraries that are available to you. For more information, see “Working with Libraries” on page 8.

Flows are saved as files. The **Explorer** section in the navigation pane enables you to access files and folders from your folder shortcuts, your server file system, and your SAS Content Server locations. For more information, see “Using the Explorer ” on page 5.

Using Nodes in a Flow

Nodes in a SAS Studio flow are like transformations in SAS Data Integration Studio jobs. Flow nodes can be accessed from the **Steps** section of the navigation pane. Only nodes in that section can be used in a flow. The following node types are included in SAS Studio Basic:

- The Export node enables you to save data to a delimited file (*.csv, *.dlm, *,tab, *.tsv), a text file, or a Microsoft Excel file (*.xls, *.xlsx). For more information, see “Exporting Data to an External File” on page 110.
- The File node specifies an external source file in a flow. For more information, see “Importing Data from an External File” on page 113.
- The Import node converts an external file to a SAS data set in a flow. For more information, see “Importing Data from an External File” on page 113.
- The Table node references a SAS data set in a flow. For more information, see “Adding a Table from a SAS Library to a Flow” on page 114.
- The SAS Program node specifies a new SAS program, a saved SAS program, or a snippet in a flow. For more information, see “Working with Code in a Flow” on page 115.
- The Query node extracts data from one or more tables in a flow according to criteria that you specify. For more information, see “Creating a Query in a Flow” on page 141.
- The Sort node orders output data in a flow by the values of one or more source columns. For more information, see “Sorting Data” on page 141.

You can use the node properties under the flow canvas to specify the attributes and content of the node, as explained in the preceding topics.