SAS® Studio: Working with Flows

2023.02*

* 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
February 15, 2023

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2023. *SAS® Studio: Working with Flows*. Cary, NC: SAS Institute Inc.

SAS® Studio: Working with Flows

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

February 2023

v_004-P1:webeditorflows

Contents

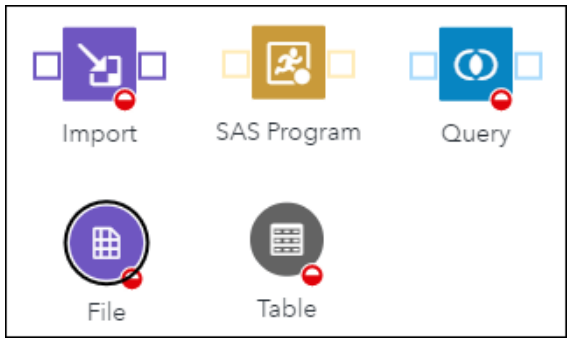
1

Introduction to Flows

What Is a Flow?

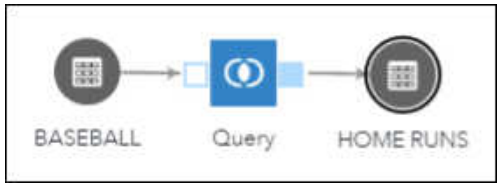
A *flow* is a sequence of operations on data. Data and operations are represented in SAS Studio by steps that you can access from the **Steps** section of the navigation pane. Each step in a flow is represented by a node on the flow canvas.

The nodes on this flow canvas represent some of the steps that are available in SAS Studio.



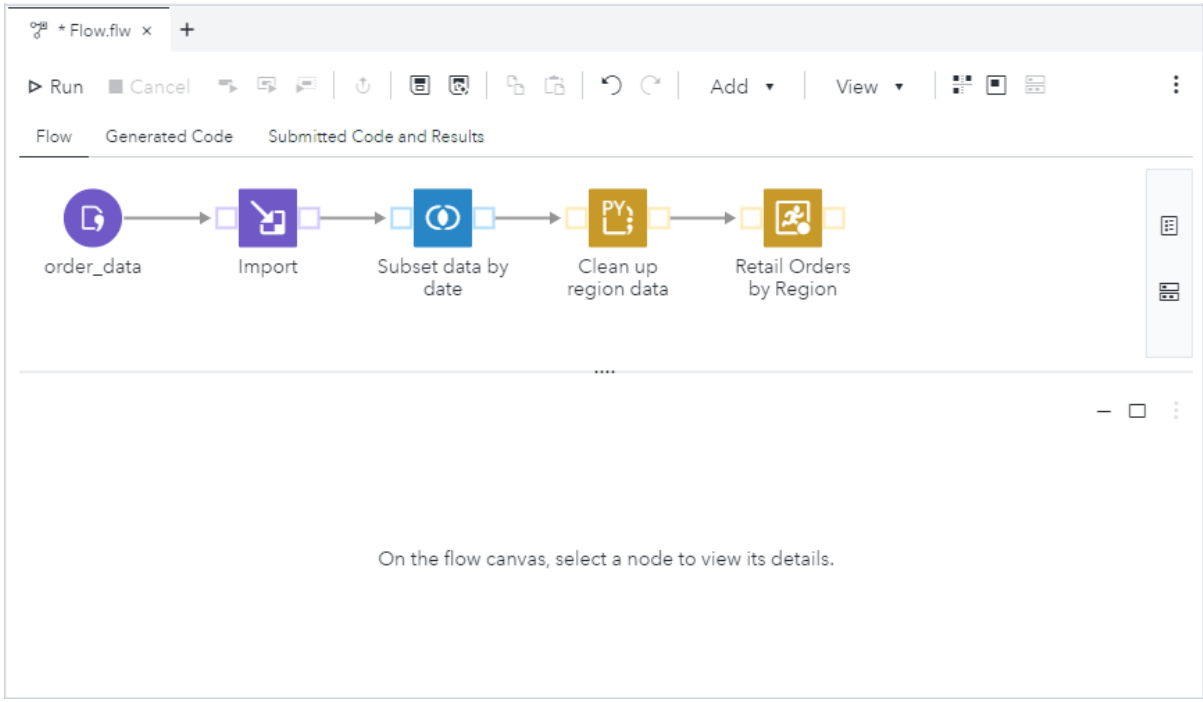
A flow can include a series of nodes in which the output of one node is the input to another node. As you build a flow, SAS Studio automatically generates SAS code for each node. You can use flows to prepare data for reporting and analysis.

This example uses the Table and Query steps to create a flow that analyzes baseball data.



- BASEBALL is a Table node that provides the input data to the query.
- 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.

You can also use a flow to combine different technologies in a single workflow. For example, you can create a flow that uses the Import step to convert an external .csv file to a SAS table, the Query step to subset the data, the Python Program step to cleanse the data, and the SAS Program step to create a report.

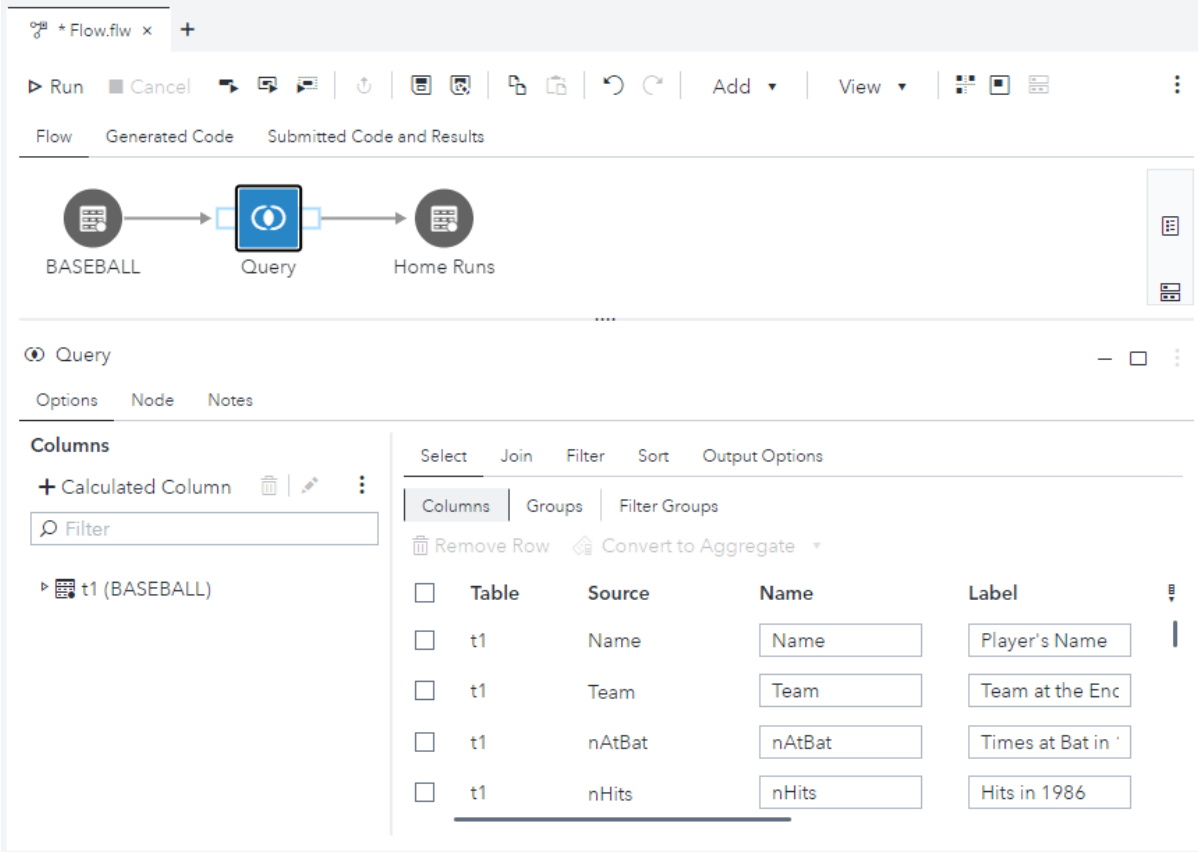


The types of steps that you can add to a flow are listed in the **Steps** section of the navigation pane. For more information, see "What Kind of Steps Can I Add to a Flow?" on page 9.

Understanding the Flow Tab

About the Flow Tab

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



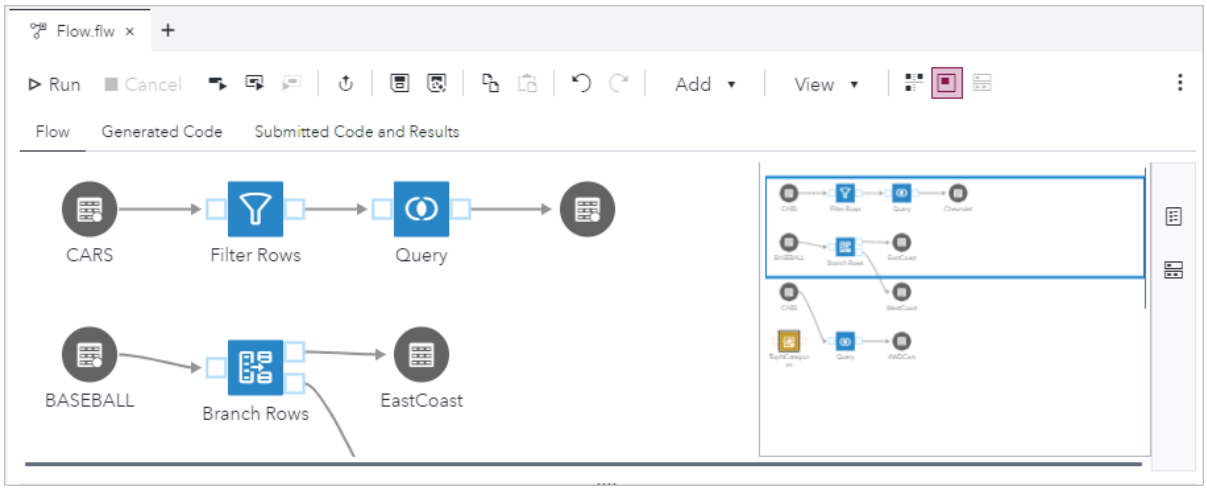
The Flow tab has two areas:

- 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 **Submitted Code and Results** tabs to view the code and log that SAS Studio automatically generates as you build the flow. The **Submitted Code and Results** 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.

Using the Overview Map

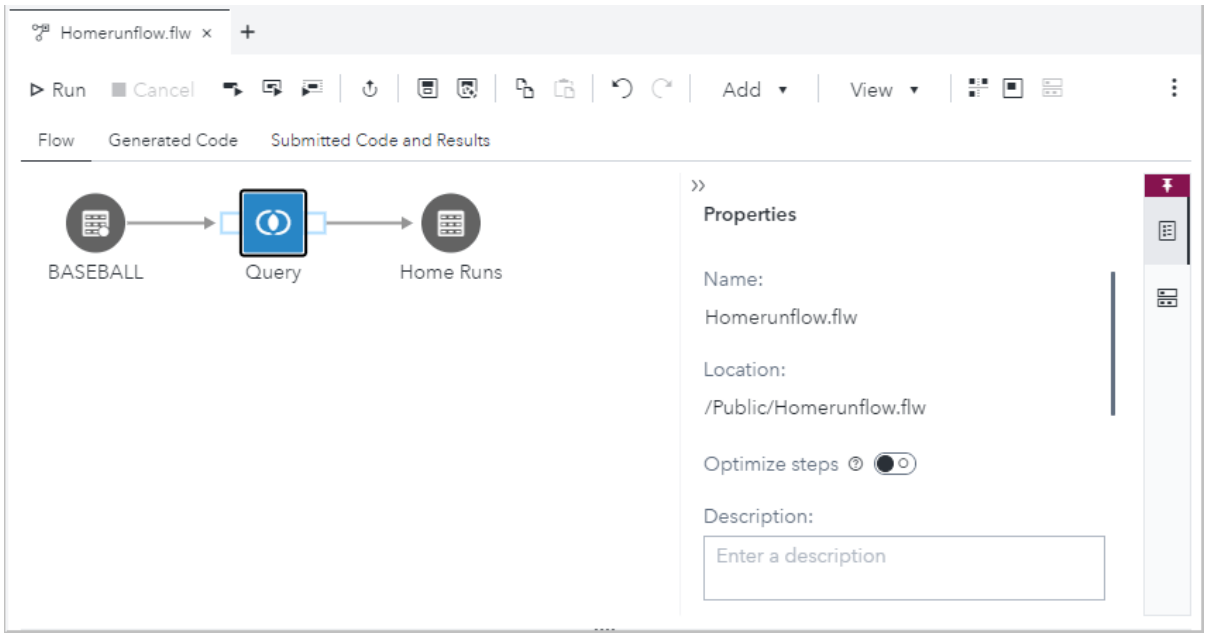
You can use the overview map to view your entire flow in a small window on the flow canvas. This feature is useful when you have a large, complex flow that you must scroll to view. Drag the rectangle on the map to move the view vertically and horizontally.

To toggle the overview map on and off, click ▣ on the flow toolbar.



Viewing Flow Properties

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.

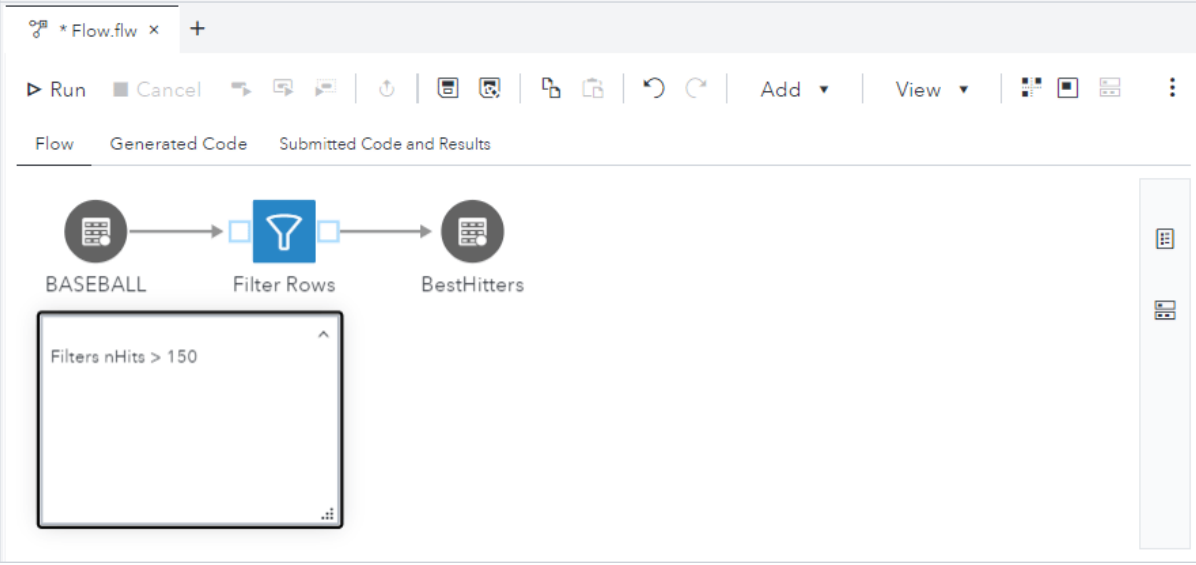
Adding a Note to a Flow

You can annotate your flow by adding notes to the flow canvas. Notes that are added to the flow are not associated with a specific node. You can position the notes anywhere in the flow, and you can cut, copy, and paste notes within and between flows.

To add a note to a flow:

1. On the flow toolbar, click **Add** ⇨ **Notes**.
2. Enter the text of the note in the note box.

Note: To delete a note, right-click the note and select **Delete**.



You can collapse and expand a note in a flow:

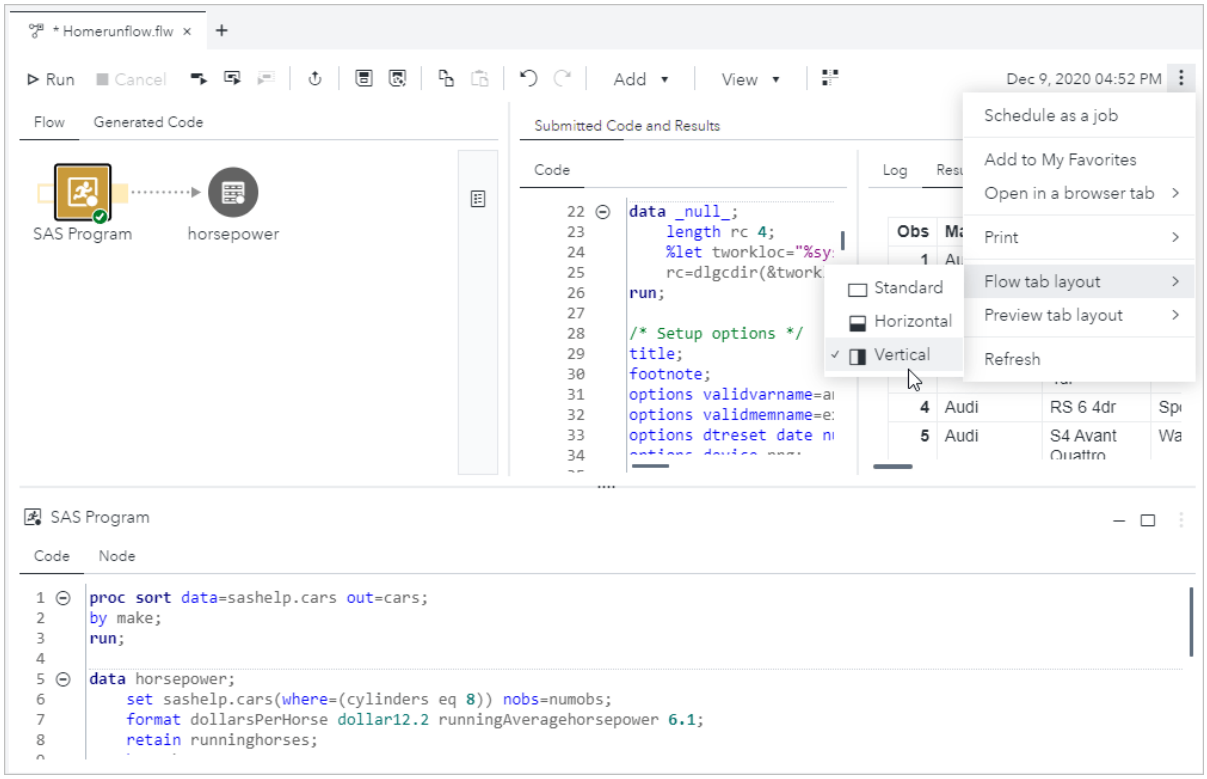
- To collapse a note in the flow, click ^ in the upper right corner of the note.
- To expand a collapsed note, double-click the note.

Customizing the Flow Tab

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

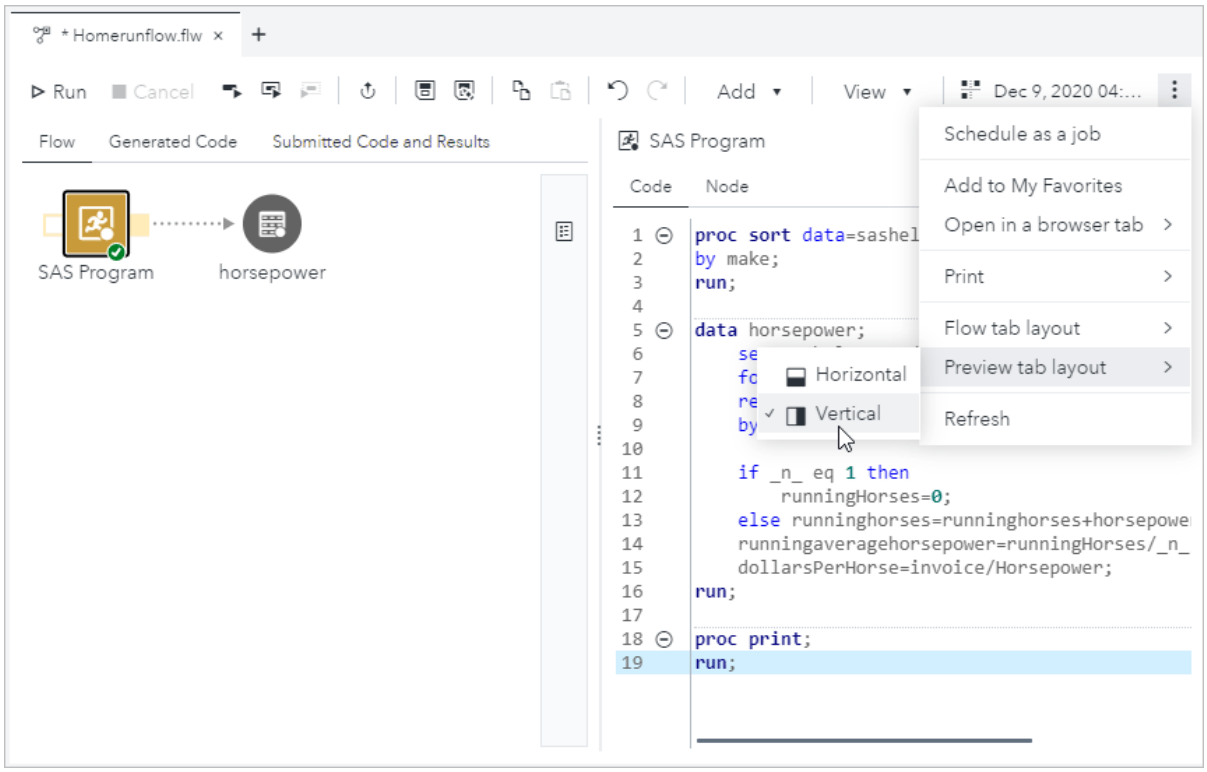
To change the layout of the default flow tabs:

- 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 Steps and Nodes

About Steps and Flow Nodes

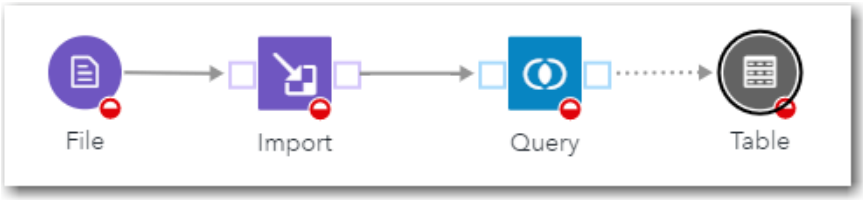
SAS Studio is shipped with many predefined steps that include queries and data transformations. Steps are represented as nodes in a flow. Flows are built from a sequence of data and operational nodes, which you can connect in order to specify the order of execution.

Data steps, such as the File and Table steps, represent data in a flow. Operational steps, such as the Import, Query, and SAS Program steps, represent operations that can be performed on data. The File and Table steps are not considered operational steps.

You can add steps 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 step, a Query step, and a SAS Table step. 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.

What Kind of Steps Can I Add to a Flow?

The types of steps that you can add to a flow are listed in the **Steps** section of the navigation pane. The steps are organized into categories that indicate the function they perform. 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.
- Python Program - enables you to write a new Python program or open a saved Python program.
- 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 details 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.

For a complete list of the steps that are available in each license of SAS Studio, see "Summary of Flow Functionality" in *SAS Studio: User's Guide*.

Note: Not all steps are available in all licenses of SAS Studio.

Adding Steps 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 steps to a flow.

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)
- Python program files (*.py)
- Comma-separated values files (*.csv)
- Delimited files (*.tsv, *.tab, or *.dlm)
- Text files (*.txt)
- Flow files (*.flw)

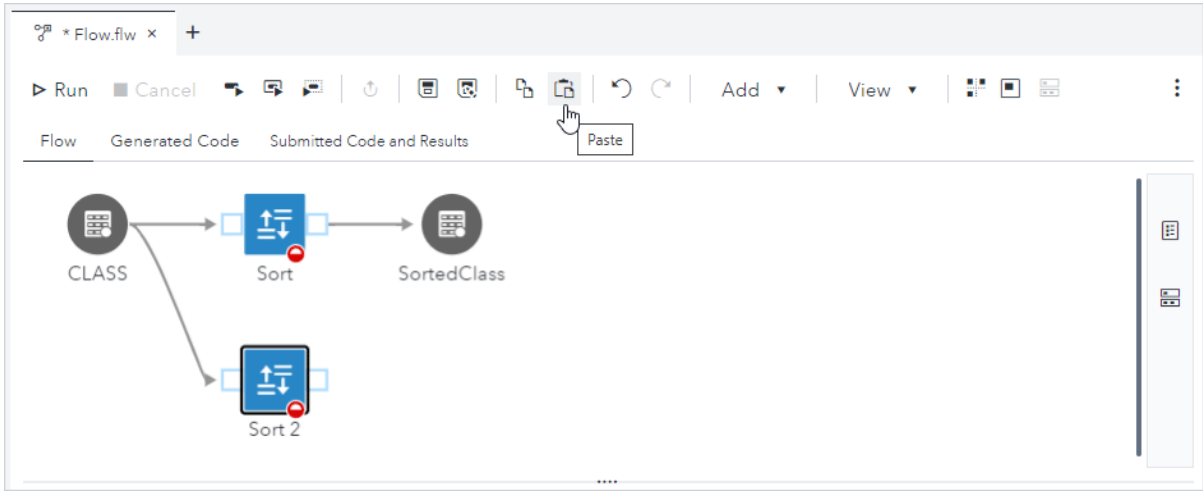
Note: The subflow feature is available only if your site licenses SAS Studio Analyst. For more information, see “About Subflows” on page 51.

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

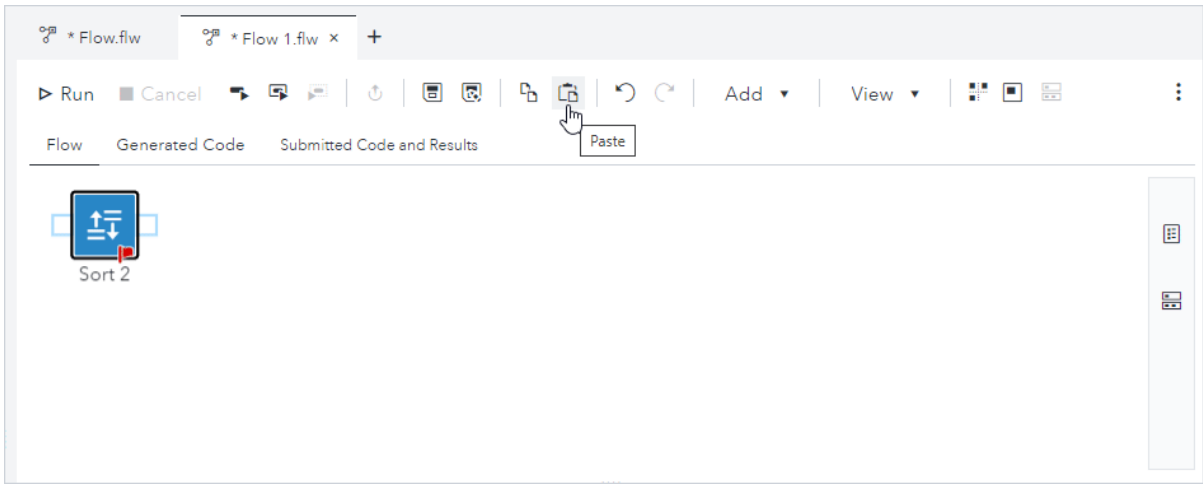
Cutting, Copying, and Pasting Nodes

You can cut, copy, and paste one or more nodes within a flow. You can also cut, copy, and paste nodes between flows. When you cut or copy nodes, the properties and values that are associated with the nodes are retained.

When you copy and paste an operational node within a flow, any connections to the input port of the original node are automatically created to the input port of the pasted node. You must manually re-create any connections from the output port of the pasted node.



When you copy and paste an operational node into a different flow, connections are not automatically created unless you have also selected and copied the connected nodes. The pasted node displays an error state until you add the required connections and update the node options as necessary.



To copy one or more nodes, select the nodes that you want to copy from the flow canvas and click .

To cut one or more nodes, select the nodes that you want to cut from the flow canvas. Right-click a selected node and select **Cut**.

To paste the nodes, open the flow into which you want to paste the nodes and click .

> **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.

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.

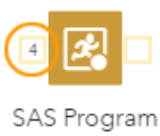
Expanding and Collapsing Node Ports

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.

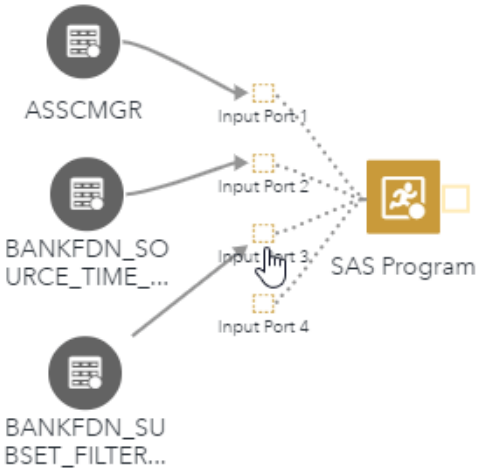
By default, ports are collapsed. To expand or collapse a port, right-click the port and select **Expand** or **Collapse**. You can expand or collapse all ports by clicking **View** on the toolbar and selecting **Expand all ports** or **Collapse all ports**.

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.



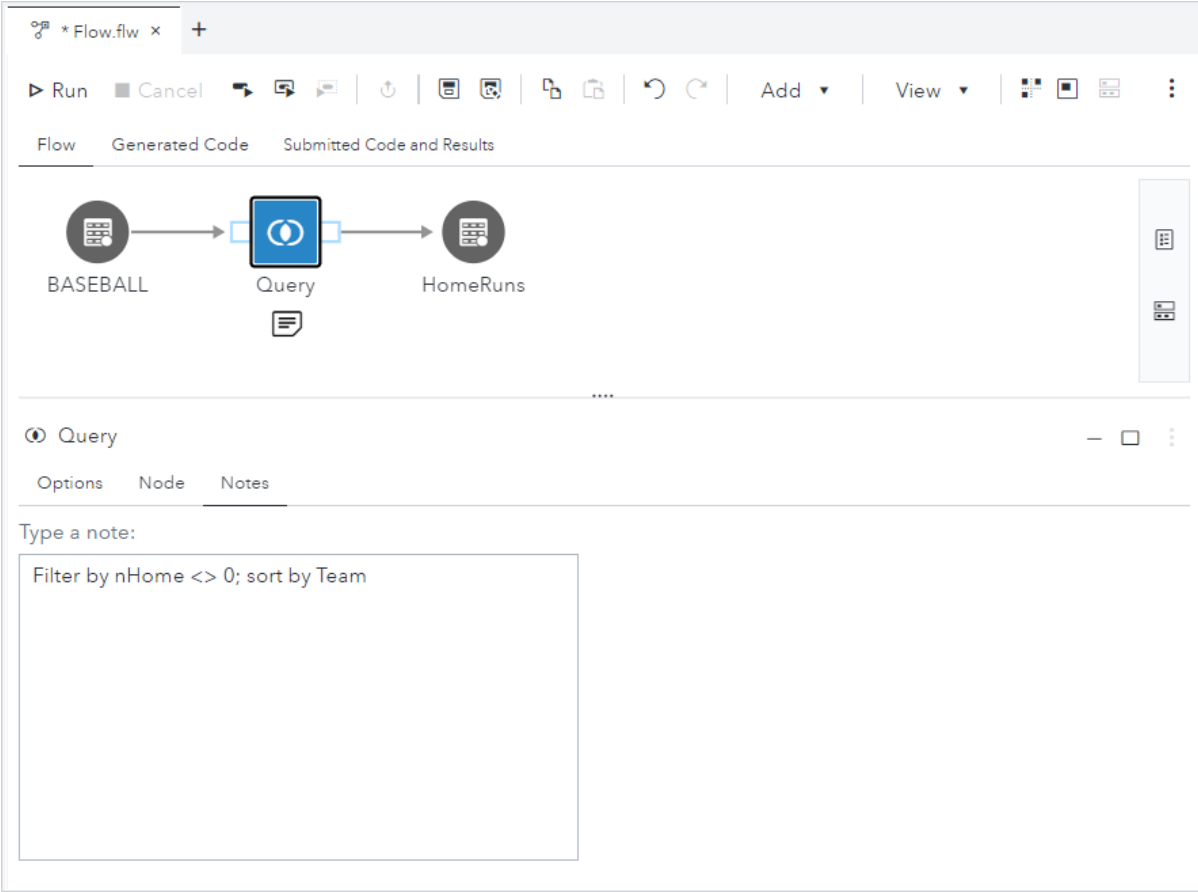
Note: When a node has multiple output ports, you must first expand the ports before creating a connection to another node.

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.

Adding Notes to a Flow Node

You can add notes to a specific node in a flow by using the Note tab in the node details. When you add a note to a node, a note icon is displayed with the node on the flow canvas.



To add a note to a node:

1 Click the appropriate node on the flow canvas.

2 Click the **Notes** tab in the node details area and enter the note text.

You can edit a note in these ways:

- Click the **Notes** tab in the node details area and update the note text.
- Click the note icon on the flow canvas to expand the note text, and then double-click the note text box. Enter your changes in the note text box. Click anywhere on the flow canvas to save your changes.

To expand a node note, single-click the note icon.

To collapse a node note, click ^ in the upper right corner of the note.

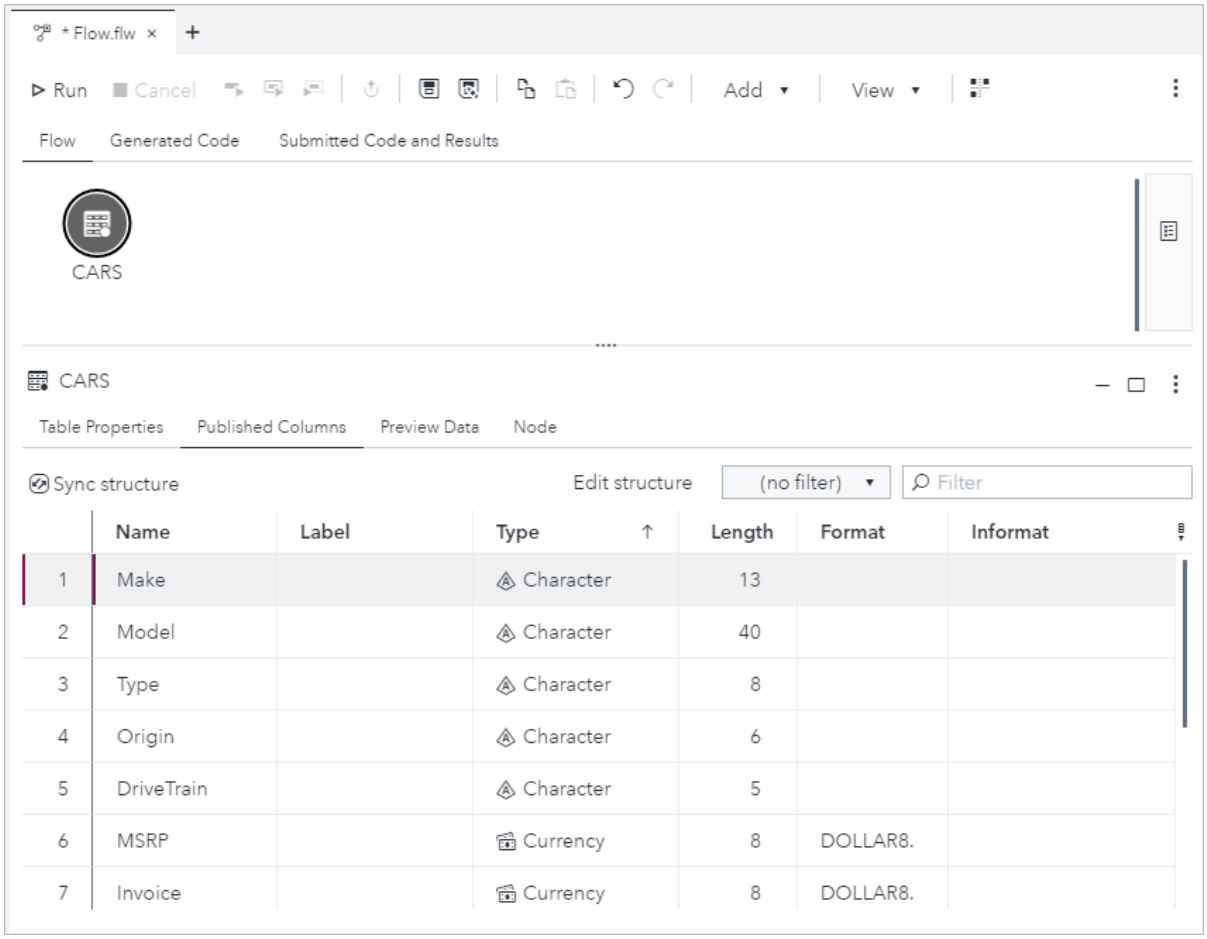
To remove a note from a node, delete the text in the note.

2

Working with Data in a Flow

About the Table Node

Table nodes are used to reference SAS data sets in SAS libraries or CAS tables in CAS libraries. You can use table nodes as input and output for operational nodes.



The column structure of the table is displayed on the Published Columns tab of the node details.

- To filter the columns on the tab by data type, click the filter drop-down list and select the filter criteria. You can also enter filter criteria for the Name and Label columns in the filter box.
- To add or delete columns or change the properties of an existing column, click **Edit structure**. To exit edit mode and reset the column structure back to the structure of the table that is specified on the Table Properties tab, click **Edit structure** again. Any changes that you have made to the column structure are overwritten.
- To synchronize the column structure with the structure of the table that is specified on the Table Properties tab, click **Sync structure**. This option is not available when you are in edit mode.

TIP You can use the **Sync structure** and **Edit structure** options to copy the column structure of an existing table to a new table.

1 Click the table node to which you want to copy a new column structure, and then click the **Table Properties** tab.

Note: Before you can copy the column structure of an existing table, you must specify a library and table name for the new table (for example, Work.NewTable).

2 Change the values of the **Library** and **Table name** fields to match the library and table names of the table whose structure you want to copy (for example, Sashelp.Class).

3 Click the **Published Columns** tab. Click **Sync structure** to synchronize the column structure of the new table with the existing table, and then click **Edit structure**. You can add, delete, or edit columns, if necessary.

4 Click the **Table Properties** tab again and change the values of the **Library** and **Table name** fields back to the values of the original table (for example, Work.NewTable).

You can view the data in the table on the Preview Data tab. The Preview Data tab interface is very similar to the interface for the stand-alone table viewer. For more information, see "About the Table Viewer" in *SAS Studio: User's Guide* and other topics in the "Working with Data" chapter.

Specifying Advanced Output Table Options

You can specify additional options to apply when an output table is created or when data is updated or inserted in the table. Often, the additional options are used to improve performance. The advanced table options can be applied to tables in these situations:

- Table node that is connected to the output port of an operational node – advanced options for a Table node can include options that are associated with creating a table. For example, you could use additional options to specify a label, specify how observations in the output table are compressed, or you could create single and composite indexes for a SAS output table.

  Note: The advanced output table options are not available if the Table node is connected to the output port of a SAS Program, Python Program, Insert Rows, Execute Decisions, or custom step node.

- Target table that is defined in the node details of an operational node – advanced options for the target table in an operational node can include options

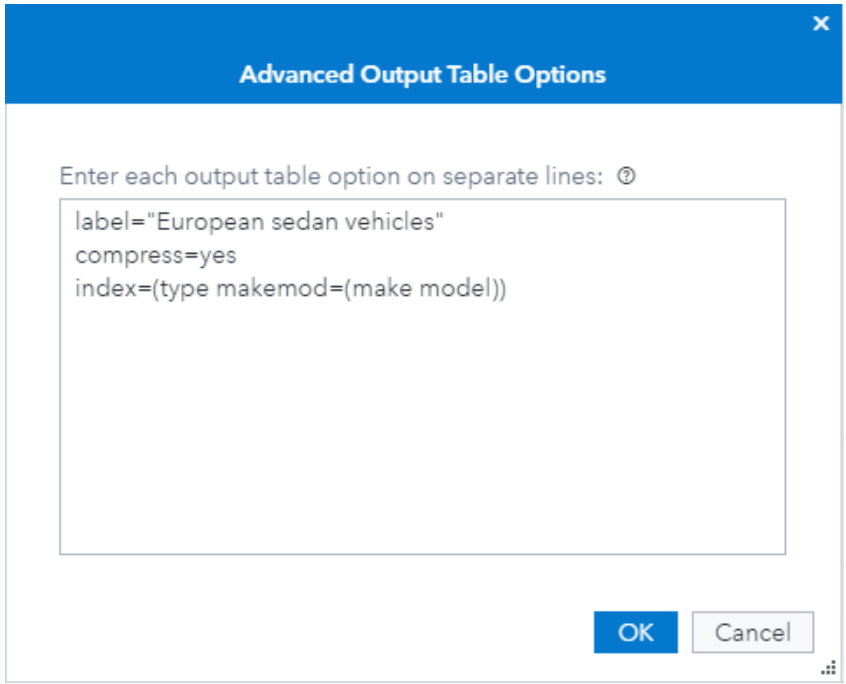
that are associated with the operations performed by the node on the table. For example, in a Load Table node, you could use options that are associated with inserting and updating data in a table, such as options to specify the number of rows to insert or to specify whether to use a DBMS-specific bulk-load facility.

To specify advanced table options:

1 If you are specifying options for a Table node, click the output table in the flow, and then click the **Table Properties** tab.

If you are specifying options for the target table in an operational node, click the operational node and then click the **Options** tab.

2 Expand **Output Table Options**, and click **Advanced table options**. Enter each option on a separate line. This example shows options for a Table node.



The syntax for the advanced options can vary depending on the code that is generated for the step. In most cases, the output options should be data set options, but some steps require CASL table parameters or FedSQL table options. You must include any quotation marks that are required for string values. You can find more information about options here:

- SAS data set options
- CASL table parameters
- FedSQL table options

Note: This is an option for advanced users. It is recommended that you avoid using any options that might change the column metadata, such as DROP=, KEEP=, or RENAME=. If you use options that change the column metadata, you can generate syntactically valid code that can cause errors in downstream nodes. If you need to subset columns in the table or change column names or labels, you should use an additional step such as Manage Columns.

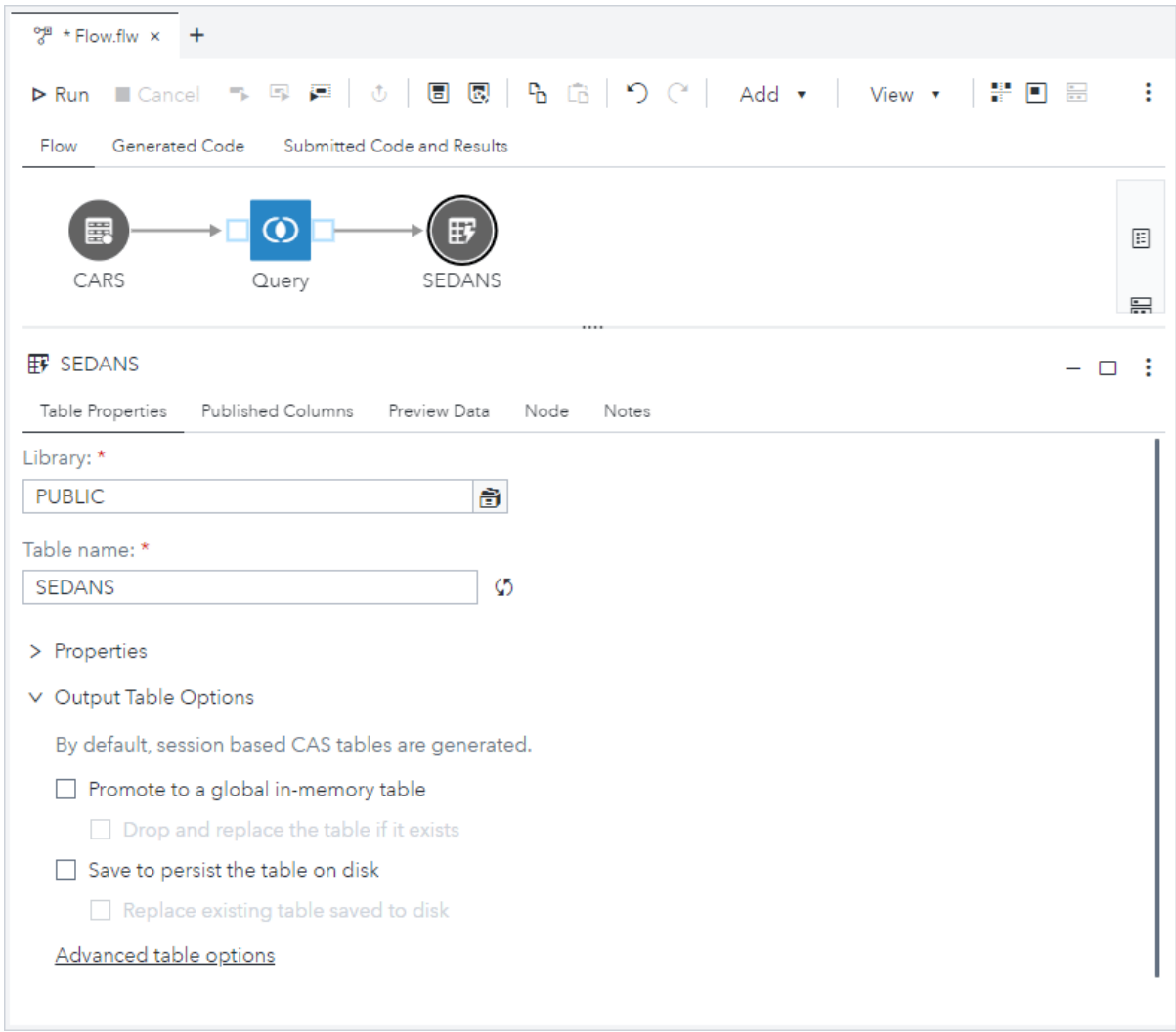
Setting Options for CAS Output Data

If you are running an operational node that creates CAS output data, you can specify whether the table is a session-scope table or a global-scope table, and you can save the table to the CAS server. You can also specify additional options to apply when the table is created or updated.

Note: The output table options are not available if the table node is connected to the output port of a SAS Program, Python Program, Insert Rows, or custom step node.

To specify CAS output table options:

1 Click the output table in the flow, and then click the **Table Properties** tab. Expand **Output Table Options**.



2 Select from the following options:

- **Promote to a global in-memory table** - creates a global-scope table that can be viewed by other sessions. Global-scope tables persist after the current CAS session is terminated. If you want to automatically replace an existing global-scope table of the same name, select **Drop and replace the table if it exists**. If you do not create a global-scope table, a session-scope table is created. A session-scope table can be viewed only by your current CAS session and is dropped when the session ends. Session-scope tables are created by default.
- **Save to persist the table on disk** - saves the table to the CAS server as a SASHDAT file. If you want to automatically replace an existing table of the same name, select **Replace existing table saved to disk**.

3 To specify additional options to apply when the table is created or updated, click **Advanced table options** and enter each option on a separate line. For more information, see "Specifying Advanced Output Table Options" on page 17.

Exporting Data to an External File

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.

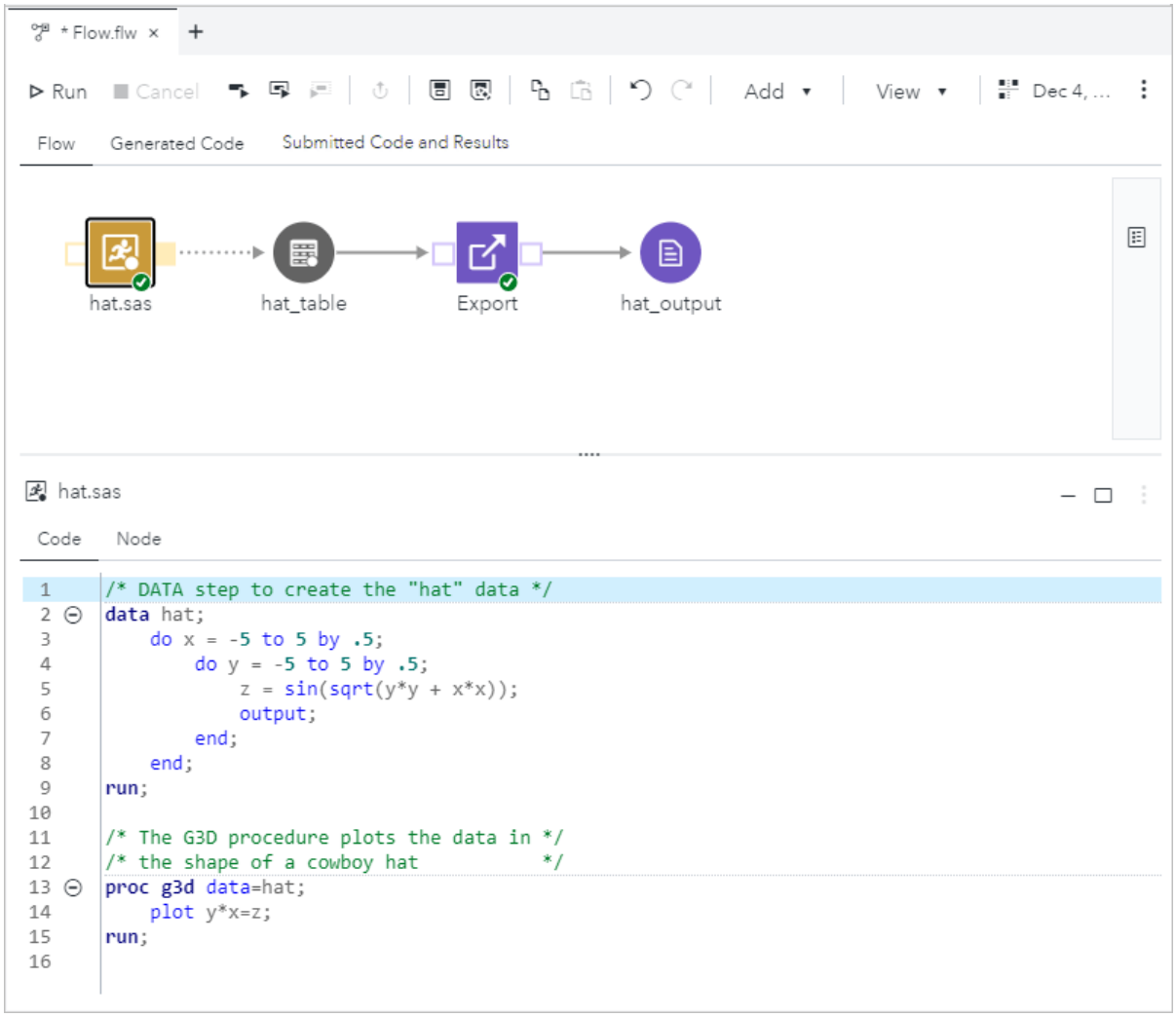
3 Connect the input port of the Export node to a data source, such as a Table node or another node that creates an output table, such as a Query node or an Import node. For more information, see "Connecting Nodes" on page 12.

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 21.

> **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:

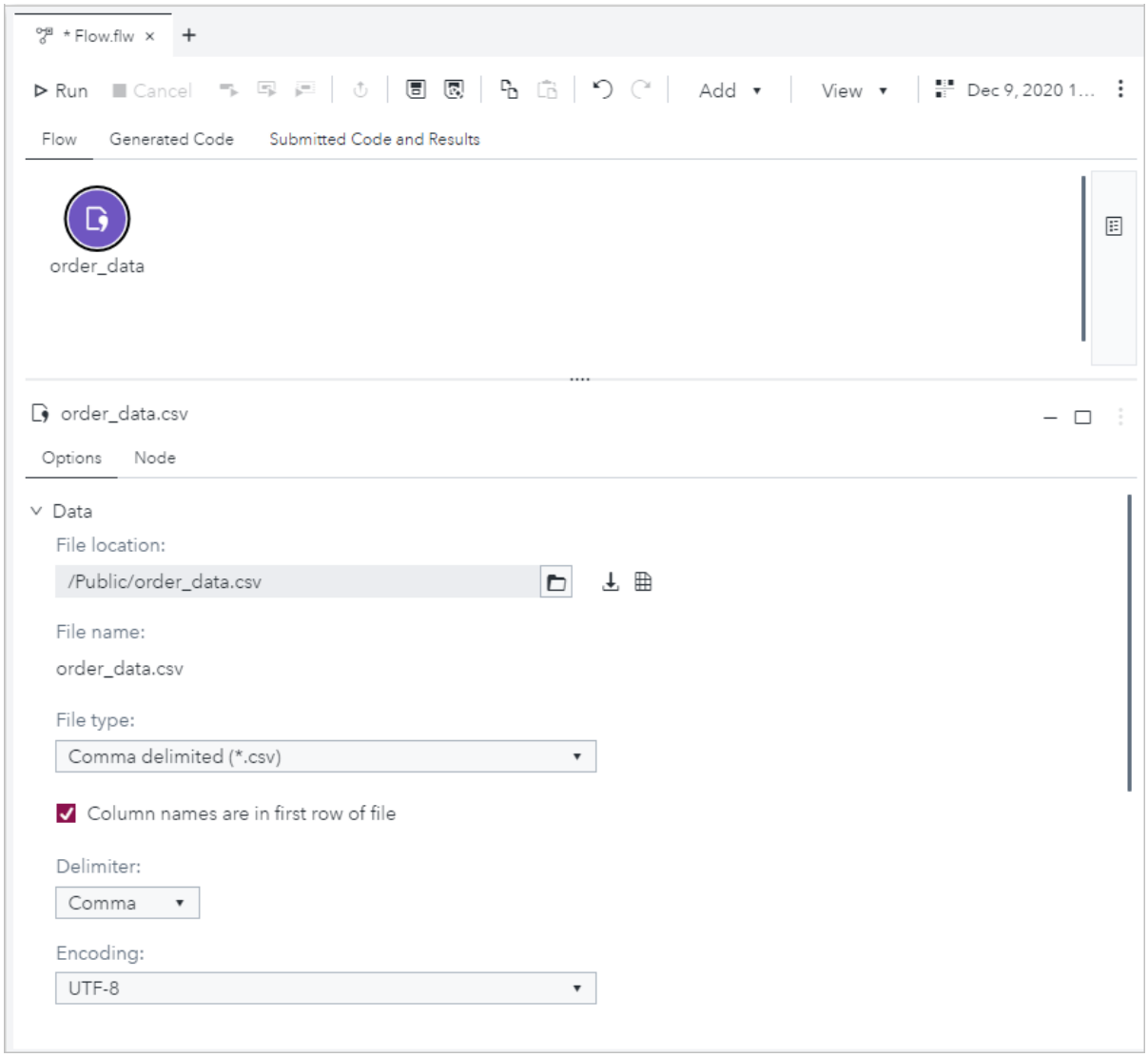
- text (*.txt).
- tab-delimited (*.tab, *.tsv).
- delimited (*.dlm).
- comma-delimited (*.csv) .
- Microsoft Excel (*.xlsx). To import XLSX files, you must license and install SAS/ACCESS to PC Files.
- Microsoft Excel 97–2003 (*.xls).

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.

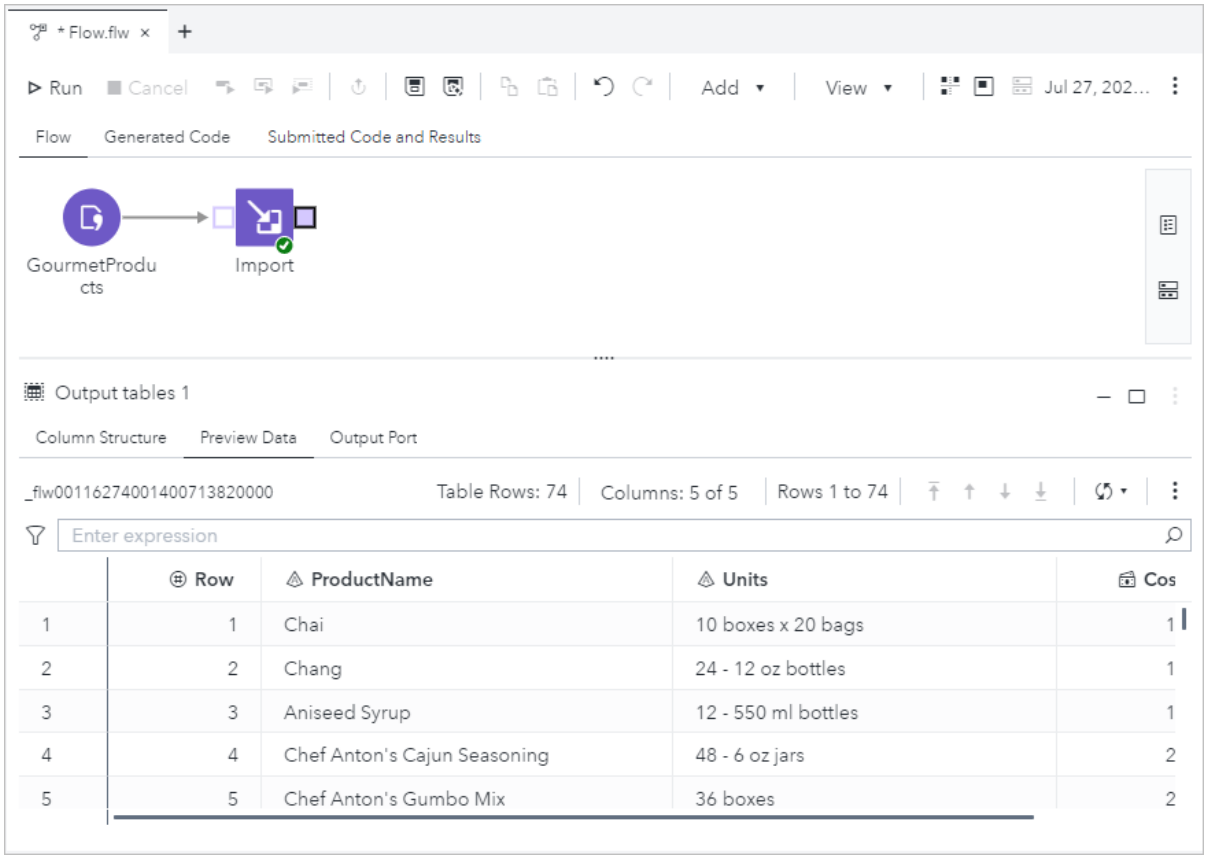
- You can update the following fields:
  - **File location**.
  - **File type**.
  - **Column names are in first row of file** - indicates whether the data includes a header row.
  - **Delimiter**.
  - **Encoding**.

Importing Data from an External File

About the Import Step

The information in external files, such as delimited files and Microsoft Excel spreadsheets, must be converted to a SAS table in order to be queried, filtered, or analyzed by SAS. The Import node enables you to convert the data in external delimited, text, fixed width, and Microsoft Excel files to SAS tables.

The next figure shows a flow that converts the information in a delimited file (GourmetProducts) 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.



There are three main steps to import a file:

1. Analyze the data, and then verify that SAS Studio is reading the data correctly. If necessary, update the options that are used to analyze the data or load the column structure from an external file.
2. Verify that the column properties for the output data are correct. If necessary, update the options that are used to analyze the data or load the column structure from an external file.
3. Run the flow to import the data.

Node Connection Requirements

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

Input Port	Output Port
■ File node	No connection is required. **Note:** 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. By default, the output data is written to a temporary table in the Work library. You can specify the library and name of the

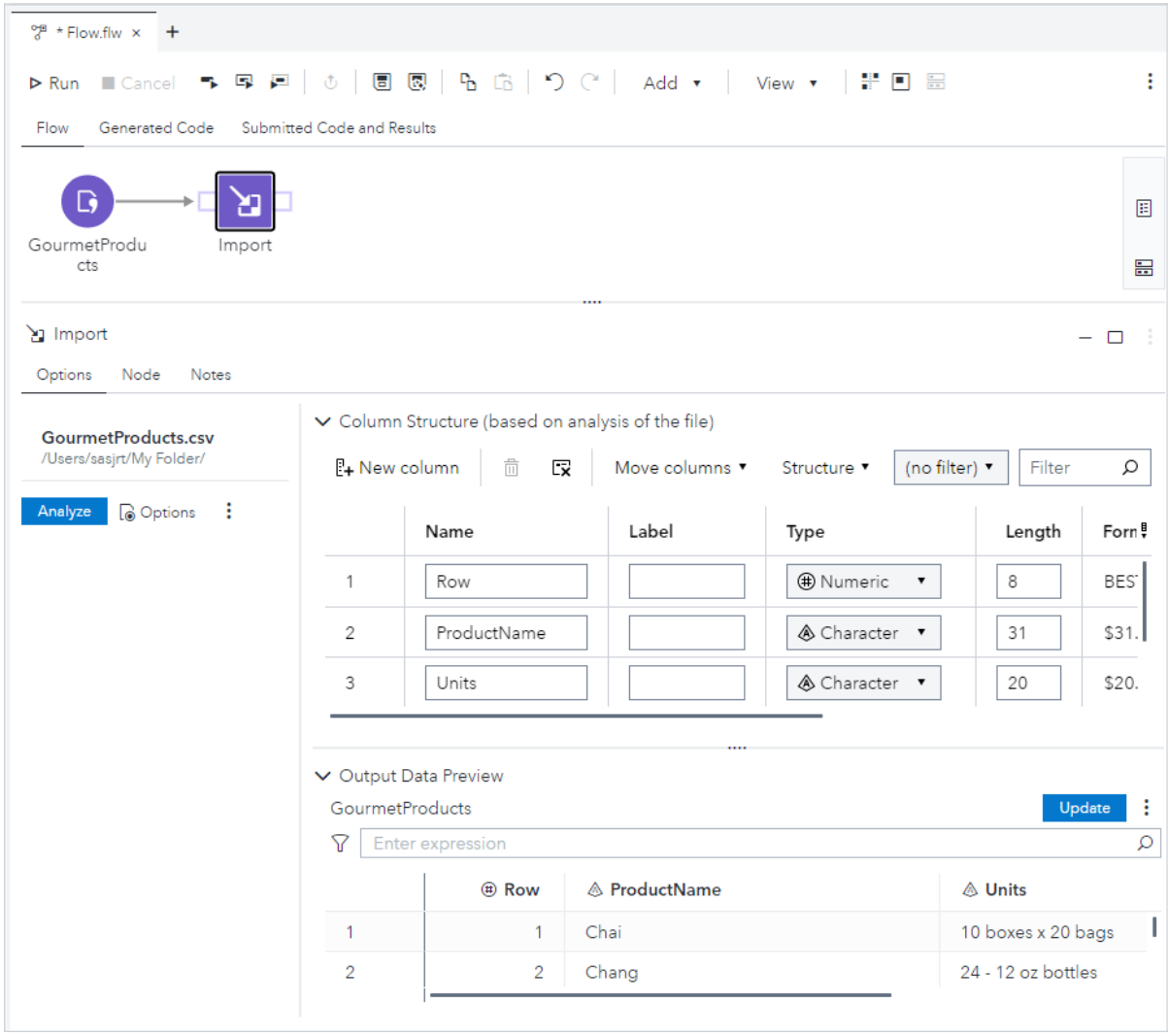
Input Port	Output Port
	output table by connecting the output port to a Table node. For more information, see “Adding a Table from a SAS Library to a Flow” on page 34.

Import an External File

Step 1: Analyze the Data

To analyze the data in the file that you want to import:

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 connected to the input port of the Import node.
2. Click the Import node, and then click the **Options** tab. Click **Analyze** to identify the structure of the external file. The following figure shows the results from analyzing an external delimited 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 If you are importing a fixed-width file, the data is loaded into a single column. Use the Column Structure area to add the other columns to the output table and specify column properties. Use the Start Position and End Position columns to specify where in the data each column starts and ends.

Note: If you change the file type of the file node that you are importing, any changes that you made to the column structure are not saved.

Step 2: Review Column Structure for Output File and Adjust As Needed

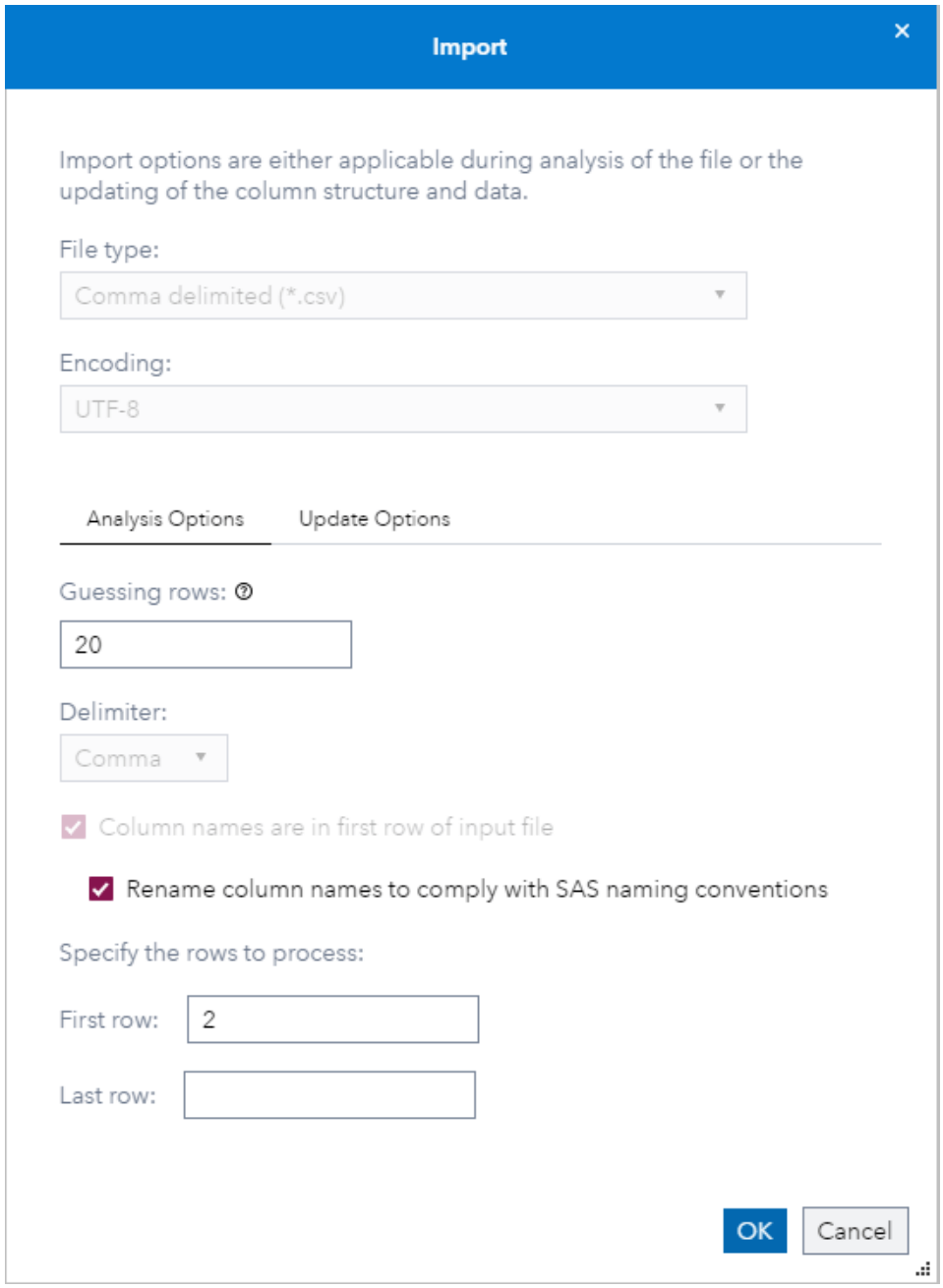
Review the column structure for the output file. To make adjustments to the column structure, update the options that are used to analyze the data or load the column structure from an external file.

- Update the Analysis Options

  To update the options that are used to analyze the column structure of the file that you want to import, click **Options**, and then click the **Analysis Options** tab.

  Note: If **Options** is not visible on the Import Options toolbar, click ⋮ and select **Options**.

  The following figure shows the Analysis Options tab for a comma-delimited file:



The data options depend on the file type.

Table 2.1 *Data Options for Each File Type*

File Type	Available Data Option
Delimited file	■ **Guessing rows** - indicates the number of rows to scan in the input file to determine the appropriate data type, informat, format, and length of columns. Setting this option to higher values could adversely affect performance or cause SAS Studio to time out. It is recommended that you set this option to a low value initially to avoid potential performance or time-out issues. If guessed column properties are not as expected, the option

File Type	Available Data Option
	value can be increased and the file re-scanned. The limit is 2,147,483,647 rows. ■ **Rename column names to comply with SAS naming conventions** - updates column names to comply with SAS variable name rules. ■ **Specify the rows to process** - specifies the first and last rows to process in a delimited text file. To update the **Column names are in the first row of input file** or **Delimiter** options, click the File node in the flow and update the options on the Options tab.
Fixed-width file	**Specify the rows to process** - specifies the first and last rows to process in a delimited text file.
Microsoft Excel file	■ **Rename column names to comply with SAS naming conventions** - updates column names to comply with SAS variable name rules. ■ **Limit the range of imported columns and rows** - enables you to specify a range of cells to import from the Microsoft Excel file.

After you update the Analysis options, click **OK**, and then click **Analyze** again to apply the changes to the column structure.

- Load Column Structure from an External File

  You can use a column structure metadata file to load the structure of the file that you are importing. Using a column structure metadata file can increase the reliability and speed of your import process. You can use the column structure metadata file to make necessary edits to the column structure rather than manually editing each row of your data files.

  The column structure metadata file must meet these requirements:

  - All values must be comma-delimited.
  - The first line must include the column properties and the order in which they appear. Column property names are case-sensitive.
  - Each column must be listed on a separate line with its properties.
  - The file can be saved as a comma-separated values (*.csv) or a text (*.txt) file.

  Note: You can add comments to the column structure metadata file by entering # at the start of the line.

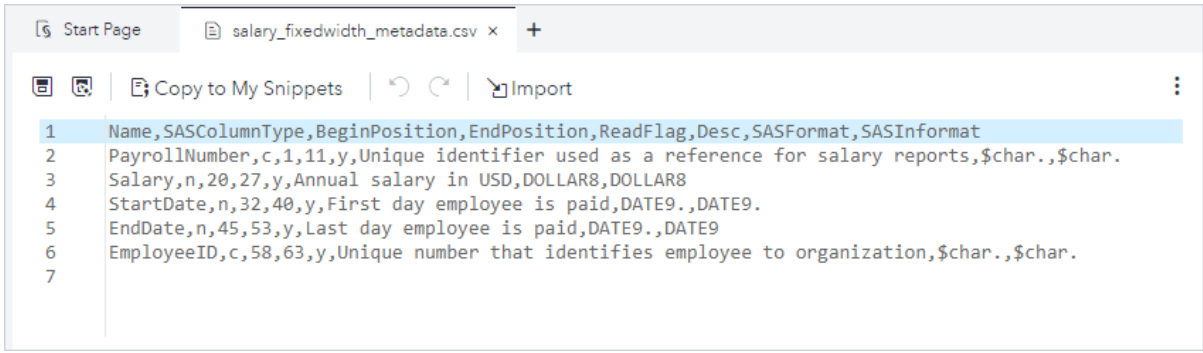
  To load the column structure from an external file, select **Structure** ⇨ **Load structure**. Select the column structure metadata file that you want to use and click **OK**. The new column structure appears in the Column Structure area.

  Note: You can generate a column structure metadata file from the column structure by selecting **Structure** ⇨ **Generate structure**. You can save the file as

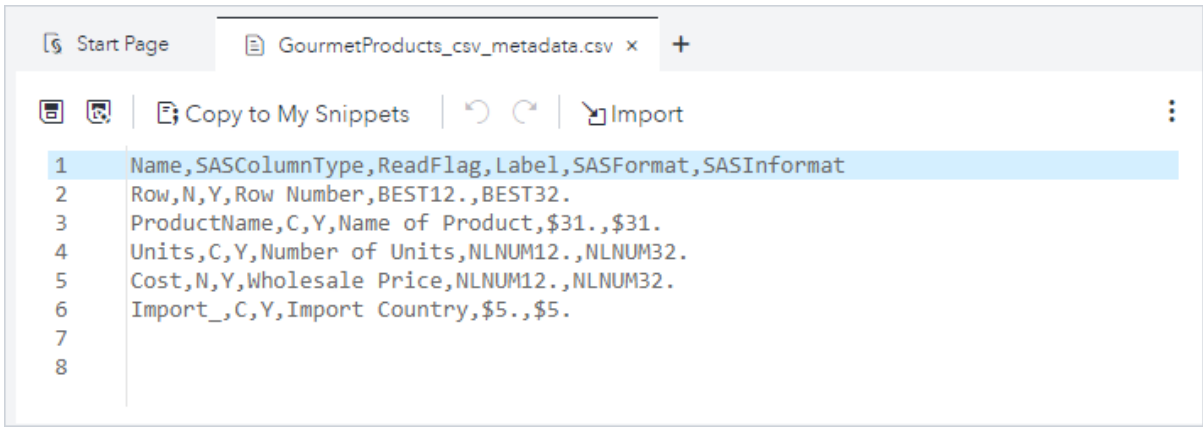
a comma-delimited (*.csv) file or a text (*.txt) file. The column structure metadata file is always saved with UTF-8 encoding.

Here is an example of a column structure metadata file for a fixed-width file:

- Line 1 of the salary_fixedwidth_metadata.csv file defines the column properties and the order in which they appear:
  `Name,SASColumnType,BeginPosition,EndPosition,ReadFlag,Desc,SASFormat,SASInformat`
- Lines 2–6 list the properties for these five columns: PayrollNumber, Salary, StartDate, EndDate, and EmployeeID.



Here is an example of a column structure metadata file for a CSV file. Note that the first line does not include the BeginPosition and EndPosition columns because the imported file is not a fixed-width file. In addition, the metadata in this example includes a column label instead of a description.



When you create a column structure metadata, you can include all of these column properties or a subset of these properties. The properties can be listed in any order depending on what is required for the data that you want to import.

Name of Metadata Column	Required?	Description
Name	Yes.	Name of the column in the data file. In the example, PayrollNumber is the first column name.
SASColumnType	No.	SAS column type. Columns can be

Name of Metadata Column	Required?	Description
		`character` (c) or `numeric` (n). If you do not specify a value, the type defaults to `character`.
BeginPosition	Yes, if you are importing a fixed-width file. This value is ignored if you are importing a delimited file.	The first position in a record where the data for this column begins. If you specify a non-numeric value for this position, the property is not set in the column metadata and the column length defaults to `8` for both numeric and character values.
EndPosition	Yes, if you are importing a fixed-width file. This value is ignored if you are importing a delimited file.	The last position in a record where the data for this column ends. If you specify a non-numeric value for this position, the property is not set in the column metadata and the column length defaults to `8` for both numeric and character values.
ReadFlag	No.	Specifies whether to read the records for this column. Valid values are `y` or `n`. If you specify `n`, the column is not read. If you do not specify a value for this property, the record is read.
Desc or Label	No.	Specifies either a description of the column or a label for the column.
SASFormat	No.	The SAS format that is used to display the data in the imported table.
SASInformat	No.	The SAS informat that is used to read the data in from the fixed-width file.

Step 3: Update the Column Properties

In the Column Structure area, you can update the properties for each column, including the column name, label, data type, length, format, and informat. You can also add, delete, and reorder columns.

- If you are importing a fixed-width file, click **New column** to add columns to the output table. Use the **Start Position** and **End Position** columns to specify where in the data each column starts and ends.
- You cannot change the column information for Microsoft Excel files.

After you make changes to the column properties, click **Update** to apply the changes to the Output Data Preview window.

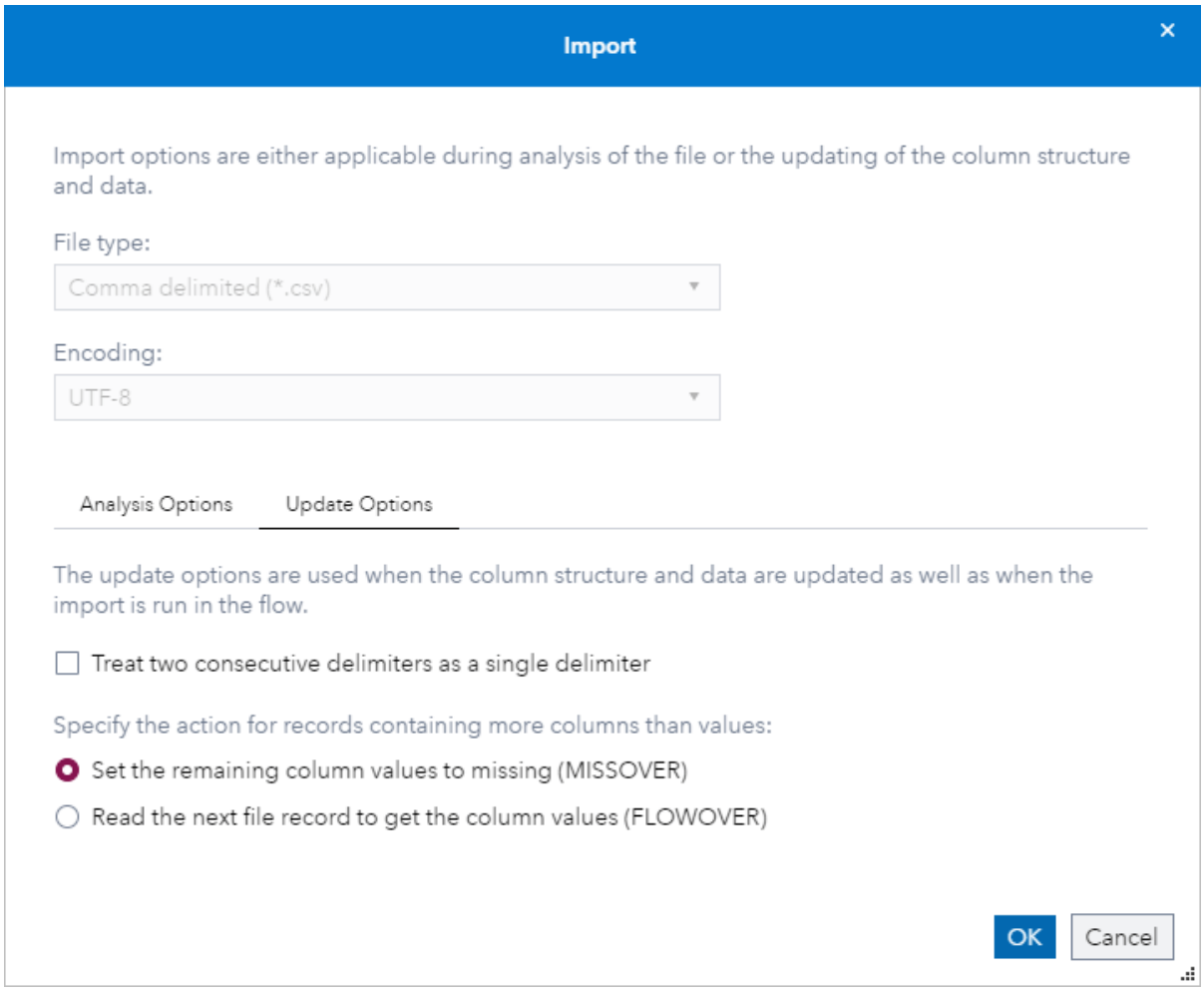
Note: If you click **Analyze** again, any changes you have made to the column properties are overwritten.

Step 4: Update the Options for the Output Table

If necessary, you can update the options that are used when the output table is generated by clicking **Options**, and then clicking the **Update Options** tab. This functionality is not available if you are importing a Microsoft Excel file.

Note: If **Options** is not visible on the Import Options toolbar, click ⋮ and select **Options**.

The following figure shows the Update Options tab for a comma-delimited file:



The data options depend on the file type.

Table 2.2 *Data Options for Each File Type*

File Type	Available Data Option
Delimited File	■ **Treat two consecutive delimiters as a single delimiter** - specifies that two consecutive delimiters should be treated as a single delimiter. If you do not select this option, the consecutive delimiters are treated as a missing value. ■ Select the option that you want to use for rows that have more columns than values: □ **Set the remaining column values to missing (MISSOVER)** - specifies that columns without any values are treated as missing values. □ **Read the next file record to get the column values (FLOWOVER)** - reads the next row of data when there are columns in the previous row without values.
Fixed-width File	■ **Pad short lines with blanks up to the specified record length** - adds blanks to the ends of lines that are shorter than the specified record length.

File Type	Available Data Option
	■ Select the option that you want to use for rows that have more columns than values: □ **Read the next file record to get the column values (FLOWOVER)** - reads the next row of data when there are columns in the previous row without values.

After you make changes to the Update options, click **OK**. Click **Update** again to apply the changes to the Output Data Preview window. The output table is not actually generated until you run the flow.

Step 5: Run the Flow and Import the Data

To import the data and generate the output table, click ▷ Run.

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 edit and add columns to the table, click **Edit structure**. 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 of the table that is specified on the Table Properties 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.

3

Developing Code in a Flow

SAS Program Step: Writing SAS Code in a Flow

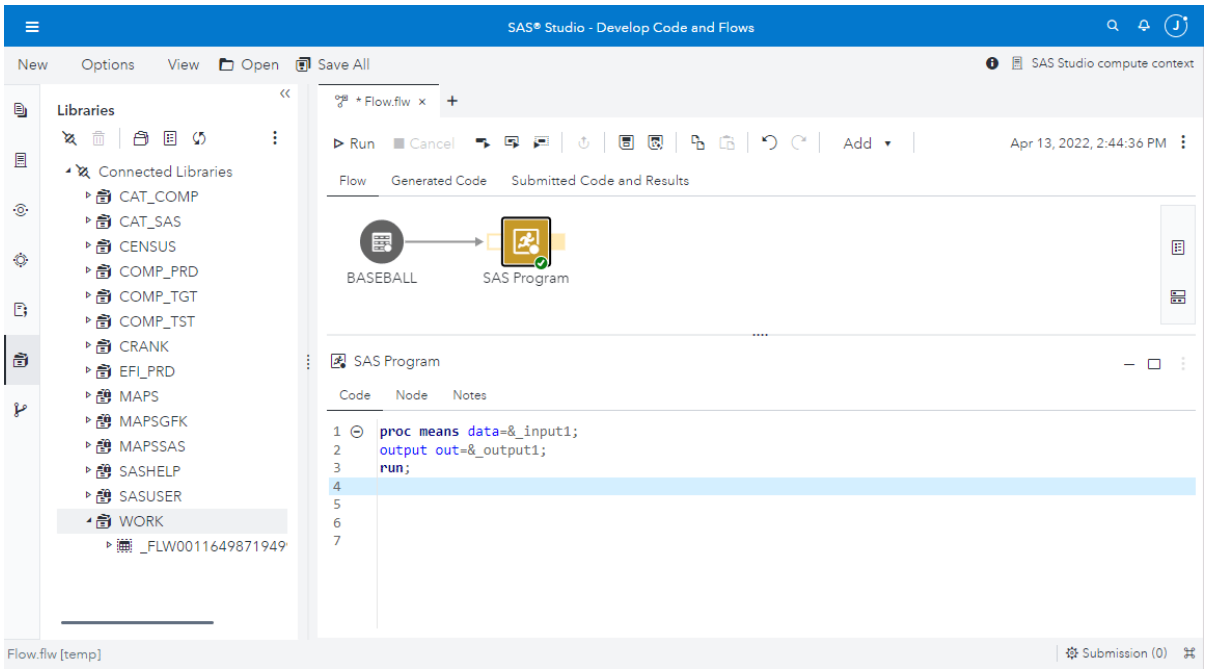
About the SAS Program Step

You can use the SAS Program step to include any type of SAS program that is required in a flow. The code that is associated with the SAS Program node is displayed in the node details on the Code tab. The Code tab includes the same

color-coded, syntax-checking editor that you use to write stand-alone SAS programs. For more information, see "About the SAS Code Editor" in *SAS Studio: User's Guide*.

In a SAS Program node, your SAS program can either be embedded in the flow or reference an external program file. Embedded programs can be accessed only from within the flow; externally referenced programs are saved program files. For more information, see "Understanding Embedded and Externally Referenced Programs" on page 43.

By default, output data from a SAS Program node is written to a table in the Work library. The next example shows a flow in which a SAS Program node runs the MEANS procedure against the BASEBALL table. The results are written to a table in the Work library.



Node Connection Requirements

No node connections are required to run the SAS Program step. You can connect the SAS Program input and output ports to Table nodes or operational nodes that create an output table, such as a Query node or an Import node, and then use macro variables to reference the tables in your program. For more information, see "Using Macro Variables to Reference Input and Output Ports" on page 47.

Note: By default, any output data is written to temporary tables in the Work library. You can specify the library and name of the output table by connecting the output port to a Table node. For more information, see "Adding a Table from a SAS Library to a Flow" on page 34.

SAS Program: Step-by-Step Instructions

The SAS Program step requires no node connections.

1. In the **Steps** section of the navigation pane, expand the **Develop** folder and double-click **SAS Program**.
2. Click the **SAS Program** node, and then use the Code tab to enter your SAS code. The SAS Program node programming interface is the same as the interface for stand-alone SAS programs.

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:

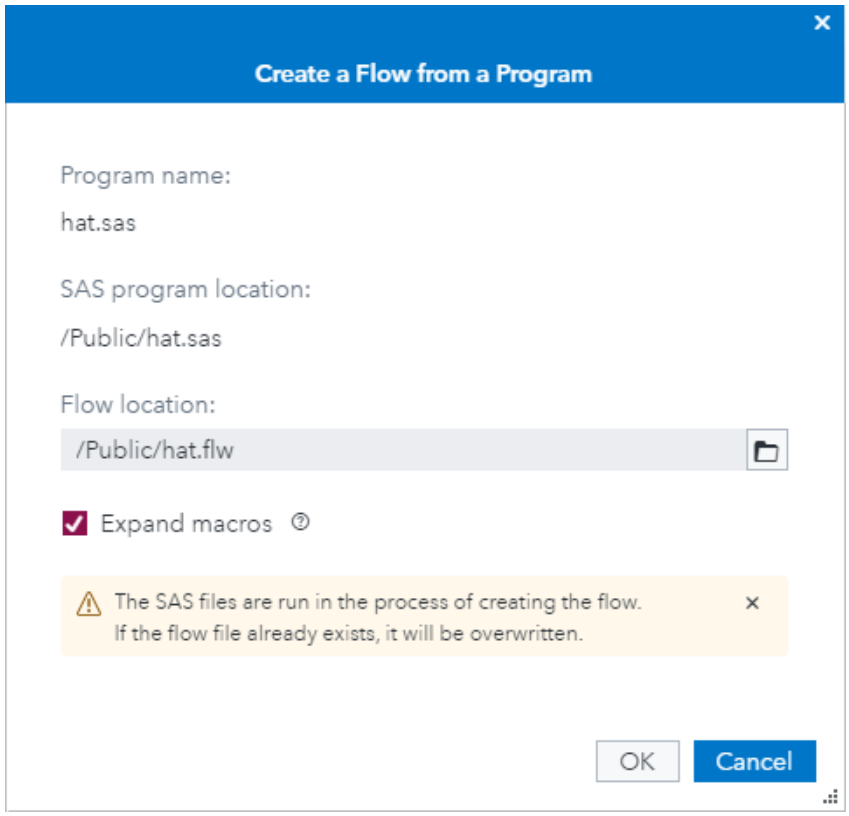
- The **Create a Flow from a Program** option cannot parse all of the possible LIBNAME options for DBMS engines. If you are creating a flow from a program that contains LIBNAME options for DBMS engines, you need to verify that the LIBNAME statement in the SAS Program node in the flow is correct and that 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.

Note: Any programs that are created are embedded in the flow. For more information, see "Understanding Embedded and Externally Referenced Programs" on page 43.

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

Python Program Step: Writing Python Code in a Flow

About the Python Program Step

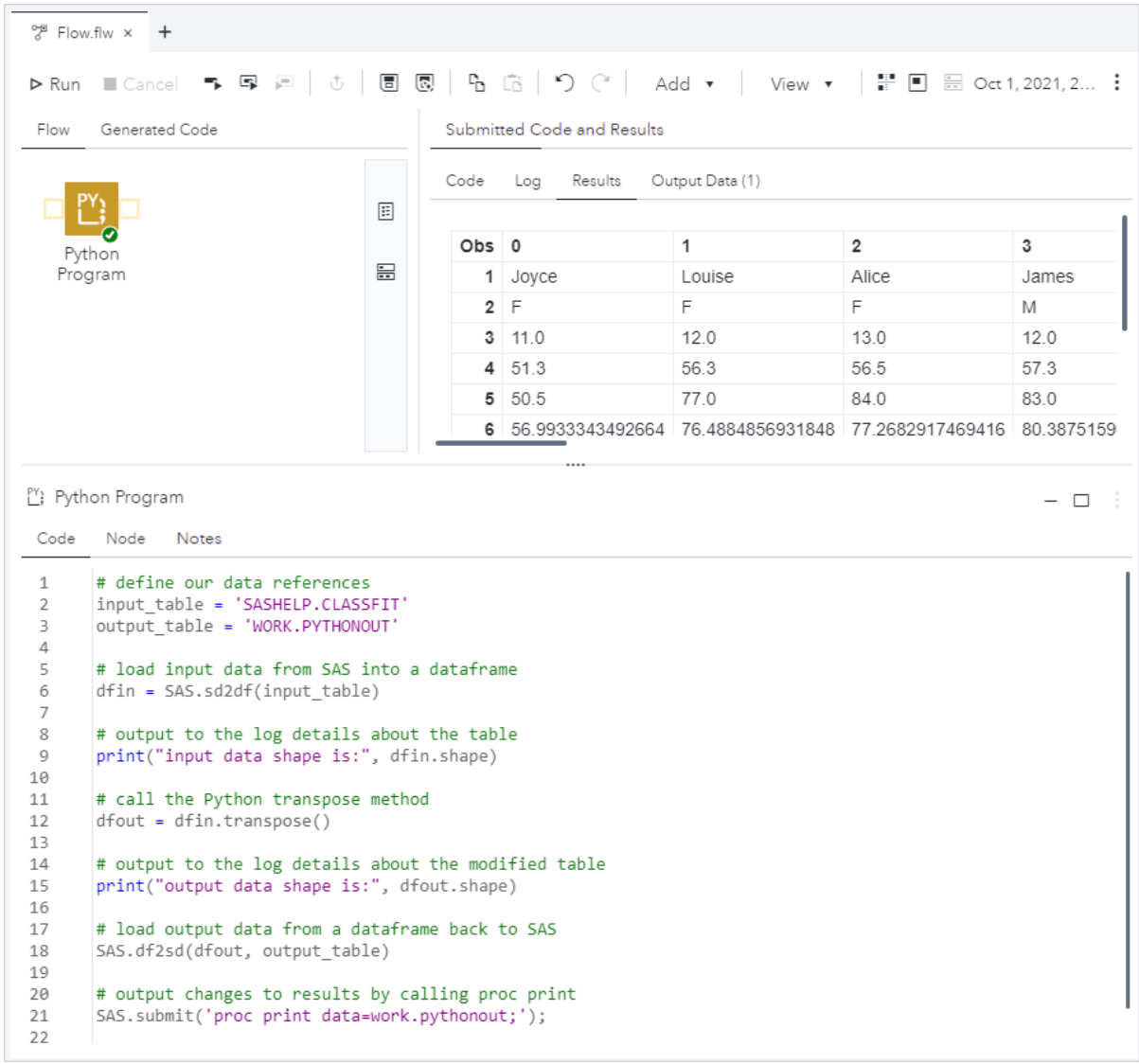
You can use the Python Program step to run Python code in a flow without explicitly using PROC PYTHON. Your PROC PYTHON options are stored in SAS Environment Manager and do not need to be included in your Python program. SAS Studio automatically uses your Python code to generate a SAS program by using PROC PYTHON. For more information, see "PYTHON Procedure" in *Base SAS Procedures Guide*.

The code that is associated with the Python Program node is displayed in the node details on the Code tab. The Code tab includes the same color-coded, syntax-checking editor that you use to write stand-alone Python programs. For more information, see "About the Python Code Editor" in *SAS Studio: User's Guide*.

In a Python Program node, your Python program can either be embedded in the flow or reference an external program file. Embedded programs can be accessed only from within the flow; externally referenced programs are saved program files. For more information, see "Understanding Embedded and Externally Referenced Programs" on page 43.

The next figure shows a flow in which a Python Program node transposes the rows and columns of a SAS data set and writes the output data to the Work library.

Note: This program includes hardcoded references to the input and output tables. You can also use macro variables to reference the input and output ports of the Python Program node so that you do not have to hardcode table names in the program. For more information, see "Using Macro Variables to Reference Input and Output Ports" on page 47.



Node Connection Requirements

No node connections are required to run the Python Program step. You can connect the Python Program input and output ports to Table nodes or operational nodes that create output tables, such as a Query node or an Import node, and then use macro variables to reference the tables in your program. For more information, see “Using Macro Variables to Reference Input and Output Ports” on page 47.

Note: By default, any output data is written to temporary tables in the Work library. You can specify the library and name of the output table by connecting the output port to a Table node. For more information, see “Adding a Table from a SAS Library to a Flow” on page 34.

Python Program: Step-by-Step Instructions

The Python Program step requires no node connections.

1. In the **Steps** section of the navigation pane, expand the **Develop** folder and double-click **Python Program**.
2. Click the **Python Program** node, and then use the Code tab to enter your Python code. The Python Program node programming interface is the same as the interface for stand-alone Python programs.

Understanding Embedded and Externally Referenced Programs

About Embedded and Externally Referenced Programs

The programs that are associated with SAS or Python Program nodes can either be embedded in the flow or externally referenced. Embedded programs are stored in the flow. You can access and edit an embedded program only from within the flow. When you create a program in your flow, the program is embedded until you save it as a file.

An externally referenced program is saved in an external location such as a SAS Content or SAS server folder. When you add an existing program to a flow, the path name of the program is stored in the program node. When you run an externally referenced program node, SAS Studio runs the external program file. You cannot make changes to an externally referenced program from within a flow. You must edit the program outside of the flow. For more information, see “Editing an Externally Referenced Program” on page 45.

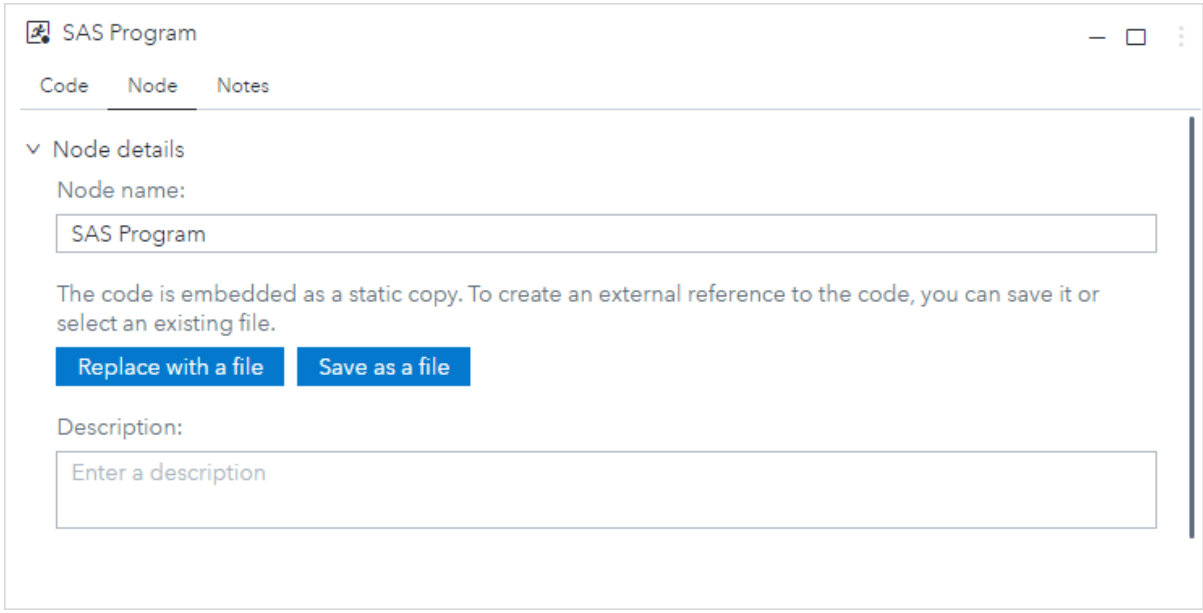
> **TIP** If you need to use a program in multiple flows, consider saving the program and adding it as an externally referenced program to each flow. If you need to make changes to the program later, you can edit the program in one place.

Changing How a Program Is Stored

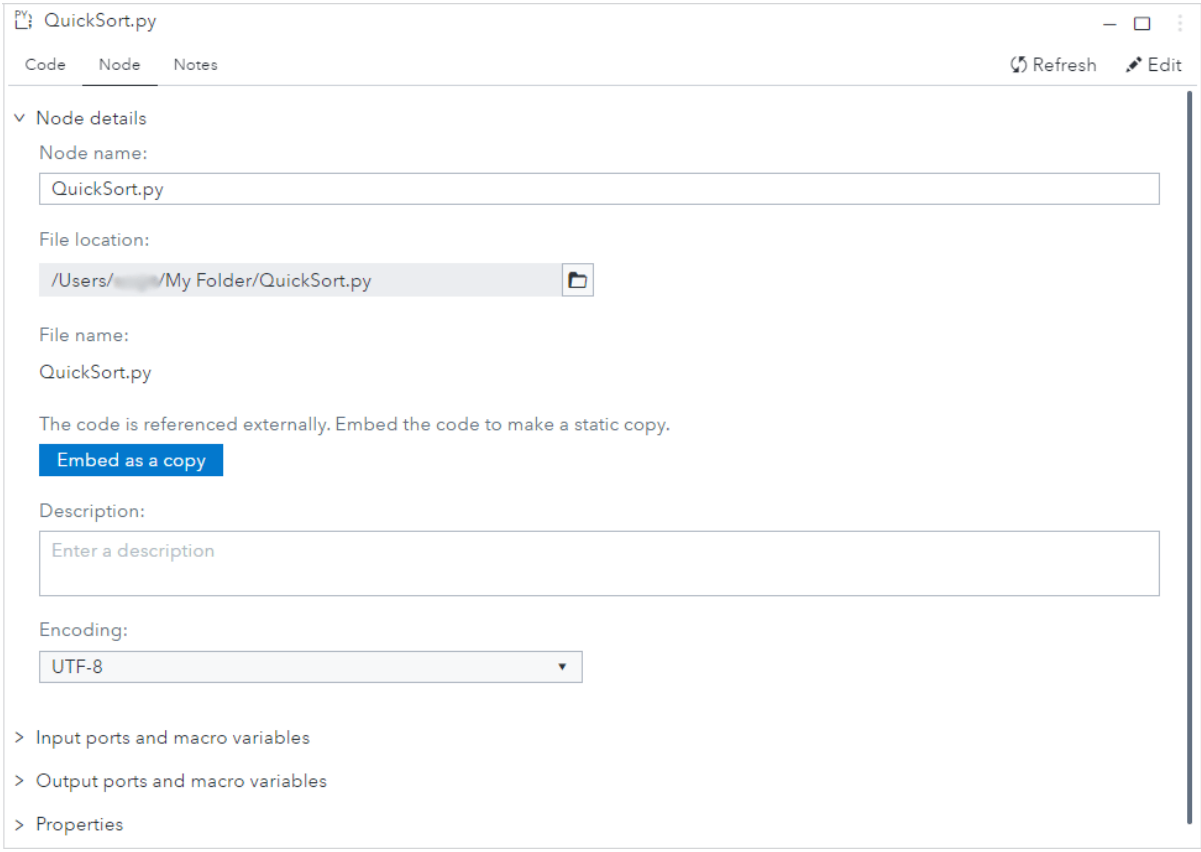
You can change whether a program is embedded in the flow or externally referenced.

To change how a program is stored:

1. Click the SAS Program or Python Program node on the flow canvas, and then click the **Node** tab in the node details.
2. You can use these options on the Node tab of the node details to change how a program is stored:
   - **Replace with a file** - replaces an embedded program with a saved program.
   - **Save as a file** - saves an embedded program as an external program.



   - **Embed as a copy** - saves a copy of an externally referenced program as an embedded program. The program no longer references the external file.



Editing an Externally Referenced Program

You cannot make changes to an externally referenced program from within a flow. To edit an externally referenced program, you must edit the program outside of the flow or embed the program in your flow.

To edit an externally referenced program:

1. Click the program node on the flow canvas, and then click the **Node** tab in the node details.
2. You can use these options to edit the program:
   - Click **Edit** on the node details toolbar to open the external program in a new tab. Edit the program and save your changes. If necessary, click **Refresh** on the node details toolbar to update the program in your flow.

     Note: You can also double-click an external program on the flow canvas to open the program in a new tab for editing.
   - Click **Embed as a copy** to embed the program in your flow. Edit the program on the Code tab of the node details. The program no longer references the external file.

Importing and Exporting Flows with Referenced Files

There are some important considerations to keep in mind when you import and export flows using SAS Environment Manager or the commmand-line interface (CLI) for the SAS Viya platform. Not all externally referenced files, including SAS programs, Python programs, and data files, are supported when you import and export a flow. For more information, see "SAS Studio Flows" in *SAS Viya Platform: Content Migration from SAS Viya 4*.

Adding Existing Programs to a Flow

You can add an existing SAS or Python program to a flow by using the **Explorer** section of the navigation pane.

Note: SAS and Python programs are added to the flow as externally referenced programs. For more information, see "Understanding Embedded and Externally Referenced Programs" on page 43.

To add an existing program to a flow:

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

  Note: You must have a flow tab active in the work area in order to use the **Add to flow** option.

  TIP You can also add snippets to the flow from the **Snippets** section of the navigation pane. Snippets are added to the flow as embedded programs. For more information, see "Understanding Embedded and Externally Referenced Programs" on page 43.

Copying Code to a Flow

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

Note: The code in the SAS Program or Python Program node is embedded in the flow and is no longer associated with the original program or task. Editing this code does not affect the original program or task. For more information, see "Understanding Embedded and Externally Referenced Programs" on page 43.

To copy code to a flow:

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

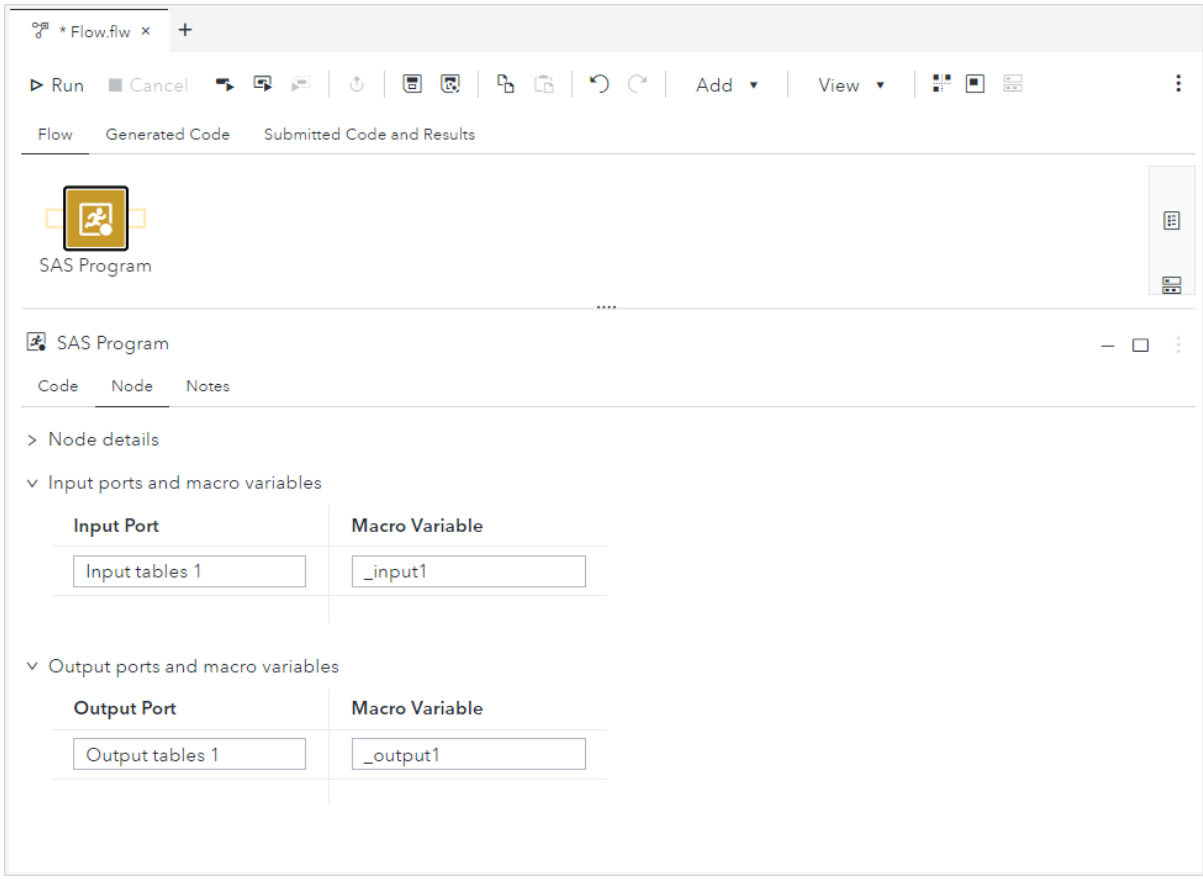
  Note: The **Code to Flow** option is available only if these conditions are met:

- A flow must be open in the work area.
- If you are copying code from a task, the input data source must be specified.

Using Macro Variables to Reference Input and Output Ports

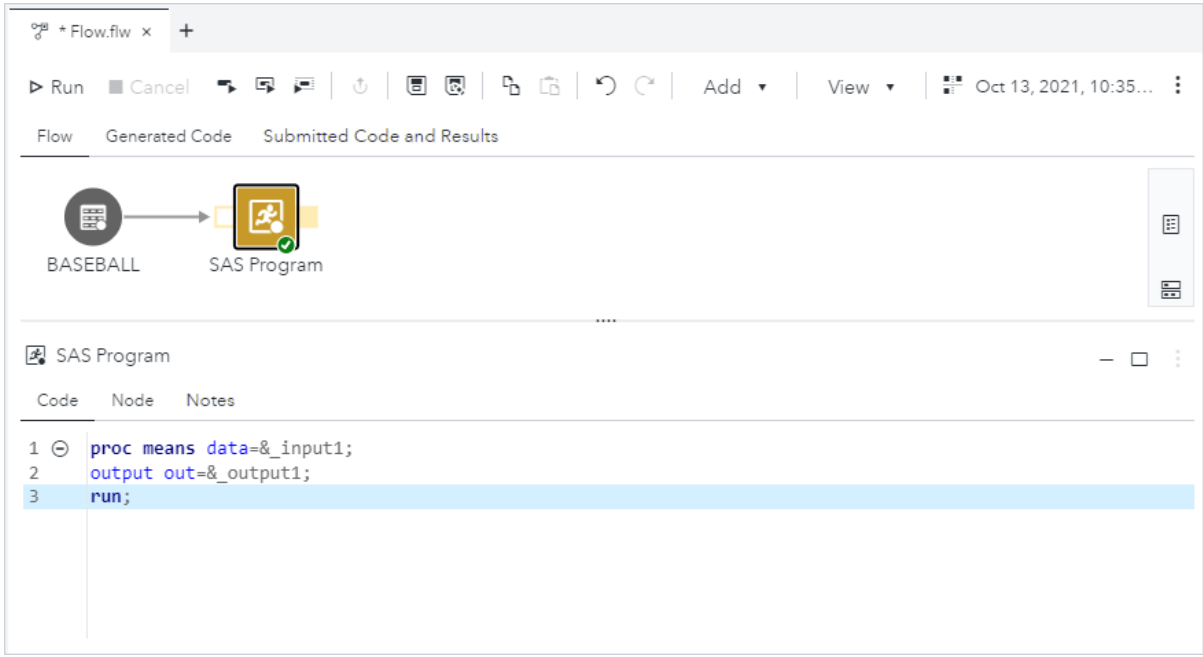
You must use macro variables in your code to reference the input and output ports of a SAS Program or Python Program node before data can be read from or written to a Table node.

You can view the default input and output ports of a 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.

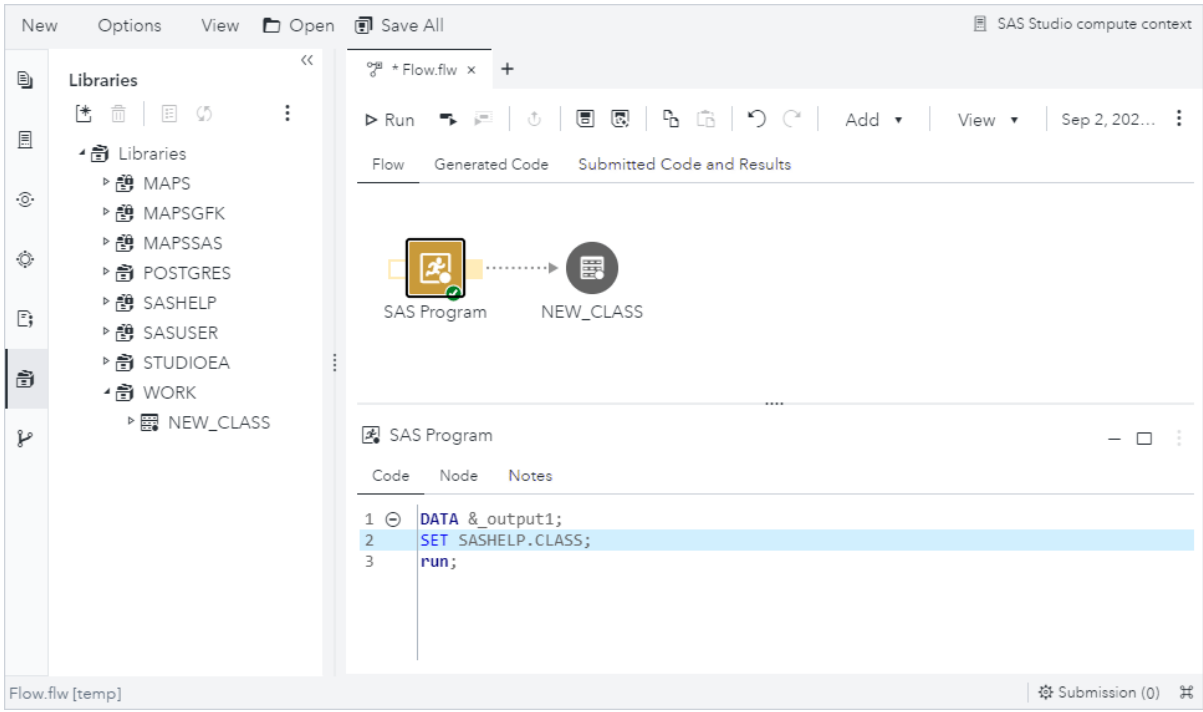


Note: You can change the default names of the macro variables. You can then reference the custom macro variable names in your code. For more information, see "Macro Variables Defined by Users" in *SAS Macro Language: Reference*.

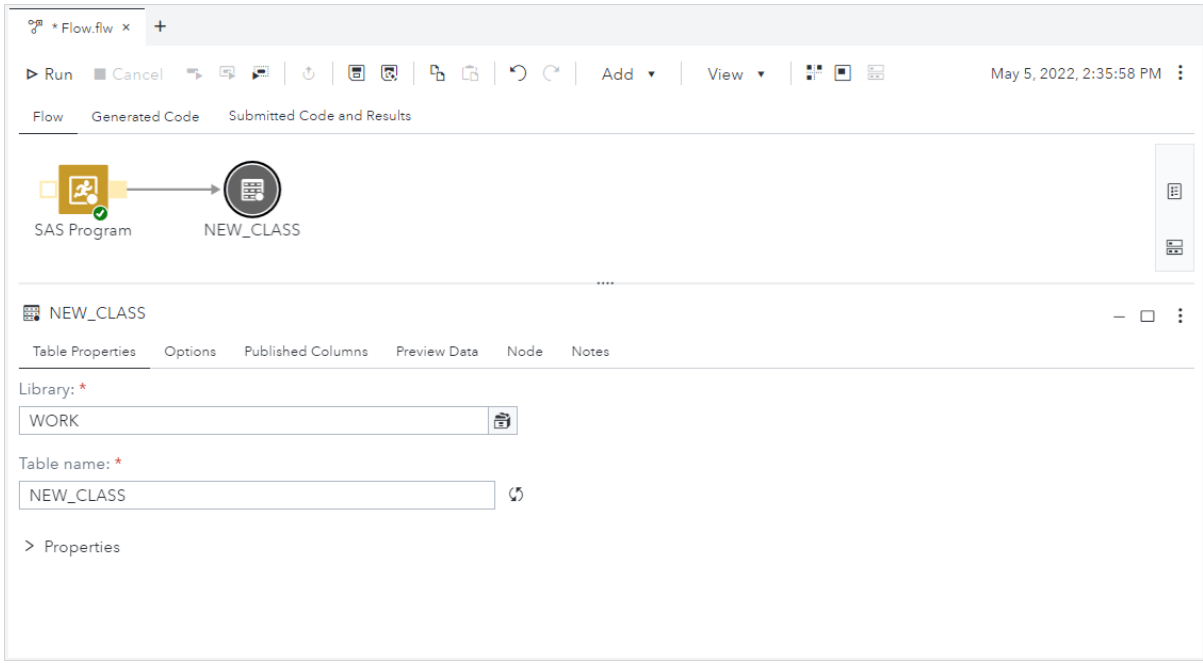
In the following example, the BASEBALL table is connected to a SAS Program node. The SAS Program node runs the MEANS procedure against the BASEBALL table. The code that is associated with the SAS Program node specifies the `&_input1` macro as the data source for the MEANS procedure and the `&_output1` 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.


* Flow.flw
Run
Cancel
Add
View
May 5, 2022, 2:35:58 PM
Flow
Generated Code
Submitted Code and Results
SAS Program
NEW_CLASS
NEW_CLASS
Table Properties
Options
Published Columns
Preview Data
Node
Notes
Library: *
WORK
Table name: *
NEW_CLASS
Properties

4

Creating a Subflow

About Subflows

You can create a subflow by adding a saved flow to your flow. A subflow can be useful if you have a group of nodes that you want to use in multiple locations. You can also use a subflow to reduce the complexity of a flow.

Note: When you create a subflow, note these reminders:

- Only flows that are saved in a SAS Content folder can be used as subflows.
- You cannot run a flow that contains itself as a subflow.
- Subflows cannot be connected to other nodes in the flow.
- The subflow feature is available only if your site licenses SAS Studio Analyst.

Adding a Saved Flow to Another Flow

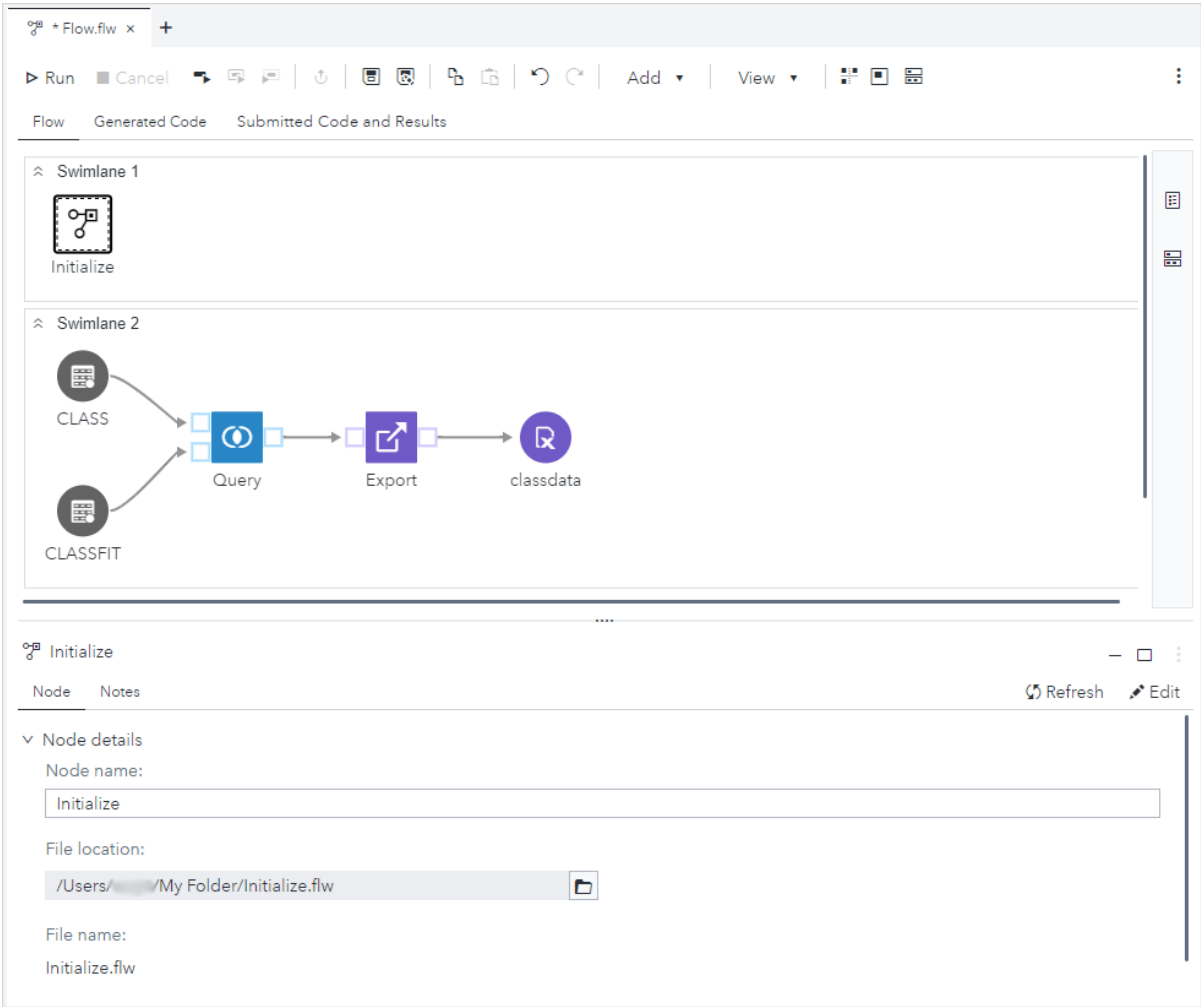
You can add saved flows from a SAS Content folder to another flow by using the **Explorer** section of the navigation pane.

To add a saved flow to another flow:

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

Note: You must have a flow tab active in the work area in order to use the **Add to flow** option. If you do not have an open flow tab, you are prompted to create a flow.

TIP You can use swimlanes to control the order in which the subflow is run in your flow. For more information, see "Controlling the Submission Order of a Flow" on page 168.



Editing a Subflow

You cannot make changes to a subflow from within another flow.

To edit a subflow:

1 You can open a subflow for editing in these ways:

- Click the subflow node on the flow canvas, and then click **Edit** in the node details.
- Double-click the subflow node on the flow canvas.
- Right-click the subflow node on the flow canvas and select **Open and edit**.

The subflow opens in a separate tab.

2 Edit the flow and save your changes.

Note: If you edit the subflow from within the same SAS Studio session, SAS Studio automatically refreshes the subflow content in your flow. If the subflow is edited in another SAS Studio session, you must click **Refresh** in the node details to update the subflow content in your flow.

Note: You can also edit the subflow by opening it from the **Explorer** section of the navigation pane and saving your changes.

5

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 “Branch Rows Step: Splitting an Input Table into Output Tables” on page 57.
- **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 “Filter Rows Step: Subsetting Rows from an Input Table into an Output Table” on page 69.
- **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 86.

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

Table 5.1 *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

	Branch Rows	Filter Rows	Query
Includes a condition for rows that do not match any other condition	x		
Writes rows to multiple output tables	x		
Selects rows from multiple input tables			x
Filters, sorts, and groups columns			x
Creates calculated columns			x

Branch Rows Step: Splitting 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 56.

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:

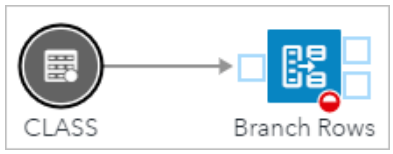
Input Port	Output Port
■ Table node or ■ Operational node that creates an output table, such as a Query node, a SAS Program node, or an Import node	No connection is required. **Note:** By default, the output data is written to temporary tables in the Work library. You can specify the library and name of the output tables by connecting the output ports to Table nodes. For more information, see "Adding a Table from a SAS Library to a Flow" on page 34.

Example: Splitting the Sashelp.Class Data Set

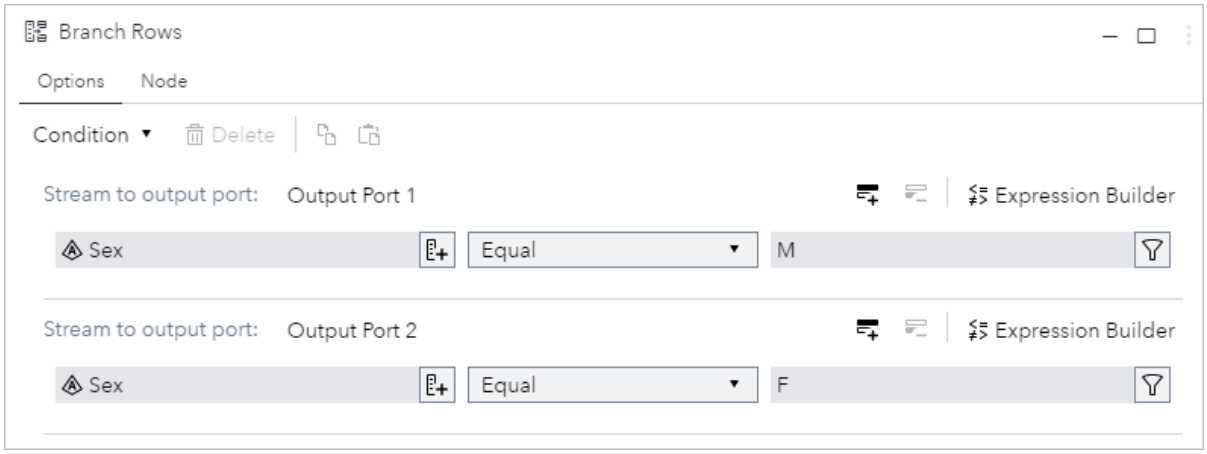
In this example, you split the Sashelp.Class data set into two tables: one for male students and another for female students.

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`.



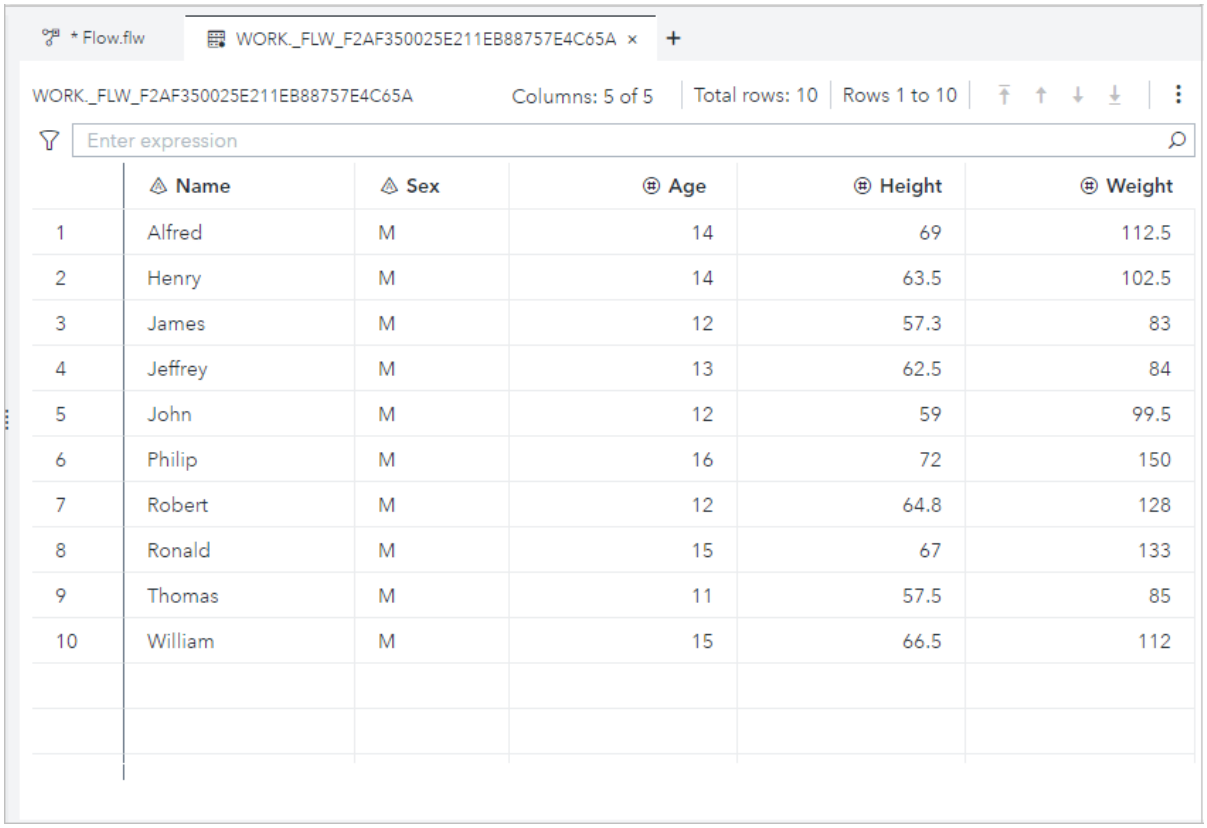
4. Click the **Branch Rows** node to specify the conditions for the output port. On the Options tab, click  for Output Port 1 and select the *Sex* column. Accept the default condition of `Equal`, and click  .
5. In the Add Filter window, click  . Click **Get Values** in the Select Value window. Select `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 34.

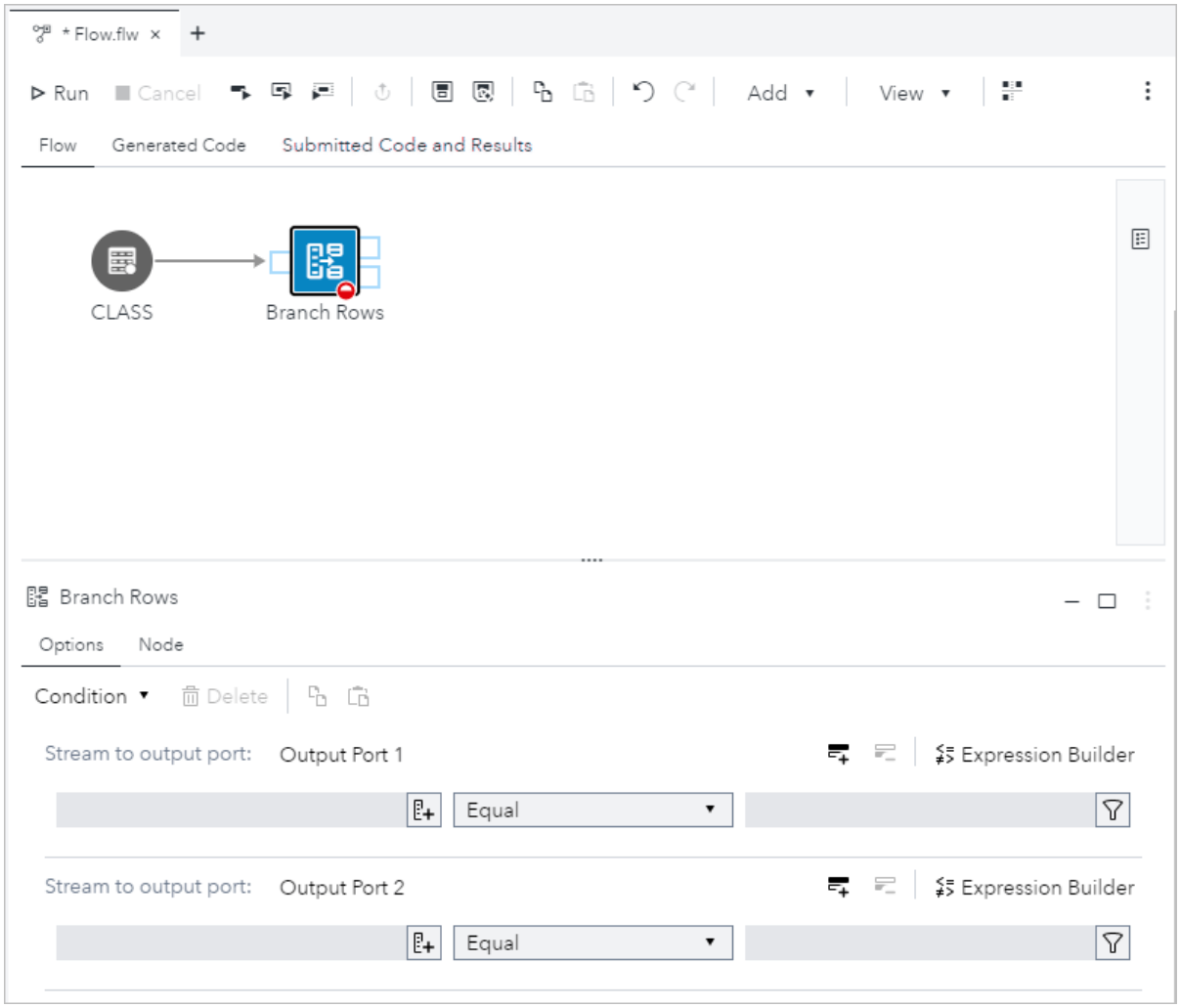


	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Henry	M	14	63.5	102.5
3	James	M	12	57.3	83
4	Jeffrey	M	13	62.5	84
5	John	M	12	59	99.5
6	Philip	M	16	72	150
7	Robert	M	12	64.8	128
8	Ronald	M	15	67	133
9	Thomas	M	11	57.5	85
10	William	M	15	66.5	112

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 12.
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" in *SAS Studio: User's Guide*.

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.

Calculate Columns: Creating a Table from an Existing Table

About the Calculate Columns Step

The Calculate Columns step enables you to create an output table that is based on the input table. The output table includes all of the columns from the input table. You can replace a column in the output table by applying a function to the corresponding column in the input table. You can also create additional columns that are based on columns from the input table.

When you replace or create a column for the output table, a calculation card is created for the column. You can use the card to edit the expression that is associated with the column and view the properties of the column. You can create multiple cards for the same column if you want to apply several functions to a column. The cards are processed in order from top to bottom. You can move columns up and down the list to change the processing order.

New columns are added to the beginning of the output table. Replaced columns are also added to the beginning of the output table, if you change any of the column attributes, including Label, Type, Length, Format, and Informat. The order in which new and replaced columns are added to the beginning of the table is based on the order of the cards.

Note: The Calculate Columns step is available only if your site licenses SAS Studio Analyst.

Node Connection Requirements

In order to run the Calculate Columns 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 more information, see “Adding a Table from a SAS Library to a Flow” on page 34.

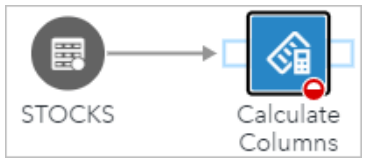
Input Port	Output Port
Program node, or an Import node	

Example: Creating a Table Based on the Sashelp.Stocks Data Set

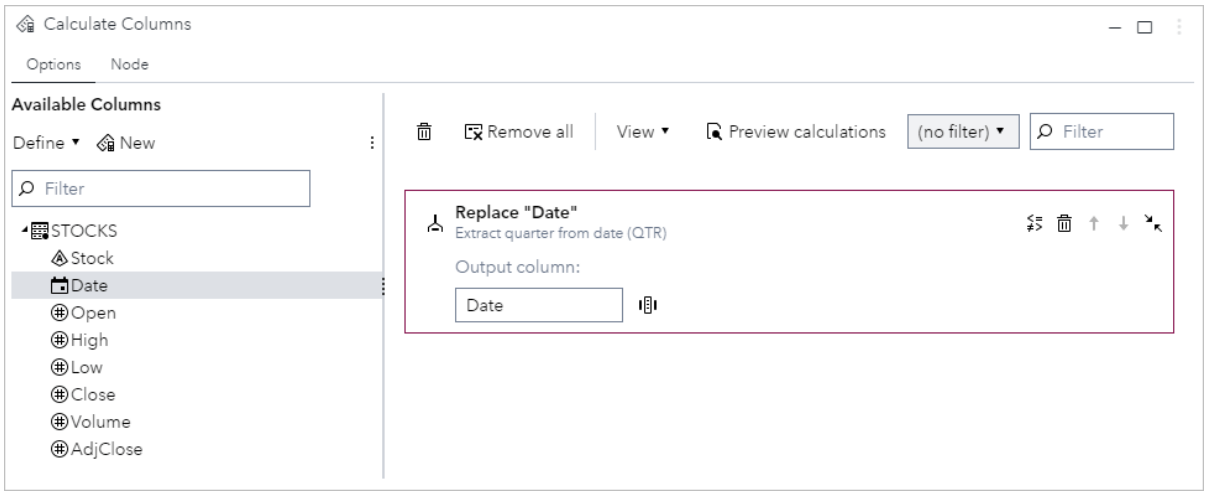
In this example, you create a table from the Sashelp.Stocks data set. In the new table, the date column is replaced with a column of calendar quarter values. You also add a column to calculate the stock price change.

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



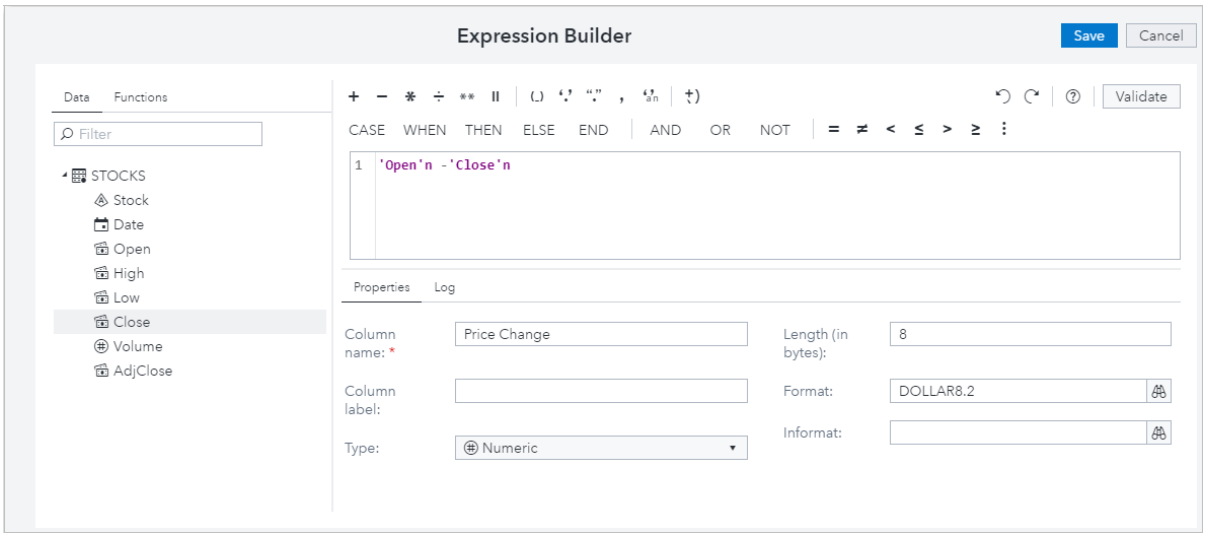
4. Click the **Calculate Columns** node to define the columns for the output table. On the Options tab, select the Date column. Select **Define** ⇨ **Extract** ⇨ **Date** ⇨ **Quarter** to extract the quarter from the Date data. A `Replace "Date"` card is added to the node calculations. In the output table, the data in the Date column is replaced with the calendar quarter.



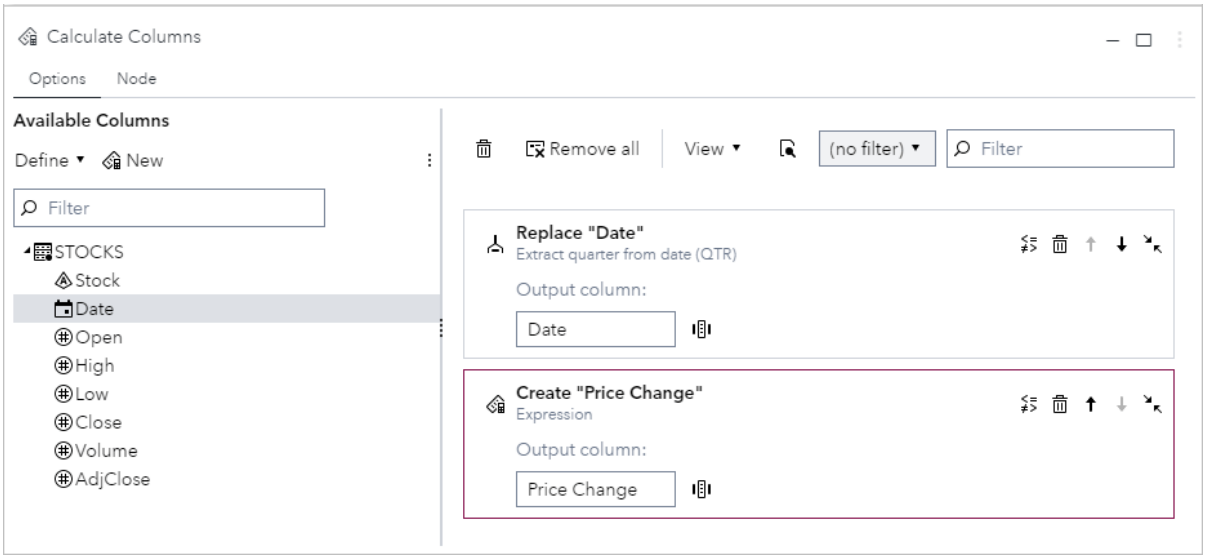
5 Add a column to the table by clicking **New** in the Available Columns area. In the Expression Builder, drag the `Open` column to the expression box. Click the minus operator, and then drag the `Close` column to the expression box. The expression box should contain this expression:

```
'Open'n -'Close'n
```

6 On the Properties tab, enter *Price Change* in the **Column name** box. Change **Type** to `Numeric`, and specify `DOLLAR8.2` as the format.



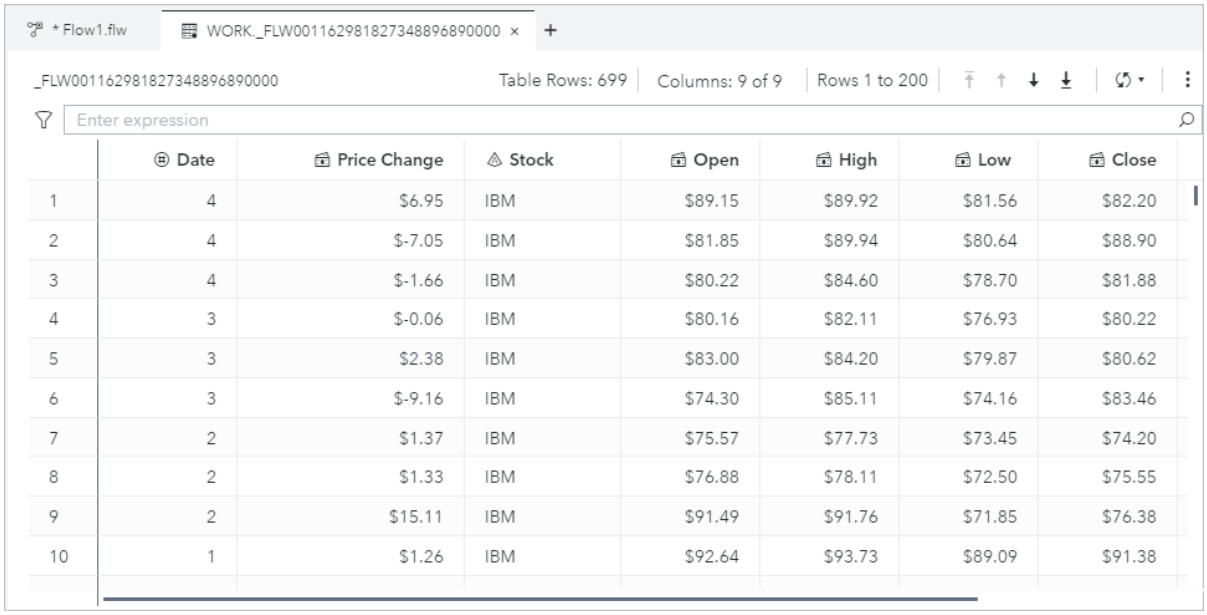
7 Click **Save** to add the new column to the table. A `Create "Price Change"` card is added to the node calculations.



8 To run the flow, click ▷ Run.

Here is the output table with the replaced Date column and the new Price Change column.

> **TIP** You can specify the library and name of the output table by connecting the output port to a Table node. For more information, see "Adding a Table from a SAS Library to a Flow" on page 34.

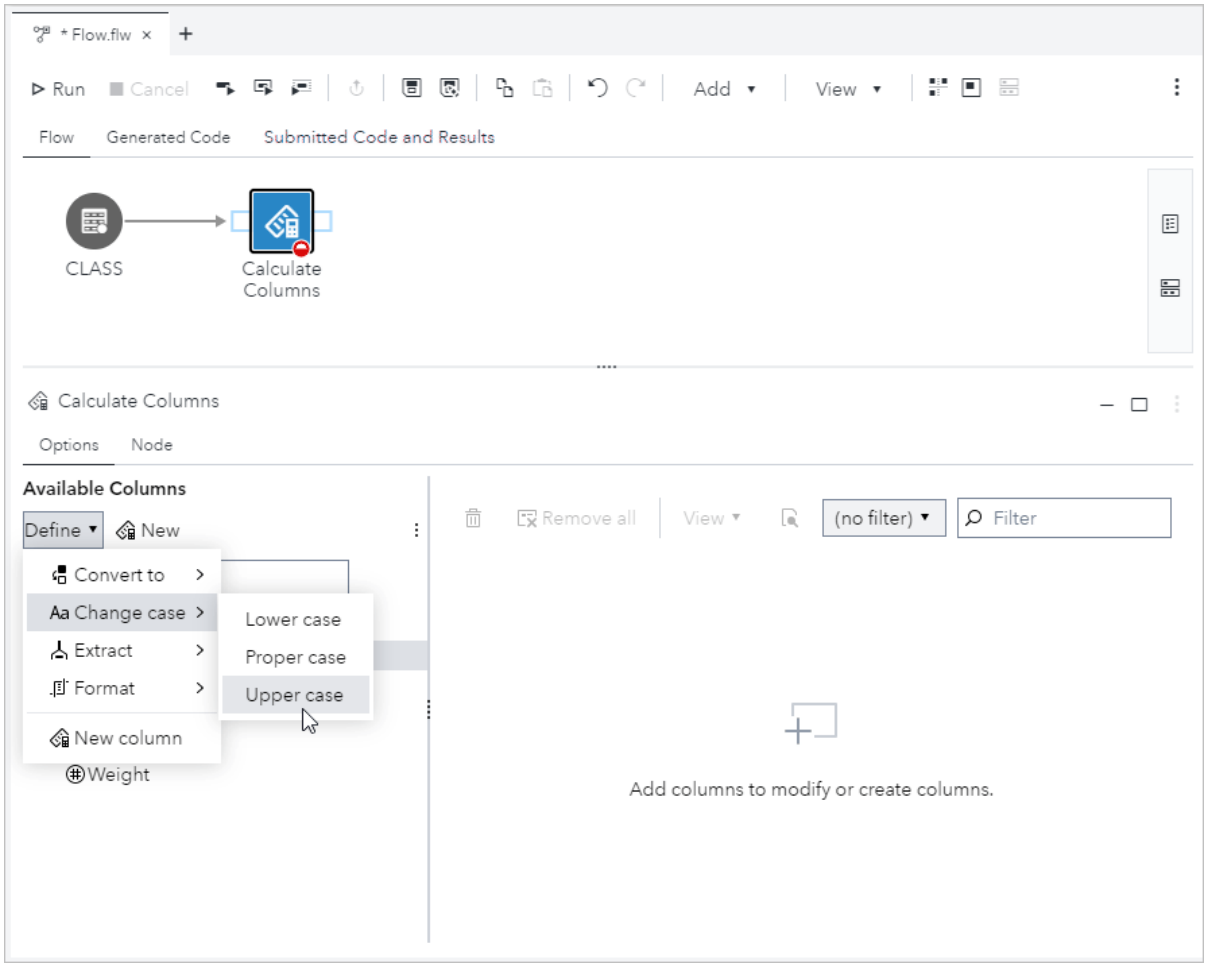


	Date	Price Change	Stock	Open	High	Low	Close
1	4	$6.95	IBM	$89.15	$89.92	$81.56	$82.20
2	4	$-7.05	IBM	$81.85	$89.94	$80.64	$88.90
3	4	$-1.66	IBM	$80.22	$84.60	$78.70	$81.88
4	3	$-0.06	IBM	$80.16	$82.11	$76.93	$80.22
5	3	$2.38	IBM	$83.00	$84.20	$79.87	$80.62
6	3	$-9.16	IBM	$74.30	$85.11	$74.16	$83.46
7	2	$1.37	IBM	$75.57	$77.73	$73.45	$74.20
8	2	$1.33	IBM	$76.88	$78.11	$72.50	$75.55
9	2	$15.11	IBM	$91.49	$91.76	$71.85	$76.38
10	1	$1.26	IBM	$92.64	$93.73	$89.09	$91.38

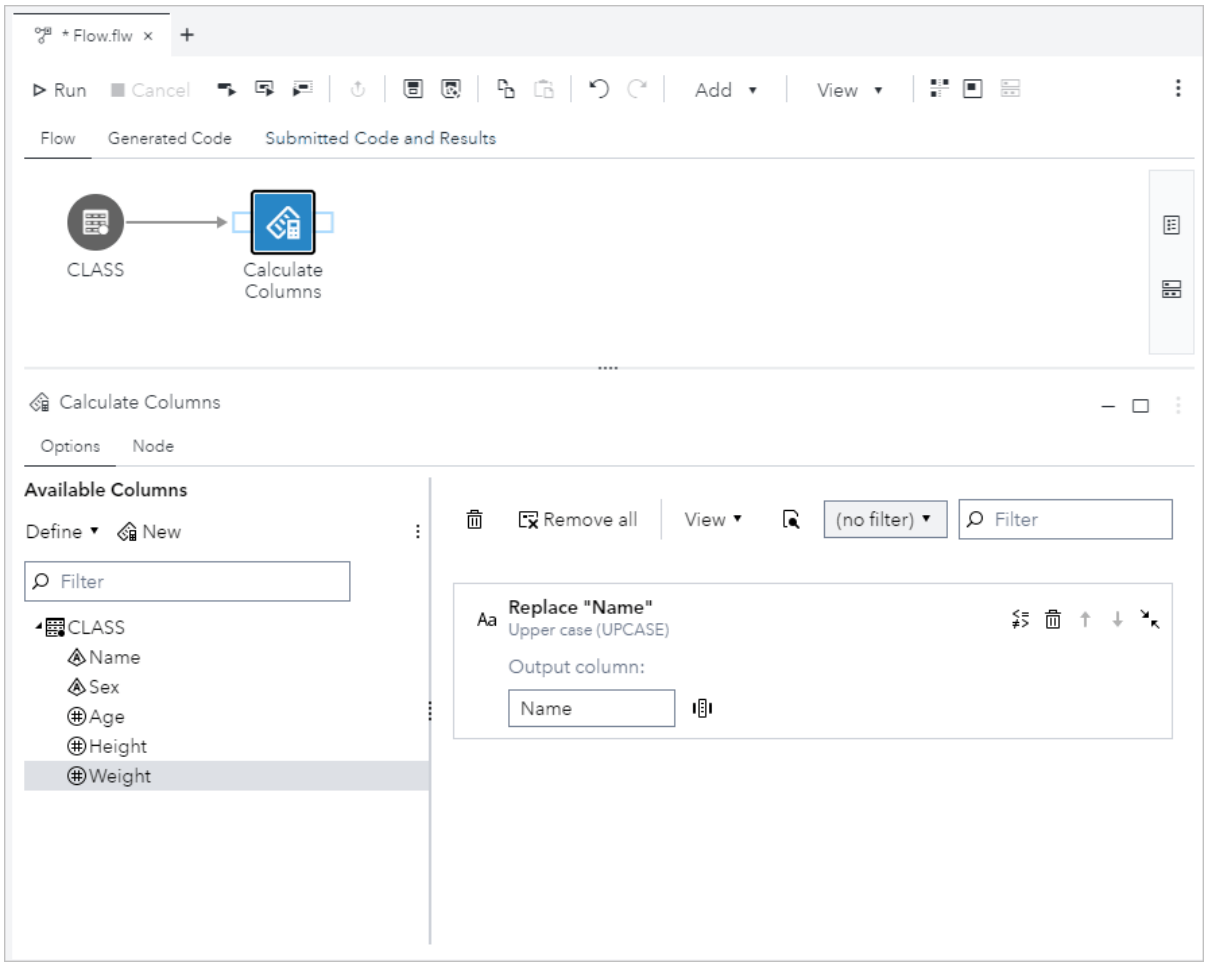
Calculate Columns: Step-by-Step Instructions

The Calculate Columns step requires one input port and one output port. You must select at least one column from the input table to replace in the output table or to use as the basis for creating a column.

1. In the **Steps** section of the navigation pane, expand the **Transform Data** folder and double-click **Calculate Columns**.
2. Connect the input port of the Calculate Columns 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 12.
3. Click the **Calculate Columns** node, and then use the Options tab to select a column to replace or use as the basis for a new column.
4. To replace a column:
   a. On the Options tab, select one or more columns that you want to replace. Click **Define** and select the function that you want to apply to the columns.



A card for each replaced column is added to the node calculations. The following figure shows a card that replaces the values in the Name column with uppercase values.



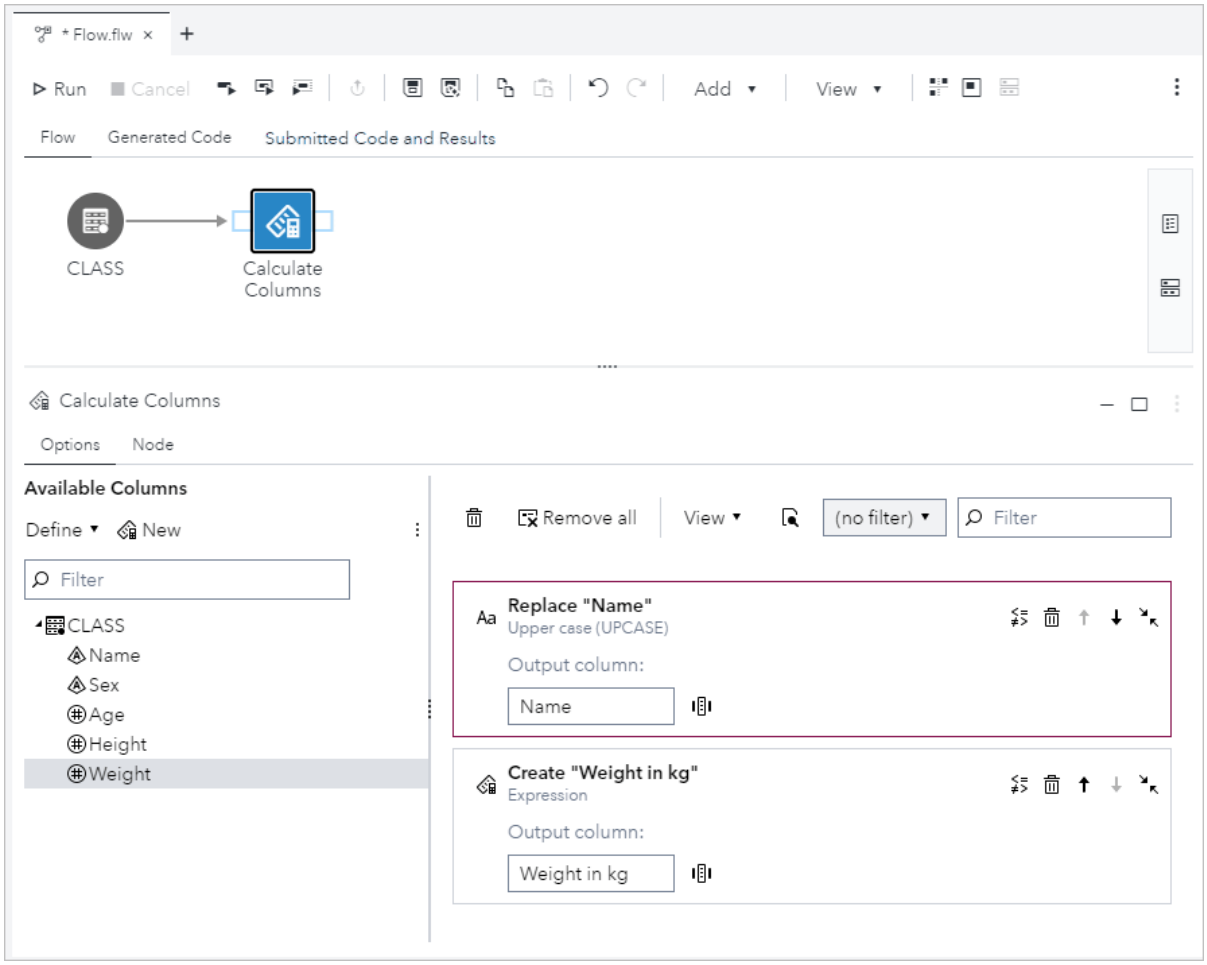
> **TIP** You can create a column in the output table by entering a new name for the output column in a Replace card. For example, if you create a card to replace the values in the Name column with uppercase values and rename the output column to `UpperName`, then the output table includes both the original Name column and the new UpperName column.

To create a column based on an existing column:

a On the Options tab, select the column that you want to use as the basis for the new column. Click **New**. The Expression Builder window opens.

b Use the Expression Builder options to create the expression for the new column, or enter the expression in the expression box.

c Use the Properties tab to specify the column name, data type, and other properties. Click **Save**. A Create card is added to the node calculations. The following figure shows a new card that is based on the Weight column and creates a column for weight in kilograms.



5 You can use the calculation cards to perform these actions:

	View or edit the expression by using the Expression Builder.
	View the output column properties.
	Delete the column calculation.
,	Move the calculation card up or down the list.
,	Expand or collapse a card.

Note: The calculation cards are processed by SAS Studio in order from top to bottom. You can move cards up and down the list to change the processing order.

> **TIP** You can use the Manage Columns step to change the order of the columns in the output table. For more information, see "Manage Columns Step: Subsetting Columns from an Input Table into an Output Table" on page 81.

6 To preview all the calculations that have been generated for the node, click . To copy the calculations to the clipboard, click **Copy**.

7 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 56.

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" in *SAS Studio: User's Guide*. 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

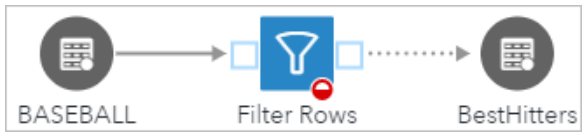
Example: Subsetting Data from the Sashelp.Baseball Data Set

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`.

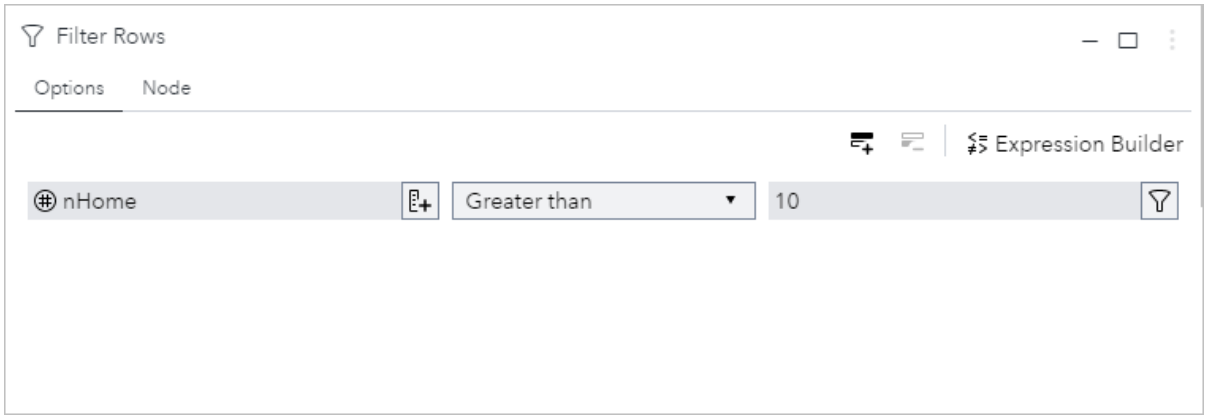
4. On the flow toolbar, click **Add** ⇨ **Table**. Connect the Filter Rows node to the Table node by clicking the Filter Rows output port and dragging the mouse pointer to the Table node.
5. Click the **Table** node. On the **Table Properties** tab, click  beside the **Library** box, and select the Work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

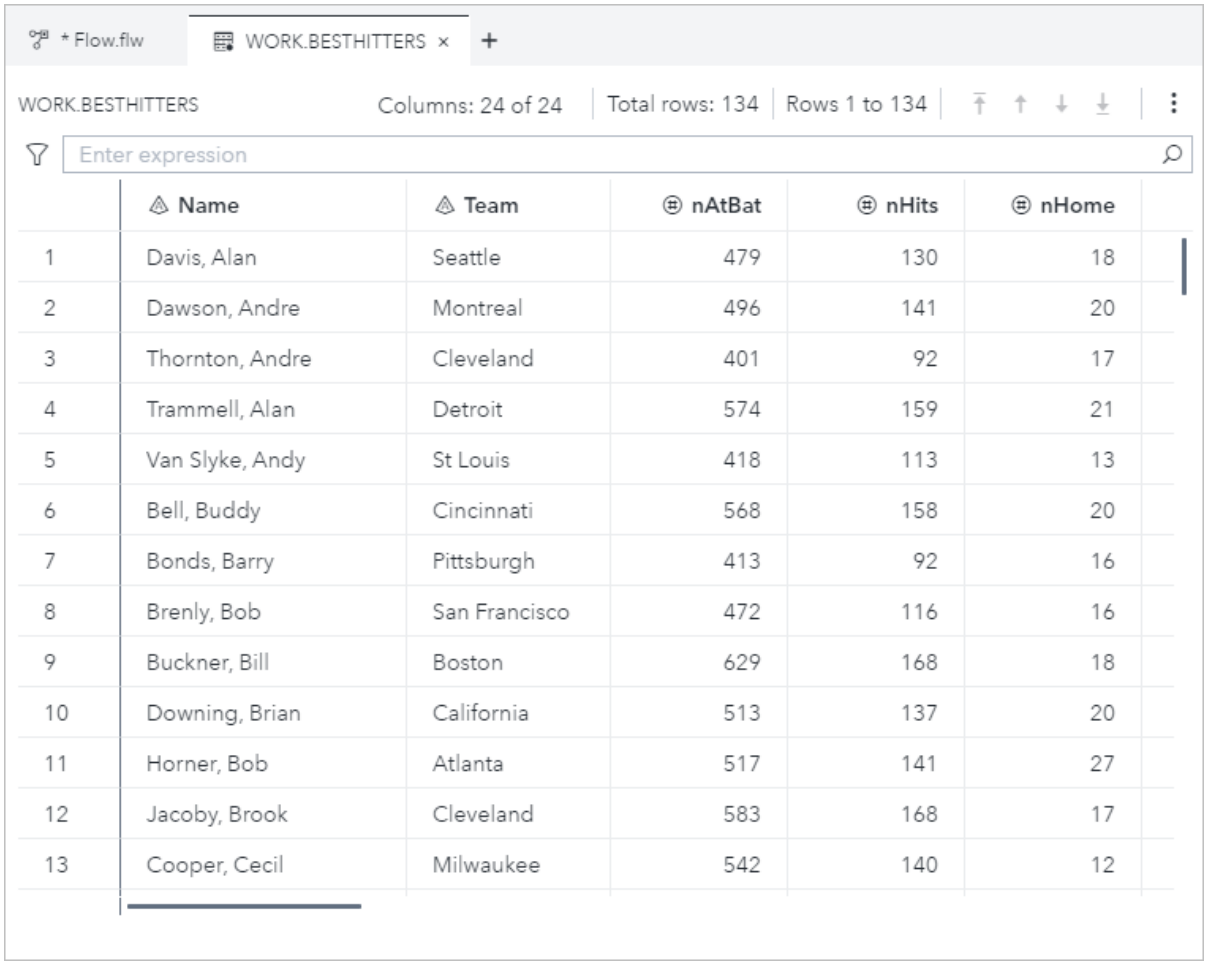
7 From the operator drop-down list, select `Greater than`, and then click  .

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

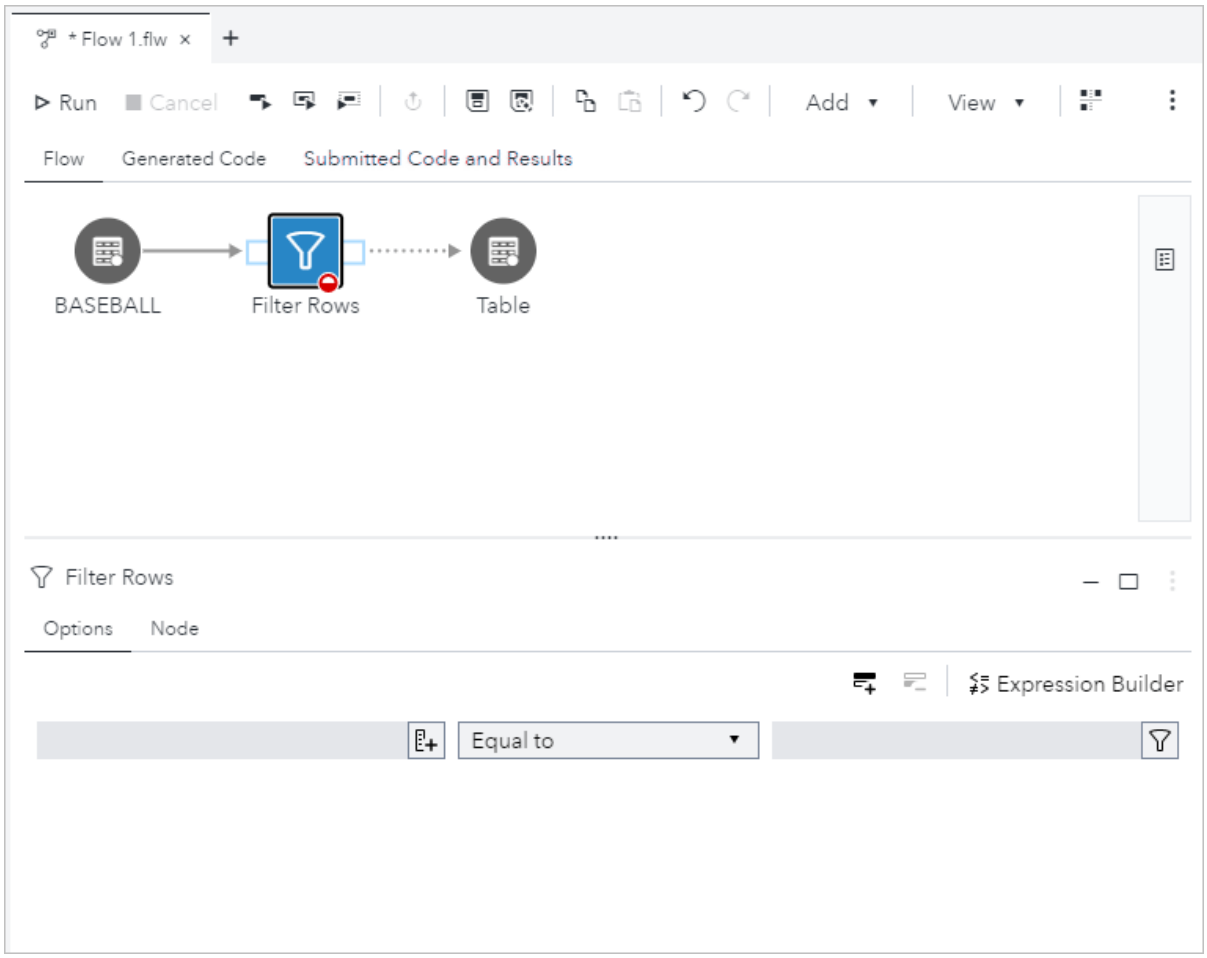
The Filter Rows step requires one input port and one output port. You can create one or more filter conditions to select a subset of the data that is written to the output table.

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 12.
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" in *SAS Studio: User's Guide*.

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: Inserting 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

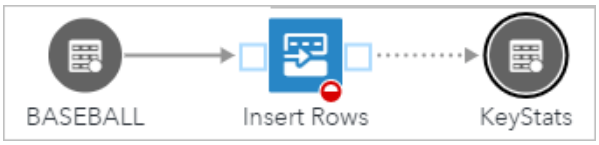
In this example, you insert rows from the Sashelp.Baseball data set into a new output table named KeyStats. The KeyStats table contains a subset of the columns that are in the Baseball data set.

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.

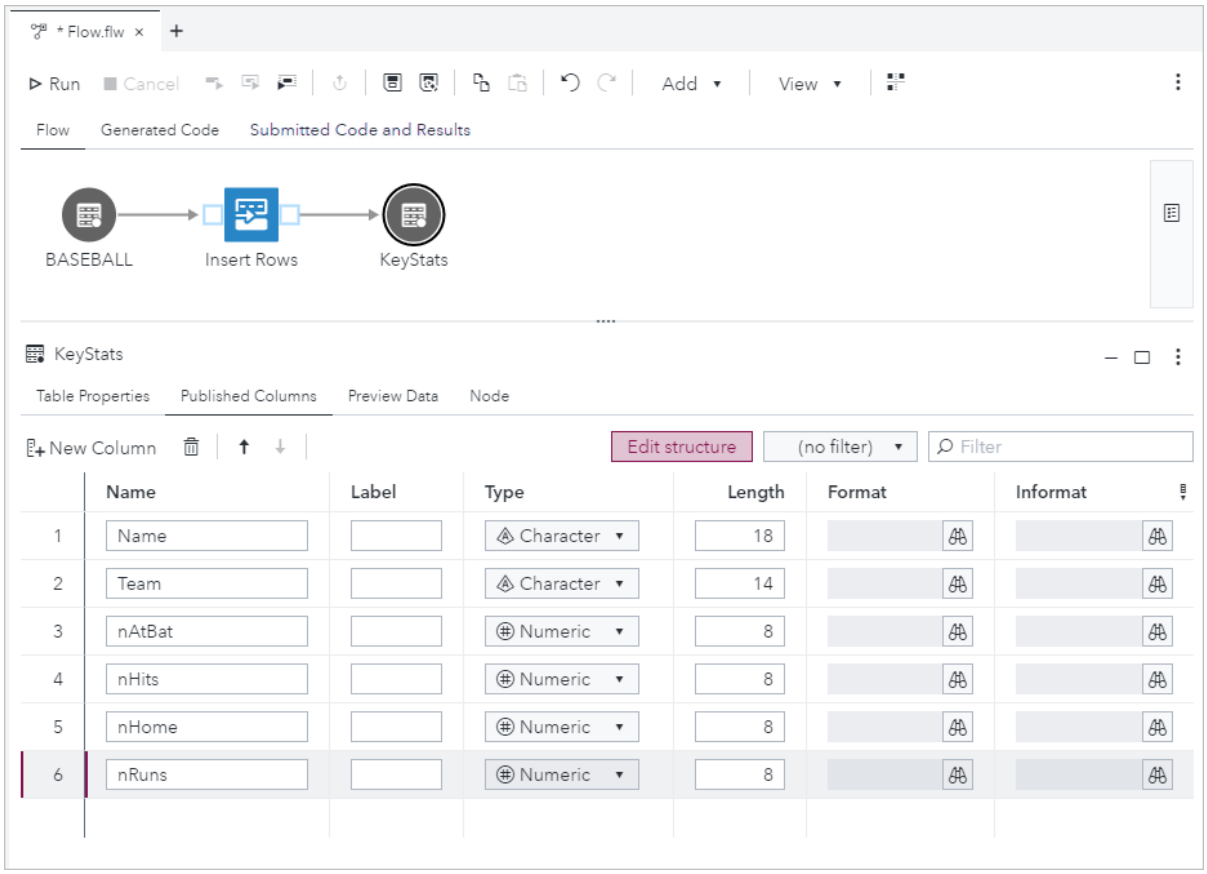
5 Click the **Table** node. On the **Table Properties** tab, click  beside the **Library** box, and select the Work library. In the Table box, enter *KeyStats*.



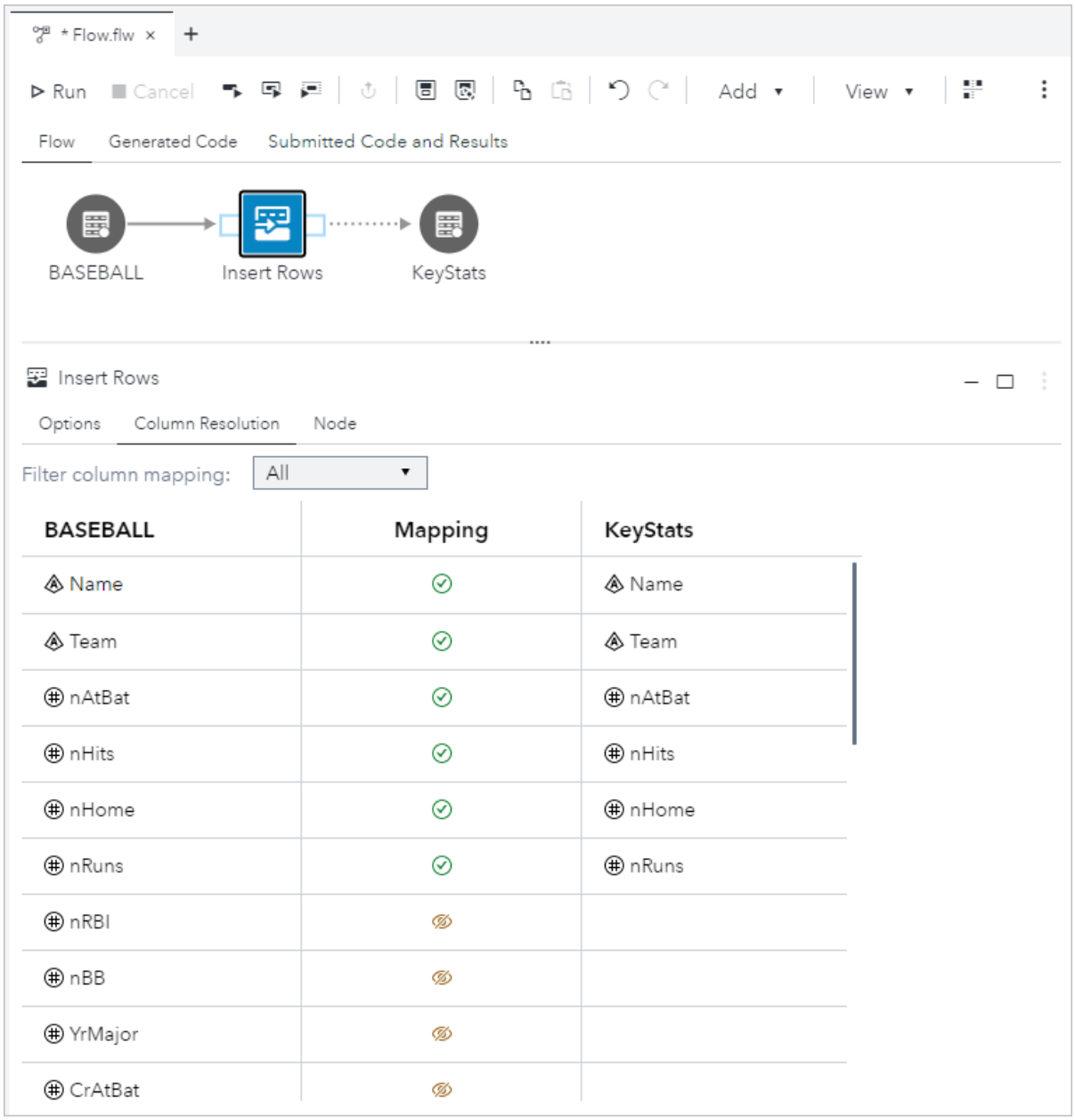
6 Click the **Published Columns** tab to add some columns from the Baseball data set. Change the table to edit mode by clicking **Edit structure**, and then clicking **Yes** in the confirmation window. Click **New Column** and enter *Name* in the **Name** column. Accept the default data type of `Character` and enter *18* for the Length. Continue adding five more columns by 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 in "About the Table Node" on page 15.

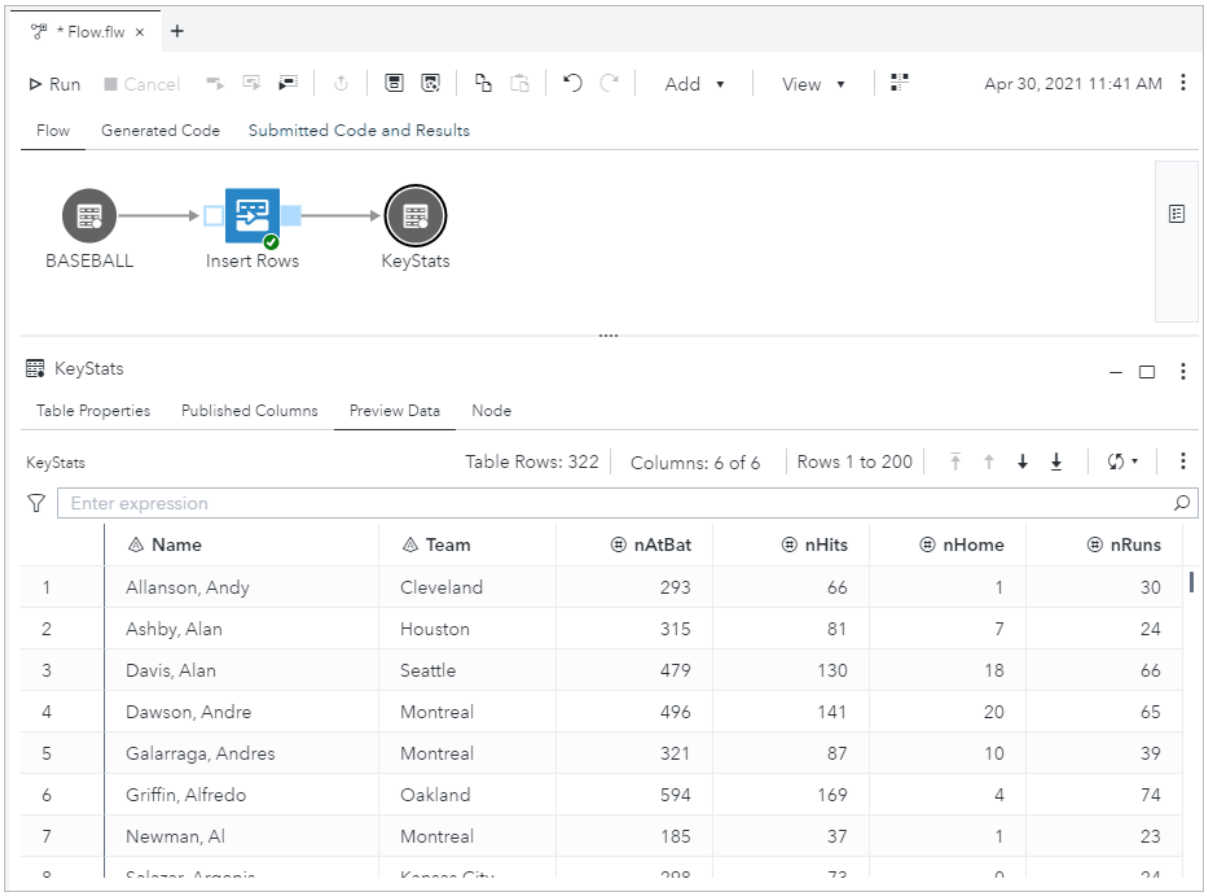


7 Click the **Insert Rows** node, and then click the **Column Resolution** tab. Notice that the columns in the KeyStats table are all mapped to columns in the Baseball data set. There are additional columns in the Baseball data set that do not map to columns in the KeyStats table.



8 To run the flow, click ▷ Run.

Here is the KeyStats output table:

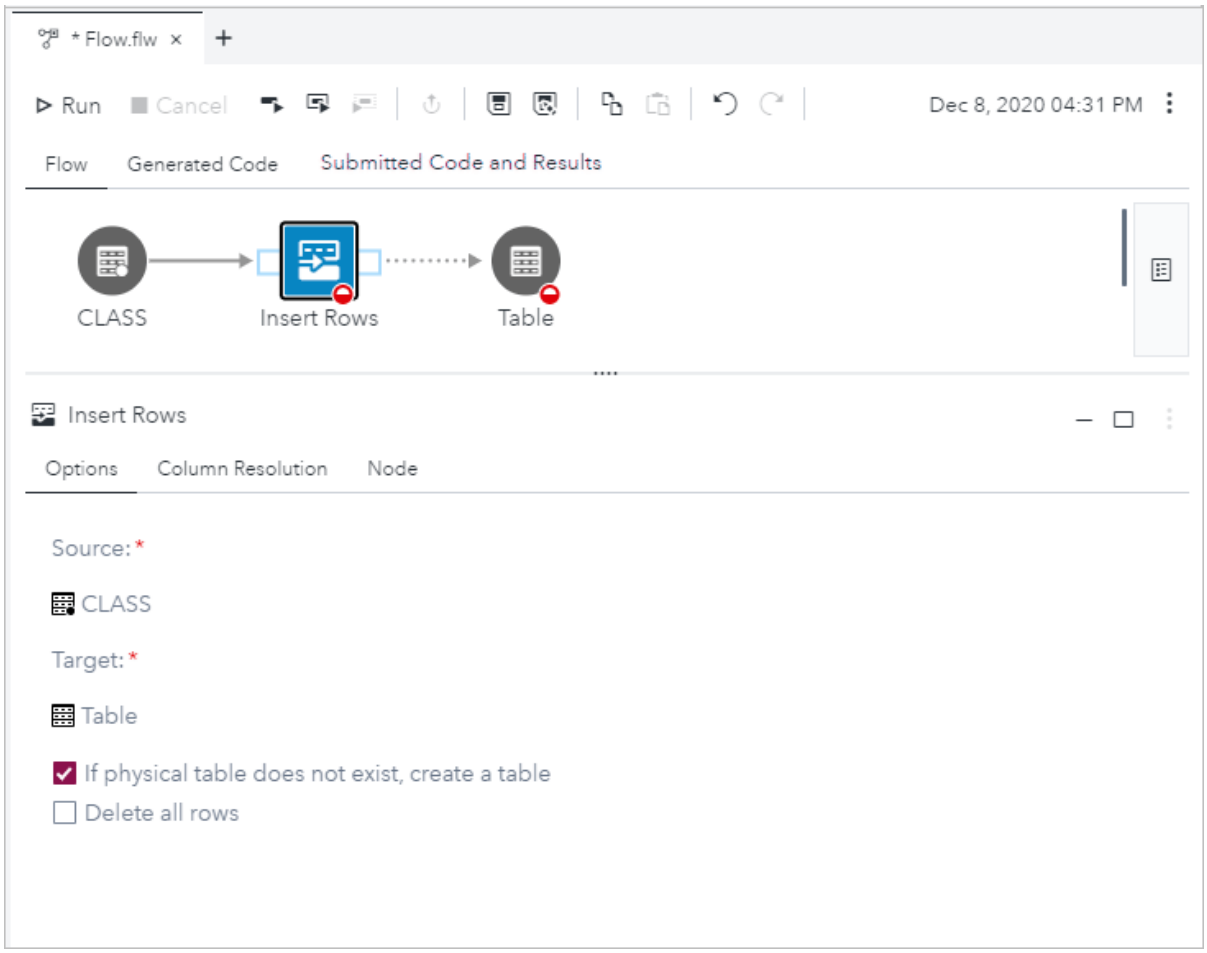


	Name	Team	nAtBat	nHits	nHome	nRuns
1	Allanson, Andy	Cleveland	293	66	1	30
2	Ashby, Alan	Houston	315	81	7	24
3	Davis, Alan	Seattle	479	130	18	66
4	Dawson, Andre	Montreal	496	141	20	65
5	Galarraga, Andres	Montreal	321	87	10	39
6	Griffin, Alfredo	Oakland	594	169	4	74
7	Newman, Al	Montreal	185	37	1	23

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 12.
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 34.

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 copy the column structure of the input table. For information about how to copy the entire column structure of the input table to the output table, see the Tip in "About the Table Node" on page 15.

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.
  - Information (ⓘ) - 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.

6 To run the flow, click ▷ Run.

Manage Columns Step: Subsetting Columns from an Input Table into an Output Table

About the Manage Columns Step

The Manage Columns step enables you to select a subset of columns from an input table and write the columns to an output table. You can also use the Manage Columns step to change the names, labels, and order of columns in the output table. The Manage Columns step can be combined with other steps in which you want to work with only a subset of the columns in a table.

For example, you might want to analyze only the career statistics of players in the Baseball table. You can use the Manage Columns step to create an output table that is based on the Baseball table and includes only the player names, teams, and the column names that begin with “Cr.” Reducing the number of columns in the data that you are analyzing can improve your performance.

Node Connection Requirements

In order to run the Manage Columns 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	No connection is required. **Note:** By default, the output data is written to a temporary table in the Work library. You can specify the library and name of the output tables by connecting the output port to a Table node. For more information, see "Adding a Table from a SAS Library to a Flow" on page 34.

Example: Subsetting Columns from the Sashelp.Baseball Data Set

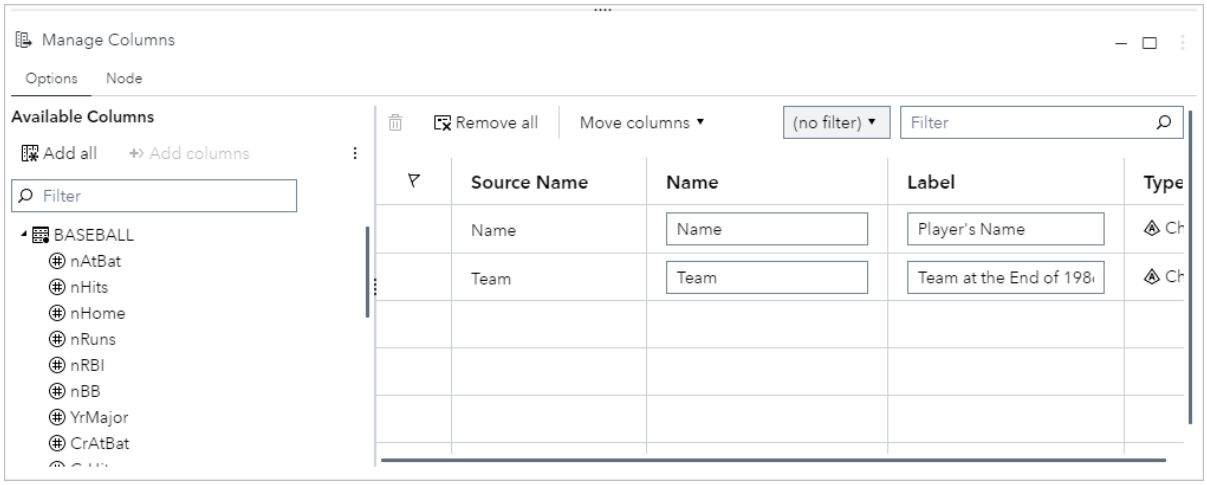
In this example, you select a subset of columns that contain the career statistics of players from the Sashelp.Baseball data set and create a new output table named CareerStats. The CareerStats table contains a subset of the columns and all of the rows that are in the Sashelp.Baseball data set.

To create this example:

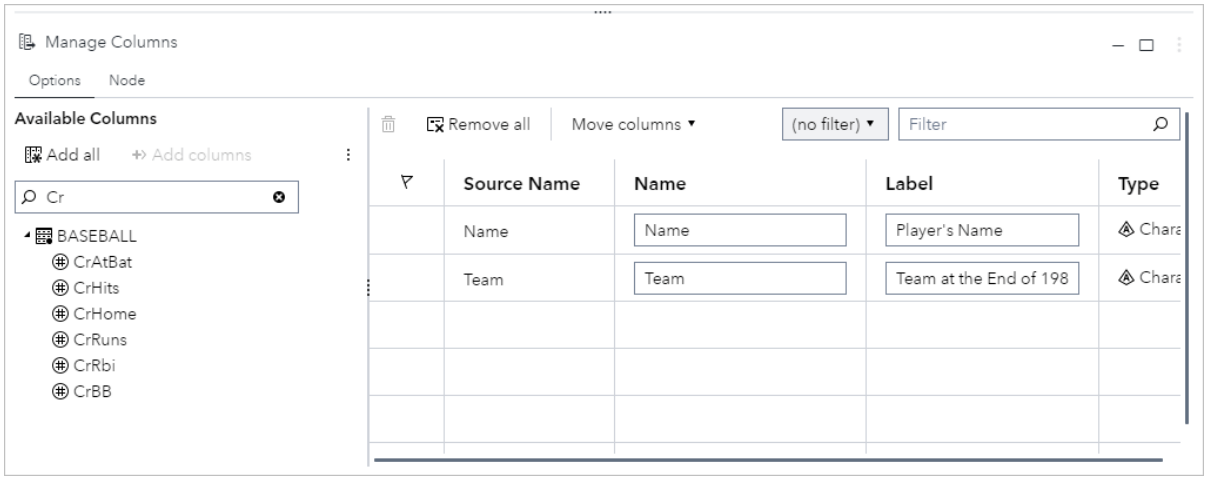
1. From the main SAS Studio menu, select **New** ⇨ **Flow**.
2. In the **Steps** section of the navigation pane, expand the **Transform Data** folder and double-click **Manage Columns**.
3. In the **Libraries** section of the navigation pane, expand the Sashelp library. Drag the Baseball data set onto the input port of the Manage Columns 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 Manage Columns node to the Table node by clicking the Manage Columns 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 then select the Work library. In the Table box, enter `CareerStats` and click **OK**.

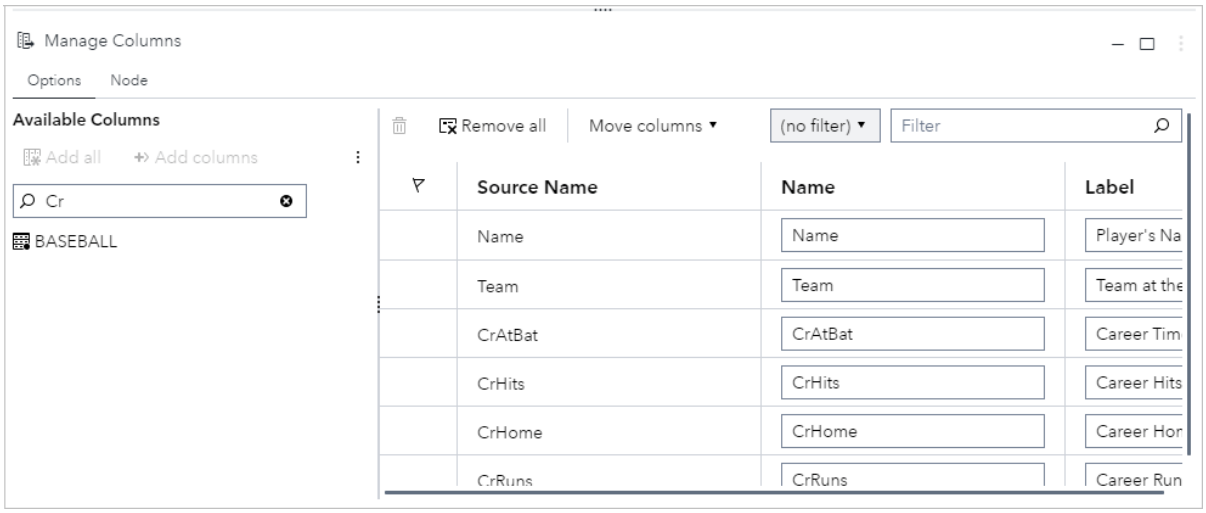
6 Click the Manage Columns node to specify columns for the output table. On the Options tab, select the **`Name`** and **`Team`** columns in the Available Columns area and click **Add columns**.



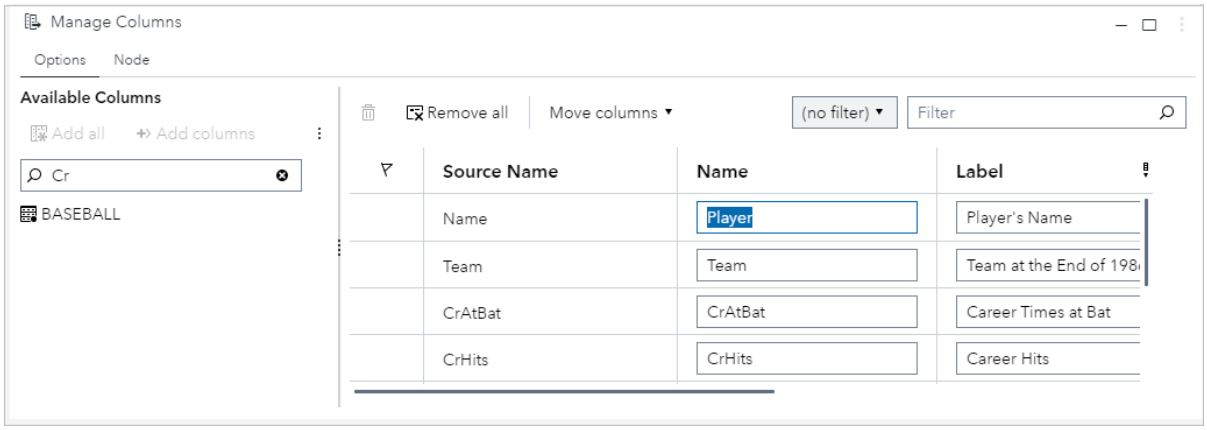
7 In the Available Columns area, enter *Cr* in the Filter box. The list of columns is filtered and displays only the career statistics columns that start with **`Cr`**.



8 Click **Add all** to add all of the visible columns to the list of columns for the output table.

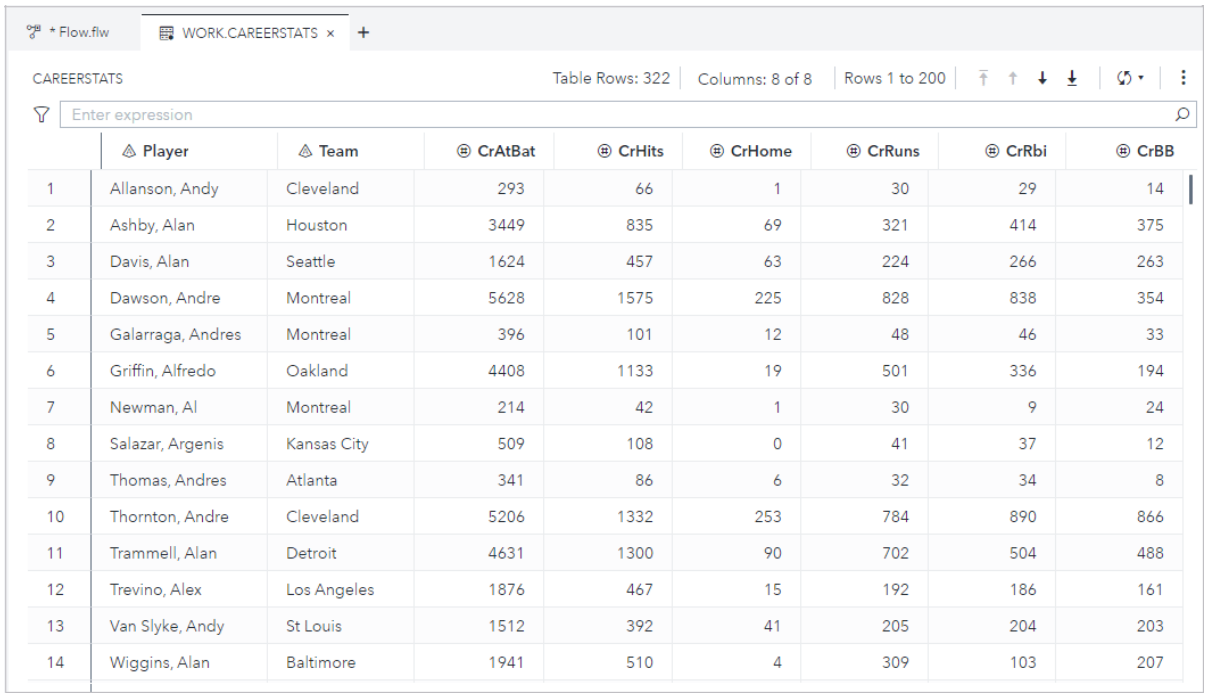


9 Change the Name column to Player by entering *Player* as the new name value.



10 To run the flow, click ▷ Run.

Here is the output table that includes all of the rows from the Sashelp.Baseball data set and only the Player, Team, and career statistics columns.



	Player	Team	CrAtBat	CrHits	CrHome	CrRuns	CrRbi	CrBB
1	Allanson, Andy	Cleveland	293	66	1	30	29	14
2	Ashby, Alan	Houston	3449	835	69	321	414	375
3	Davis, Alan	Seattle	1624	457	63	224	266	263
4	Dawson, Andre	Montreal	5628	1575	225	828	838	354
5	Galarraga, Andres	Montreal	396	101	12	48	46	33
6	Griffin, Alfredo	Oakland	4408	1133	19	501	336	194
7	Newman, Al	Montreal	214	42	1	30	9	24
8	Salazar, Argenis	Kansas City	509	108	0	41	37	12
9	Thomas, Andres	Atlanta	341	86	6	32	34	8
10	Thornton, Andre	Cleveland	5206	1332	253	784	890	866
11	Trammell, Alan	Detroit	4631	1300	90	702	504	488
12	Trevino, Alex	Los Angeles	1876	467	15	192	186	161
13	Van Slyke, Andy	St Louis	1512	392	41	205	204	203
14	Wiggins, Alan	Baltimore	1941	510	4	309	103	207

Manage Columns: Step-by-Step Instructions

The Manage Columns step requires one input port and one output port. You must select at least one column for the output table.

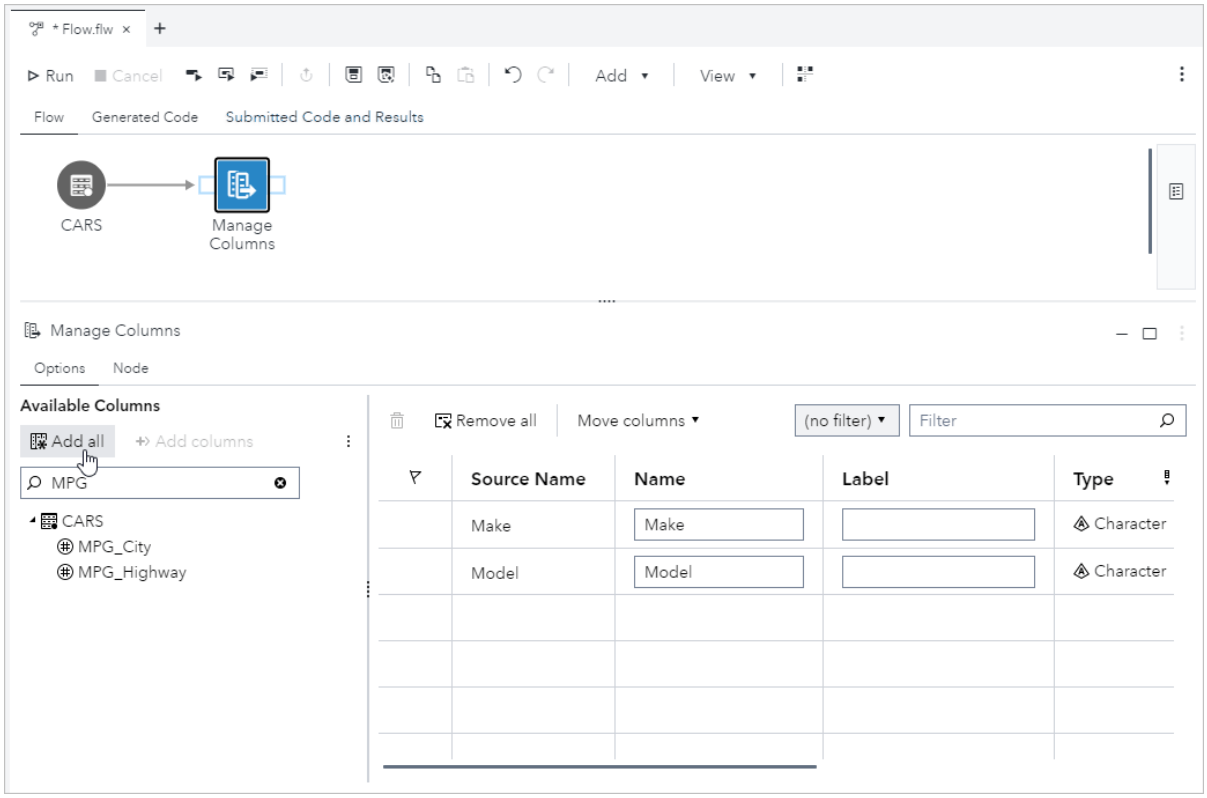
1 In the **Steps** section of the navigation pane, expand the **Transform Data** folder and double-click **Manage Columns**.

2 Connect the input port of the Manage Columns 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 12.

3 Click the **Manage Columns** node, and then use the Options tab to specify one or more columns for the output table. To add one or more columns, select the columns that you want to add to the output table, and click **Add columns**. To add all of the columns that are currently displayed in the Available Columns area, click **Add all**.

> **TIP** You can specify filter criteria in the filter box, and then click **Add all** to add all of the columns that match your filter criteria.



4 In the right pane of the Options tab, you can modify the columns that are selected for the output table in the following ways:

- To filter the columns that are displayed, click the filter drop-down list and select the filter criteria. You can also enter filter criteria for the Source Name, Name, and Label columns in the filter box.

  Note: Filters affect only which columns are displayed in the right pane. All columns in the right pane are added to the output table even if they are not displayed because of a filter.

- To change the order in which the columns appear in the output table, select the column that you want to move. Click **Move columns** and select the appropriate option. The order in which the columns are listed in the right pane of the Options tab determines the order in which the columns appear in the output table.

- To change the name or label of a column in the output table, use the **Name** and **Label** boxes.
- To remove columns from the right pane, select one or more columns and click 🗑. To remove all columns that are currently displayed in the right pane, click **Remove all**.

5 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 56.

The Query node interface is mostly the same as the interface for stand-alone queries. For more information, see "Working with Queries" in *SAS Studio: User's Guide*.

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.

Removing Duplicates

The Remove Duplicates node enables you to remove duplicate rows from an input data source.

- If your input data and output data are in a SAS library, the step runs the SORT procedure by using the NODUPKEY option to remove the duplicate data.
- If your input data and output data are in a CAS library, the step uses the simple.GroupByInfo action to remove the duplicate data.

To remove duplicates from a data source:

1. Click the **Steps** section of the navigation pane.
2. Expand the **Transform Data** folder and double-click the **Remove Duplicates** node.
3. From the **Libraries** section of the navigation pane, drag the table that you want to sort onto the **Remove Duplicates** node. When the tooltip changes to `Connect to input port`, drop the file.
4. Click the **Remove Duplicates** node. Select the **Remove duplicates across all columns** check box. Next, select the columns to group by to determine whether there are duplicate rows in the data.
5. Specify these output options:
   - **Replace existing output table with the same name**
   - For CAS output tables:
     - Specify whether to promote the table and save the table.
     - (Optional) Specify the file format (in lowercase) for the output table.

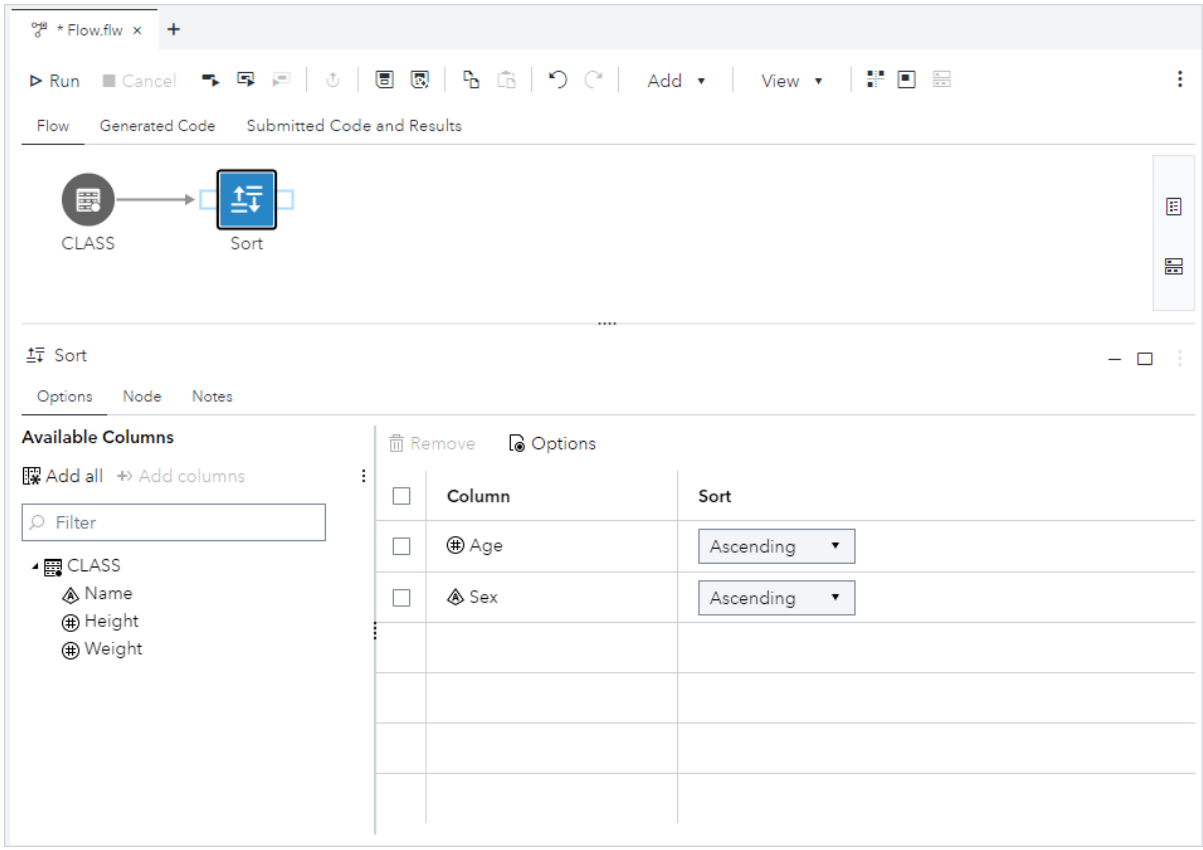
Sorting Data

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 one or more columns, select the columns that you want to add to the sort, and then click **Add columns**. To add all of the columns, click **Add all**.

   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.

Transpose Data

About the Transpose Data Step

The Transpose Data step turns selected columns of an input table into the rows of an output table. If you do not use grouping variables, then each selected column is

turned into a single row. If you use grouping variables, then the selected columns are divided into subcolumns based on the values of the grouping variables. Each subcolumn is turned into a row of the output table.

Node Connection Requirements

In order to run the Transpose Data 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	No connection is required. **Note:** By default, the output data is written to a temporary table in the Work library. You can specify the library and name of the output tables by connecting the output port to a Table node. For more information, see "Adding a Table from a SAS Library to a Flow" on page 34.

Transpose Data: Assigning Data to Roles

To run the One-Way Frequencies step, you must assign a column to the **Variables to transpose** role.

To filter the input data source, type the filter expression in the **Filter** field.

Table 5.2 *Roles in the Transpose Data Step*

Role	Description
Roles	
Variables to transpose	Each variable that you assign to this role becomes one or more rows of the output table. If you do not select any grouping variables, then an entire column is turned into a single row. If you select one or more grouping variables, then the grouping variables are used to segment each column into subcolumns, each of which is turned into a row. In this case, a column is transposed to the number of rows that is equal to the number of groups that are defined by the grouping variables.

Role	Description
	You must assign at least one column to the **Variables to transpose** role. To select a grouping variable, assign a column to the **Group analysis by** role.
Additional Roles	
Group analysis by	Each variable that you assign to this role is used to segment the about-to-be-transposed columns into subcolumns that will be transposed separately. Each subcolumn, defined by a set of values of the grouping variables, becomes a row of the output table.

Transpose Data: Setting Options

On the **Options** tab, you can set these options.

Table 5.3 *Names and Labels of Transposed Variables*

Option Name	Description
Construct New Variable Names	
Use prefix	You can specify a prefix to use in constructing the names for the transposed variables in the output data set. When you use a prefix, the variable name begins with the prefix value and is followed by the number 1, 2, and so on.
Select a variable that contains the names of the new variables	The variable that you assign to the **New column names** role is used to name the transposed variables in the output data set. If you specified to use a prefix in the name, the name for the new variable begins with the prefix and is followed by the value of the **New column names** variable. If you select the **Allow duplicate of ID values** check box, the transposed output data set contains only the last observation for each BY group.
Construct New Variable Labels	

Option Name	Description
Select a variable that contains the labels of the new variables	The values of the variable that you assign to the **New column labels** role are used to label the variables in the output data set.

Table 5.4 *Names and Labels of Original Variables*

Option Name	Description
Put original variable names in a new variable	Each row of the output table includes the name of the variable in the input table to which the values in that output row belong. To specify a heading for the output column that contains these variable names, enter the heading in the **Name** box. The name can include special characters, leading numbers, and white space, but it cannot exceed 32 characters. The default name is _Name_.
Put original variables labels in a new variable	Each row of the output table includes the label of the variable in the input table to which the values in that output row belong. To specify a heading for the output column that contains these variable labels, enter the heading in the **Label** box. The label can include special characters, leading numbers, and white space, but it cannot exceed 32 characters. The default label is _Label_.

Transpose Data: Specifying the Output Data

On the **Output** tab, you can set these options.

Table 5.5 *Output Data Options*

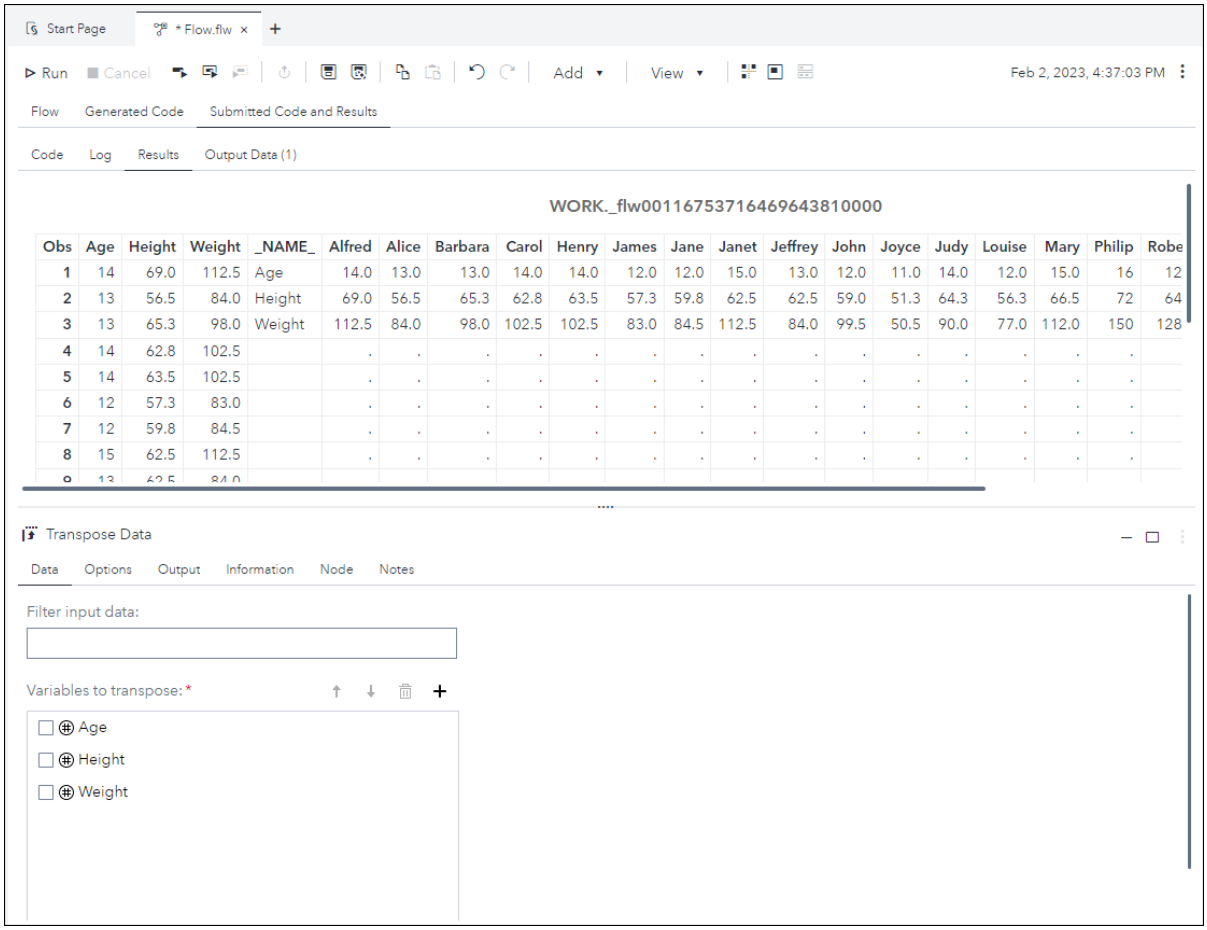
Option Name	Description
Replace existing output table	When selected, SAS Studio replaces an existing output table with the same name.
Copy to output data	Each variable that you assign to this role is copied directly from the input table to the output table without being transposed. Because these columns are copied

Option Name	Description
	directly to the output table, the number of rows in the output table equals the number of rows in the input table. The output table is padded with missing values if the number of rows in the input table does not equal the number of variables that it transposes.
Specify data to print	You can specify whether to print none of the data, a subset of the observations, or all the observations.

Example: Transposing Data in the CLASS Data Set

1. In the **Steps** section of the navigation pane, expand the **Transform Data** folder and double-click **Transpose Data** to add the step to your flow.
2. Add the Sashelp.Class table to your flow. Connect the input port of the Transpose Data node to the data source. For more information, see “Connecting Nodes” on page 12.
3. To set the options for the Transpose Data step, click the Transpose Data node.
4. On the **Data** tab, assign the **Age**, **Height**, and **Weight** variables.
5. On the **Options** tab, complete these steps:
   - Clear the **Use prefix** check box.
   - Select the **Select a variable that contains the names of the new variables** check box.
   - To the **New column names** role, assign the **Name** variable.
6. Run the flow.

Open the **Submitted Code and Results** tab to view the results.



To view the generated SAS code:

1. Open the **Log** tab.
2. Search for MPRINT_CODE.


Code
Log
Output Data (1)
Errors (0)
Warnings (0)
Notes (13)
There are no messages.
696 /* Main entry point: All the work is done in the macro. */
697 %let __locallyDefinedEntryPointMacro = 0;
698 %if %sysmacexist(__entryPoint) = 0 %then %do;
699 %let __locallyDefinedEntryPointMacro = 1;
700 %macro __entryPoint();
701 %runStep()
702 %mend __entryPoint;
703 %end;
704
705 options mprint;
706 %__entryPoint()
MPRINT(__CODE): proc transpose data=SASHELP.CLASS out=WORK._flw001167544970699549800000;
MPRINT(__CODE): var Age Height Weight;
MPRINT(__CODE): id Name;
MPRINT(__CODE): run;
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK._FLW001167544970699549800000 has 3 observations and 20 variables.
NOTE: PROCEDURE TRANSPOSE used (Total process time):
real time 0.01 seconds
cpu time 0.04 seconds

6

Integrating Data

Execute Decisions: Running a Published Decision

About the Execute Decisions Step

You can use the Execute Decisions step to add a published decision from SAS Intelligent Decisioning to your flow. Decisions enable you to create a database of rules, combine those rules into decisions, and publish the decisions for use by other

applications such as SAS Studio. For more information, see "Introduction to SAS Intelligent Decisioning" in *SAS Intelligent Decisioning: User's Guide*.

To use a decision in SAS Studio, the decision must meet the following criteria:

- The decision must be published using SAS Intelligent Decisioning.
- The decision must be published to a CAS destination.
- The input table must be a CAS table.

Note: By default, the output data is written to the output port as a session-scoped CAS table in the same CAS library as the source table.

Note: This step is available only if your site licenses SAS Studio Engineer.

Node Connection Requirements

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

Input Port	Output Port
■ Table node **Note:** The input table for the Execute Decisions node must be a CAS table.	No connection is required. **Note:** Execute Decisions node output can be connected to any flow node that accepts a SAS or CAS table as an input, such as a Table node, a Query node, or a SAS Program node. By default, the output data is written to a session-scoped CAS table in the same CAS library as the source table. You can specify the library and name of the output tables by connecting the output port to a Table node. It is recommended that a CAS output table reside in the same CAS session as the input table.For more information, see "Adding a Table from a SAS Library to a Flow" on page 34.

Execute Decisions: Step-by-Step Instructions

The Execute Decisions step requires one input port and one output port.

1. In the **Steps** section of the navigation pane, expand the **Integrate** folder and double-click **Execute Decisions**.
2. Click the **Execute Decisions** node, and then use the Options tab to select the decision that you want to use. To select a decision, click . In the **Select a Published Decision** box, specify the CAS destination and decision that you want to use.

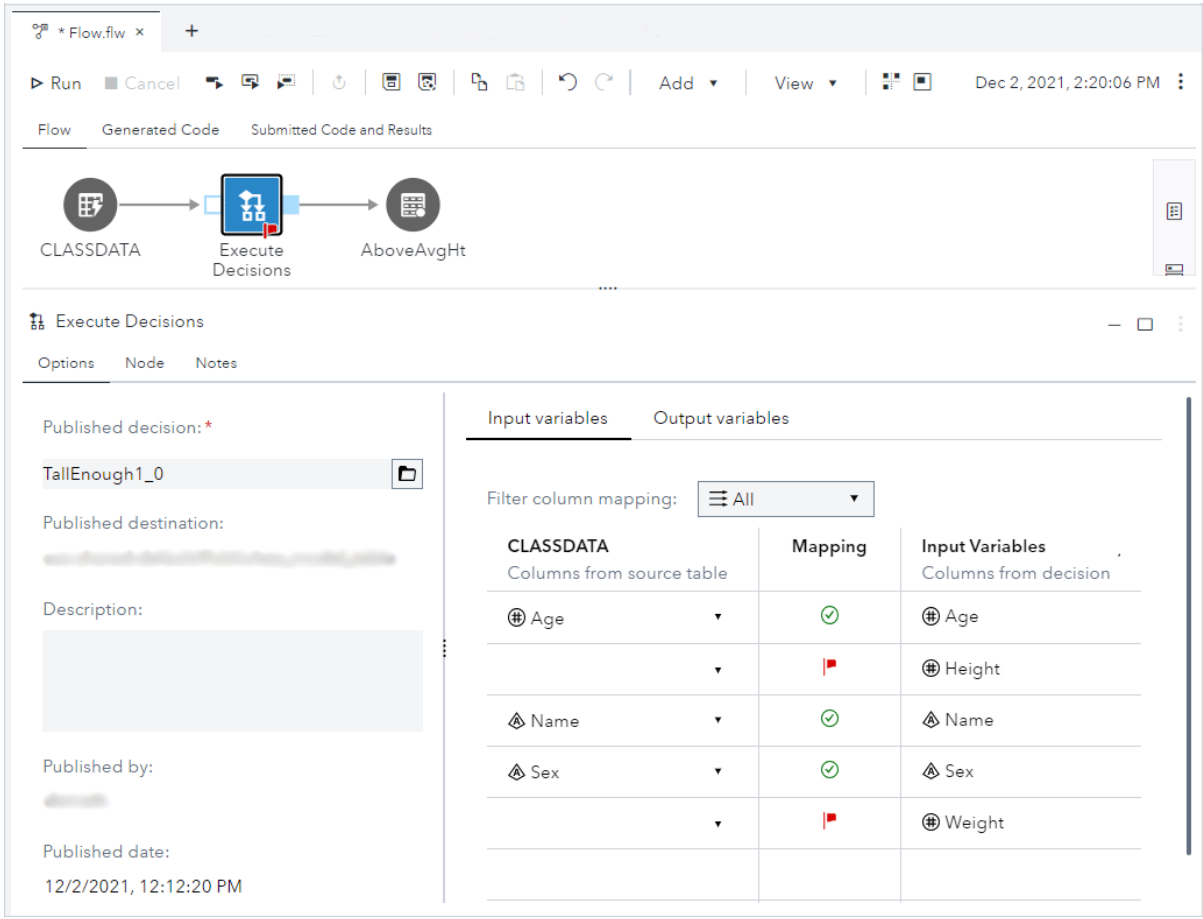
3 Connect the input port of the Execute Decisions node to the appropriate data source for the selected decision. The input data source must be a Table node that references a CAS table. For more information, see "Connecting Nodes" on page 12.

4 Click the **Execute Decisions** node to verify the column mapping for the decision. In the right pane of the Options tab, use the Input variables tab to view the mapping between the columns in the input table and the columns that are required by the decision. Columns are mapped automatically by name and data type.

Note: If the length of a column in the input table is different from the expected length of the corresponding column in the decision, the length of the column in the decision output table is overwritten, by default. If the length of an input column is shorter than the length of the output column, values in the output table can be truncated. You can use the `sas.decisions.variableLengthOverridden` property in SAS Environment Manager to keep the length of the output column. For more information, see "sas.decisions.variableLengthOverridden" in *SAS Intelligent Decisioning: Administrator's Guide*.

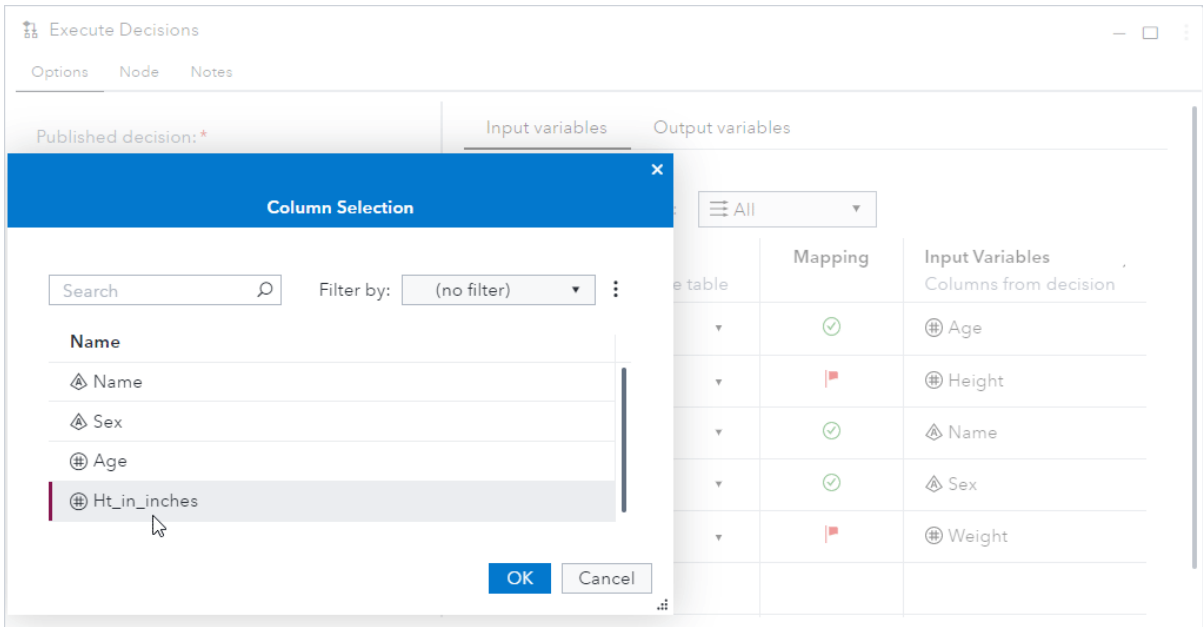
Use the **Filter column mapping** drop-down list to filter the mappings that are displayed. You can filter the mappings by the following categories:

- Successful (✓) - the input column is successfully mapped to a column that is used in the decision.
- Unresolved (⚑) - an input column is not yet mapped to the column in the decision. An unresolved column can occur when SAS Studio cannot automatically map columns based on name and data type. You must manually map any unresolved column mappings.



5 If you have any unresolved column mappings, click ▾ beside the unmapped column and select the appropriate column to map to the column in the decision.

Note: The Execute Decisions node can run only when all columns are mapped successfully.



6 To view the output columns for the decision, click the **Output variables** tab.

7 To run the flow, click ▷ Run.

Implement SCD: Storing and Managing Data over Time

About the Implement SCD Step

The Implement SCD step enables you to use the slowly changing dimensions (SCD) process to load data into target dimension tables. The data changes slowly, rather than changing on a time-based, regular schedule. The target tables are structured so that they can retain a history of changes to their data. This record of data changes can provide a basis for analysis. You can also choose to overwrite your data with new values and not preserve the historical data.

For example, a table named Customers might have columns for customer ID, home address, age, and income. Each time the address or income changes for a customer, a new row could be created for that customer, and the old row could be retained. This historical record of changes could be combined with purchasing information to forecast buying trends and to direct customer marketing campaigns.

SAS Studio supports two types of slowly changing scenarios:

- Type 1 SCD - no history of data changes. Type 1 SCD overwrites the specified columns in the target table without retaining a history of changes. Type 1 SCD is useful for maintaining columns that are not used in historical analysis.
- Type 2 SCD - history of data changes is maintained. Type 2 SCD maintains multiple records for each business key in the target table. The latest entry is the current entry for that business key. Other rows comprise the historical record of data changes.

Note: This step is available only if your site licenses SAS Studio Engineer.

There are seven main steps to use the Implement SCD step:

1 Add the Implement SCD step to your flow and connect the source table.

2 Specify the library and name of the target table.

3 Specify one or more key columns that are used to match the rows between the source and target tables.

4 (Optional) Specify columns from the target table for SCD Type 1 changes. The selected columns are updated with values from the source table.

5 (Optional) Specify columns from the target table for SCD Type 2 changes. The selected columns are used in the new current row that is created. The columns are populated with data from the corresponding columns in the source table.

6 (Optional) Specify columns, which have not been selected for SCD Type 1 or Type 2 changes, to include when a new row is added to the target table.

7 Run the flow to update the target table.

Note: In order to run the Implement SCD step, you must specify either SCD Type 1 or SCD Type 2 changes or both.

Node Connection Requirements

The Implement SCD node includes only one input port. The target table is specified in the node details rather than with an output port. In order to run the Implement SCD step, you must create a connection to the input port of the node as indicated here:

Input Port	Output Port
■ Table node. The source and target tables must reside on the same database server. The tables must reside in one of the following databases: □ Oracle □ Singlestore □ Snowflake □ SQL Server □ Teradata	Not applicable. The target table is specified in the node details.

Implement SCD: Step-by-Step Instructions

Step 1: Add the Implement SCD Step and Connect a SourceTable

To add the Implement SCD step to your flow:

1 In the **Steps** section of the navigation pane, expand the **Integrate** folder and double-click **Implement SCD**.

2 Connect the input port of the Implement SCD node to a data source by using a Table node. The table must reside in one of the following databases: Oracle, Singlestore, Snowflake, SQL Server, or Teradata. For more information, see "Connecting Nodes" on page 12.

Note: If you are connecting to Singlestore data, you must specify `S2` as the SAS/ACCESS engine name and set DEFAULT_AUTH_PLUGIN=mysql_native_password in the LIBNAME statement. For more information, see "DEFAULT_AUTH_PLUGIN= LIBNAME Statement Option" in *SAS/ACCESS for Relational Databases: Reference*.

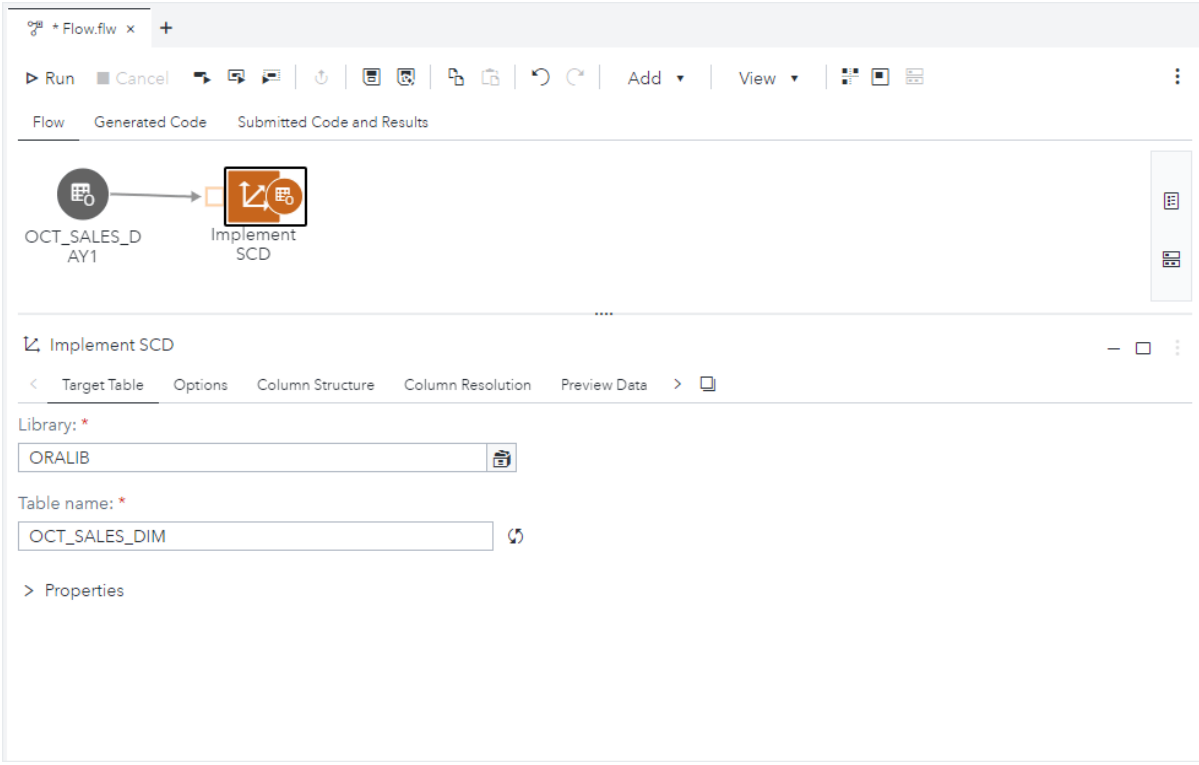
Step 2: Specify the Target Table

The source and target tables must reside on the same database server.

1 Click the **Implement SCD** node, and then click the **Target Table** tab.

2 Use the **Library** and **Table name** boxes to specify the target table.

Note: The Implement SCD node does not have an output port. You must specify the target table in the Implement SCD node details. However, you can connect the Implement SCD node to another operational node that requires a table as input.



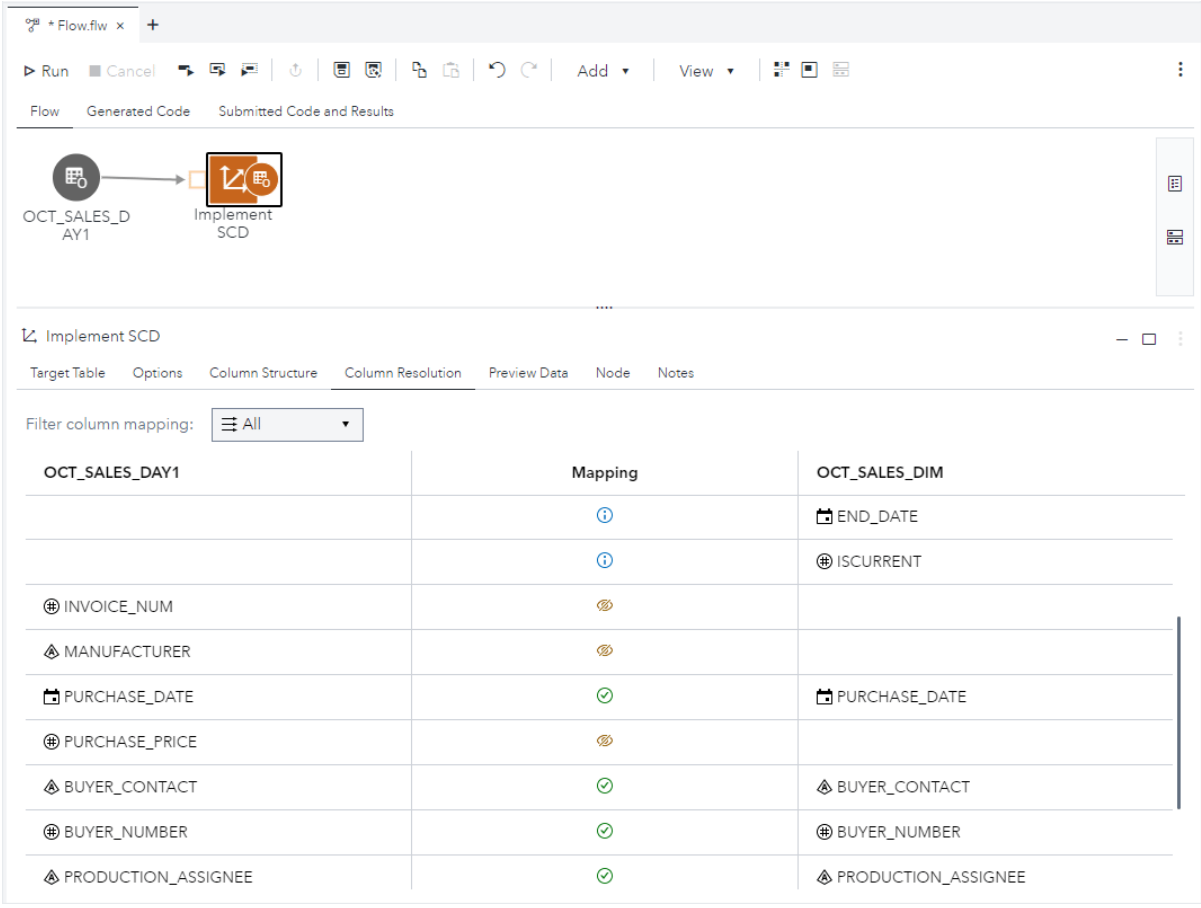
3 To view the structure of the target table, click the **Column Structure** tab.

To view the column mapping between the source and target tables, click the **Column Resolution** tab.

Note: On the Column Resolution tab, you can 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 source column is successfully mapped to a column in the target table.
- Ignored (👁) - the source column does not have a corresponding column in the target table. Data for this column is ignored in the target table.
- Information (ⓘ) - a source column is not mapped to the column in the target table. An unresolved column can occur when SAS Studio cannot automatically map columns based on name and data type.

In this example, there are two columns in the target table that do not have corresponding columns in the source table, and there are three columns in the source table that do not have corresponding columns in the target table.



Step 3: Specify One or More Key Columns

You must specify at least one business key column to run the Implement SCD step.

Note: To preview the expression that is generated by SAS Studio, click **Preview expression**. You can also copy the code to use in a SAS program. This option is available only when you have specified one or more key columns and all of the options for at least one SCD Type 1 or Type 2 change.

1. Click the **Options** tab and select **Identify business keys**.
2. To add a key column, click **Select columns**.
3. In the Select Columns window, select one or more key columns. You must select at least one key column that uniquely identifies each row in the source table. Do not select all of the columns in the list. Click **OK**.
4. SAS Studio attempts to match key columns in the source and target tables based on column name, regardless of case and data type. If a match is not found, click  to select the source business key column. Click **OK**.

(Optional) Step 4: Specify the SCD Type 1 Changes

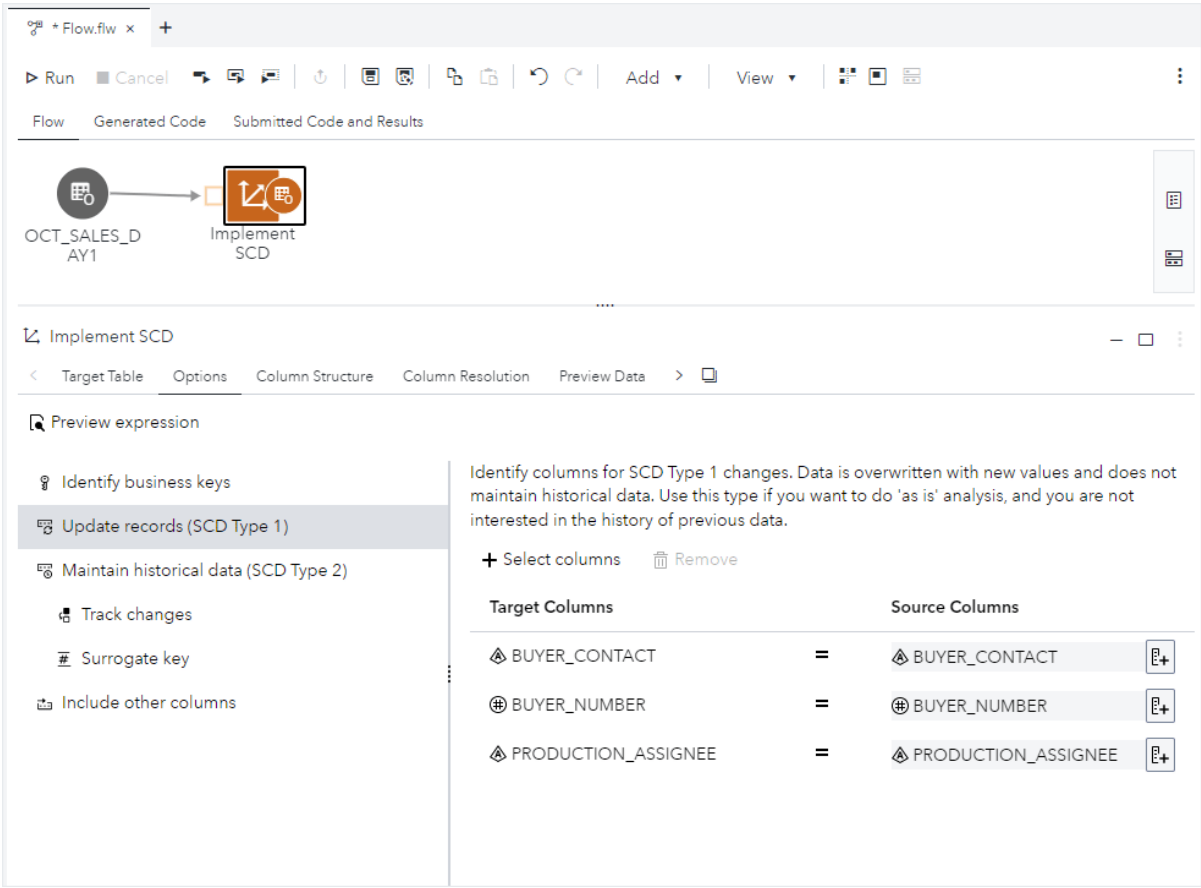
You can specify columns of data in the target table to overwrite with new data. SCD Type 1 changes do not preserve the historical data.

Note: You cannot update multiple Type 1 target columns with the same source column. Each source column can be used to update only one Type 1 target column.

1. Click the **Options** tab and select **Update records (SCD Type 1)**.
2. To select columns to update, click **Select columns**.
3. In the Select Columns window, select one or more target columns to update in the target table. Click **OK**.

   Note: Any columns that you have selected for other options are not available.

4. SAS Studio attempts to match columns in the source and target tables based on column name, regardless of case and data type. If a match is not found, click  to select the column that you want to use from the source table. Click **OK**.



(Optional) Step 5: Specify the SCD Type 2 Changes

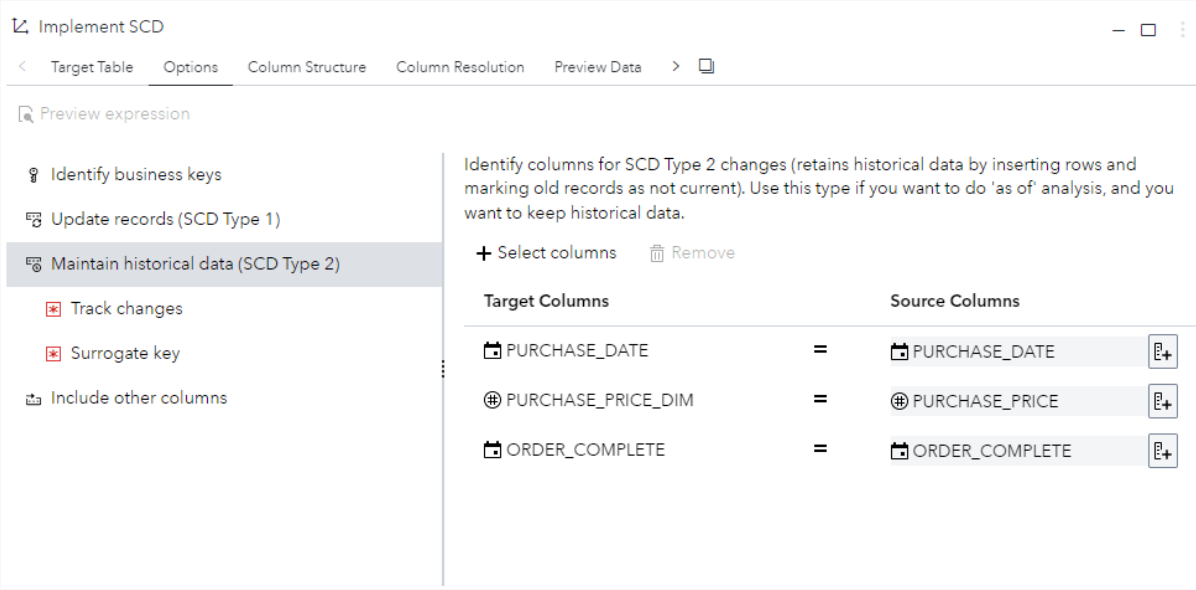
You can specify columns to insert in a new row in the target table for Type 2 changes. A new row is created in the target table and marked as the current row, and the old rows are preserved.

Note: You cannot match multiple Type 2 target columns to the same source column. Each source column can be matched with only one Type 2 target column.

1. Click the **Options** tab and select **Maintain historical data (SCD Type 2)**.
2. To select columns to add to the new row, click **Select columns**.
3. In the Select Columns window, select one or more target columns. Click **OK**.

   Note: Any columns that you have selected for other options are not available.

4. SAS Studio attempts to match columns in the source and target tables based on column name, regardless of case and data type. If a match is not found, click to select the column that you want to use from the target table. Click **OK**.



5 To specify how the progression of changes is tracked, click **Track changes**. You must specify both the effective date and flagging methods.

   a In the **Effective Date Method** area, specify the column that contains the start date timestamp that identifies the effective start date and time for the row.

   Note: The column that you specify for the start date must be either a date column or a numeric column that contains date values. If the column does not contain date values, the flow cannot run.

   Use the **Start on** field to specify the timestamp when the row was created. The default value is the current date.

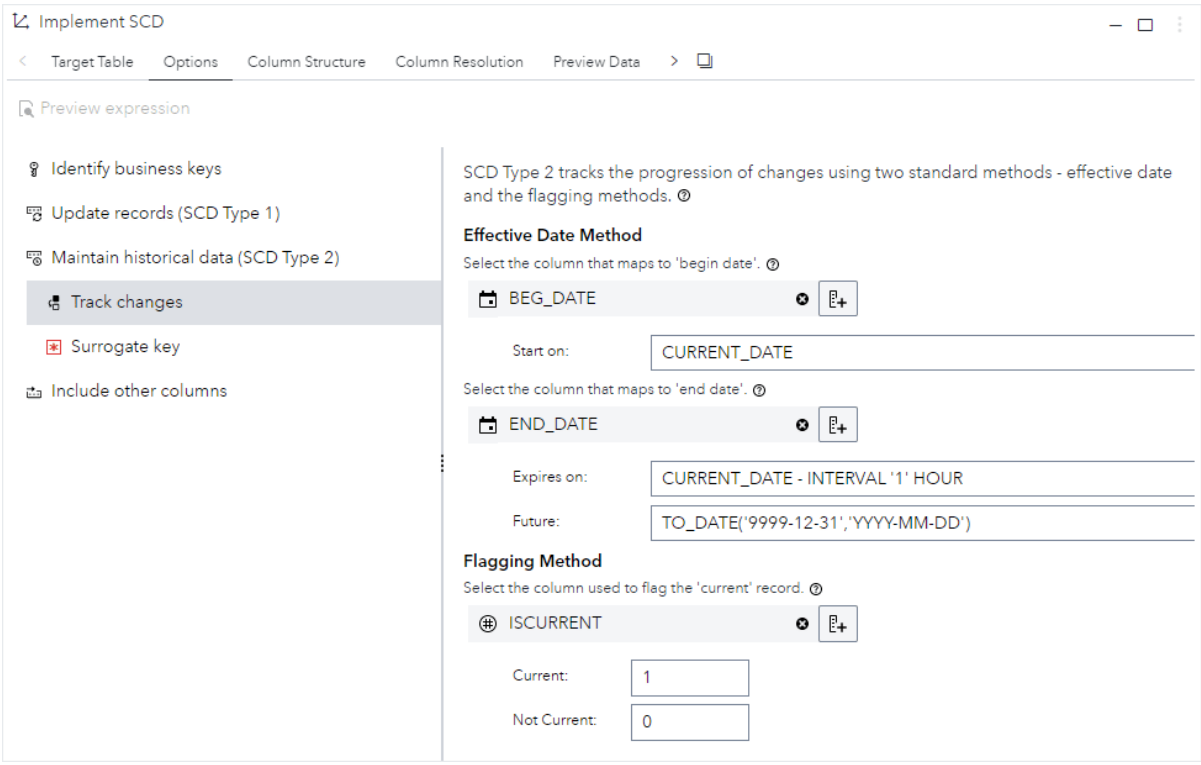
   b Specify the column that contains the end date that identifies the current row.

   Note: The column that you specify for the end date must be either a date column or a numeric column that contains date values. If the column does not contain date values, the flow cannot run.

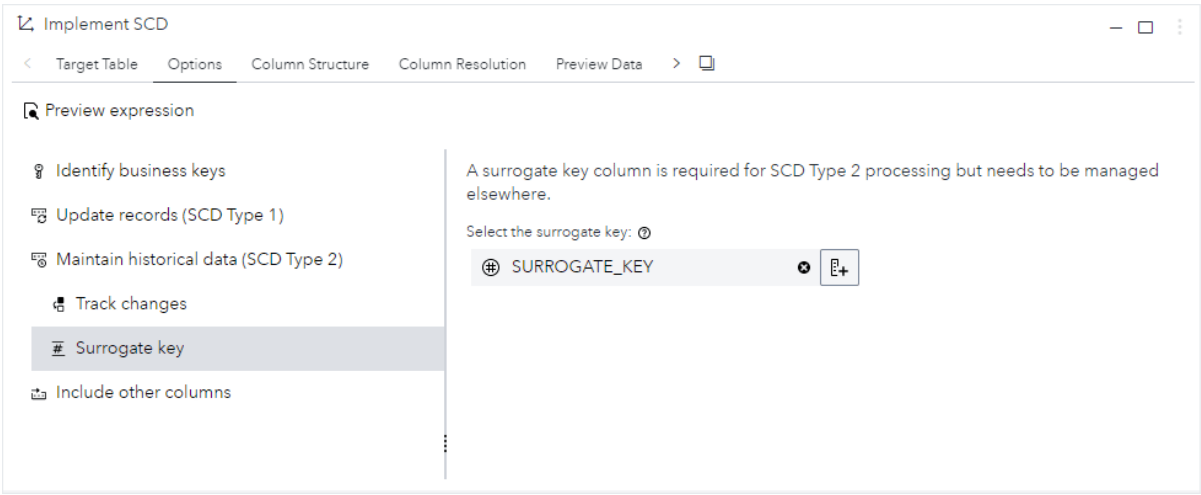
   Use the **Expires on** field to specify an expression that defines the end date value when closing out a row that was previously current. The default expression sets the effective end date timestamp to one hour earlier than the effective start date timestamp of the newly created current row.

   Use the **Future** field to specify the end date value for a row that is current. Typically, the future date value is set far into the future to ensure that the end date value does not expire while the row is still current. The default value is `9999-12-31`.

   c In the **Flagging Method** area, specify the column that contains the Boolean-type value that identifies the current row. Use the **Current** and **Not Current** fields to specify the values that are used to label the rows. By default, for character columns, the values of `Y` and `N` are used to identify current and historical rows. For numeric columns, the default values are `1` and `0`.



6 Click **Surrogate key** to specify the surrogate key in the target dimension table. To select the surrogate key column, click and select the target column. Click **OK**. The surrogate key column values are system-generated values that are used to uniquely identify a row in the table. The surrogate key and its values are required for Type 2 changes and must be generated outside of SAS Studio.



(Optional) Step 6: Specify Other Columns to Include in the Target Table

When new rows are created, you can include additional columns that have not already been identified as a surrogate key, business key, Type 1 column, Type 2

column, effective start and end date column, or current record indicator column. Columns that are not selected are added as missing values when new rows are created in the target table.

1 Click the **Options** tab and select **Include other columns**.

2 To select columns to update, click **Select columns**.

3 In the Select Columns window, select one or more columns to update in the target table. Click **OK**.

Note: The columns that you have already selected for other options are not available.

Step 7: Run the Flow and Update the Target Table

To run the Implement SCD step and update the target table, click ▷ Run.

Load Table: Loading Rows from a Source Table into a Target Table

About the Load Table Step

The Load Table step enables you to load a source table into a target table. The step supports tables from SAS, Oracle, Teradata, Synapse, Snowflake, and SingleStore. When you use the Load Table step, you can control how data is loaded into the target table. You can choose to insert new source rows into the target table, update existing rows in the target table, or both. You can also control how existing rows in the target table are removed before new rows are inserted.

For example, you might have a sales table that you need to update periodically with new pricing data. You can use the Load Table step to update the price column of the sales table by using the product ID as the key column.

When you load a table, you can choose the load technique that you want to use:

- Insert rows – inserts all of the rows in the source table into the output table.

Note: If you are inserting rows from the source table into the target table, you can specify a preprocessing action that determines how existing rows in the target table are removed before rows are inserted.

- Update rows – updates existing rows in the target table by using matching rows from the source table based on one or more key columns.
- Upsert rows – inserts new rows into the target table and updates existing rows in the target table based on one or more key columns.

Note: This step is available only if your site licenses SAS Studio Engineer.

There are five main steps to use the Load Table step:

1 Add the Load Table step to your flow and connect the source table.

   Note: The Load Table node does not have an output port. You must specify the target table in the Load Table node details.

2 Specify the library and name of the target table.

3 Select the load technique that you want to use: Insert rows, Update rows, or Upsert rows. If you are using the Insert rows technique, you can also specify a preprocessing action. If you are using the Update rows or Upsert rows techniques, you must specify at least one key column.

4 (Optional) Specify output table options.

5 Run the flow to load the data.

Node Connection Requirements

The Load Table step includes only one input port. The target table is specified in the node details rather than with an output port. In order to run the Load Table step, you must create a connection to the input port of the node as indicated here:

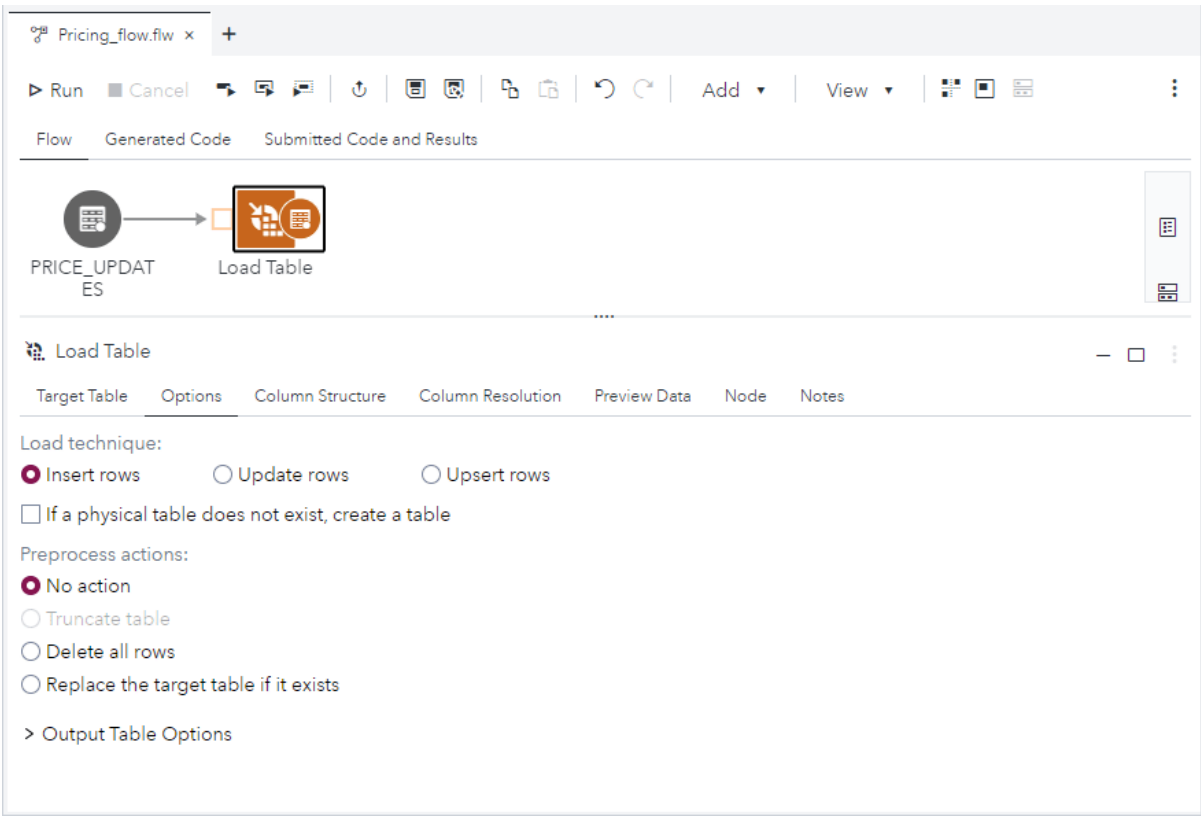
Input Port	Output Port
■ Table node or ■ Operational node that creates an output table, such as a Query node, a SAS Program node, or an Import node	Not applicable.

Load Table: Step-by-Step Instructions

Step 1: Add the Load Table Step and Connect a SourceTable

To add the Load Table step to your flow:

1. In the **Steps** section of the navigation pane, expand the **Integrate** folder, and double-click **Load Table**.
2. Connect the input port of the Load Table node to a data source by using a Table node or another node that creates an output table, such a Query node, a SAS Program node, or an Import node. For more information, see “Connecting Nodes” on page 12.



Step 2: Specify the Target Table

The Load Table step can run only if the metadata that defines the column structure of the target table exists.

To specify the target table:

1 Click the **Load Table** node, and then click the **Target Table** tab.

2 Use the **Library** and **Table name** boxes to specify the target table.

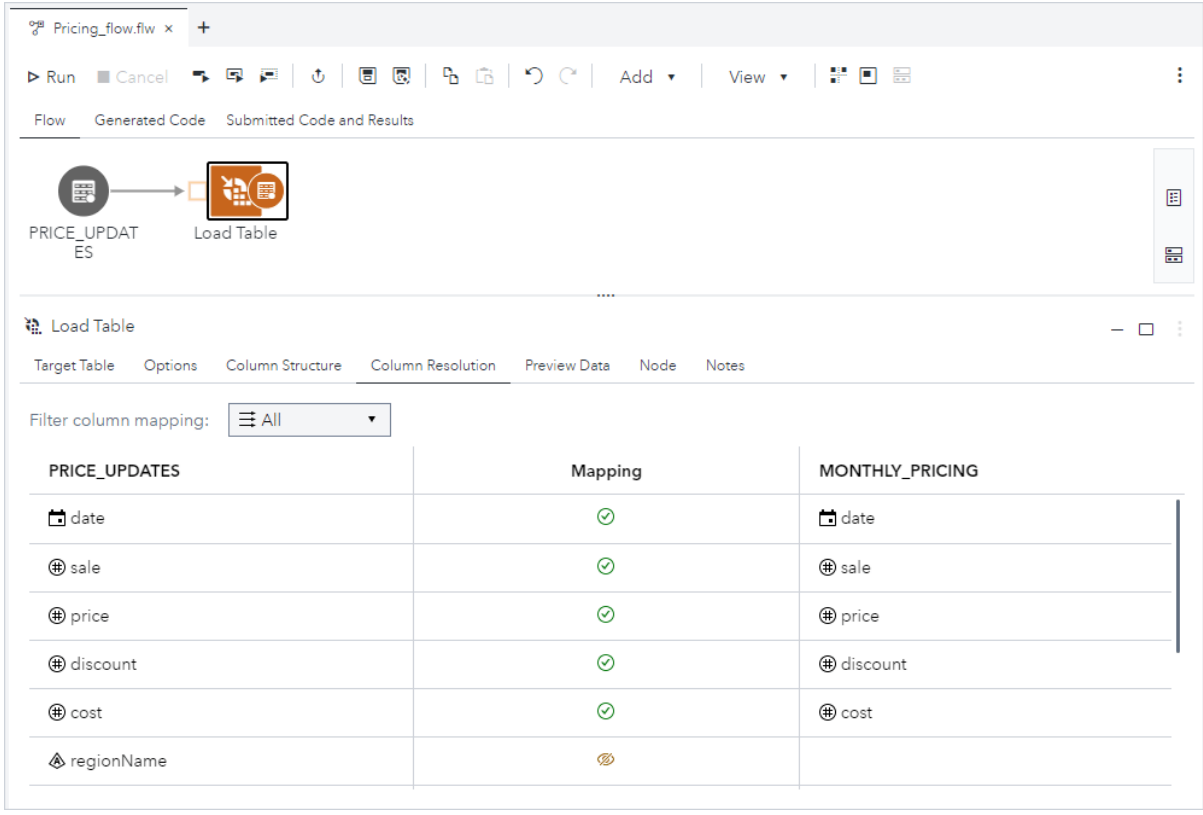
Note: If no metadata is found for the target table that you selected, you must either select a new target table or use the Column Structure tab to specify the column structure. You can copy the column structure from the source table or load the structure from a structure definition file. For more information, see "Specifying the Column Structure for the Target Table" on page 114.

3 To view the structure of the target table, click the **Column Structure** tab.

To view the column mapping between the source and target tables, click the **Column Resolution** tab.

Note: On the Column Resolution tab, you can 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 source column is successfully mapped to a column in the target table.
- Ignored (👁) - the source column does not have a corresponding column in the target table. Data for this column is ignored in the target table.
- Information (ⓘ) - a source column is not mapped to the column in the target table. An unresolved column can occur when SAS Studio cannot automatically map columns based on name and data type.



Step 3: Select the Load Technique

- Load Data Using the Insert Rows Technique

  1 Click the **Options** tab and select **Insert rows**.

  2 If the physical target table does not exist, you can automatically create the table by selecting **If a physical table does not exist, create a table**. If no metadata is found for the target table that you specified, you must either select a new target table or use the Column Structure tab to specify the column structure. You can copy the column structure from the source table or load the structure from a structure definition file. For more information, see "Specifying the Column Structure for the Target Table" on page 114.

  If the physical target table does not exist and you do not select this option, the step cannot run successfully.

  3 Select the preprocess action that you want to use:

  - **No action** - no preprocess action is applied. This option is selected by default.
  - **Truncate table** - removes all rows from the table before the source data is loaded, but maintains the table structure, metadata, constraints, and indexes. Foreign key constraints are not maintained. This option uses the database TRUNCATE statement.

    Note: The **Truncate table** option is available only for SAS data sets and databases that support the TRUNCATE statement, such as Oracle, Snowflake, Singlestore, and Azure Synapse.

  - **Delete all rows** - deletes all rows from the target table before the source data is loaded. If the target table is an Oracle, Teradata, Snowflake, Singlestore or Azure Synapse table, this option uses the DELETE statement that is associated with the data source. For all other data types, this option uses the PROC SQL DELETE statement.

    Note: When the **Delete all rows** action is used repeatedly to clear a SAS table, the size of that table should be monitored over time. **Delete all rows** performs only logical deletes for SAS tables. Therefore, the table's physical size continues to increase, which can negatively affect performance.

  - **Replace the target table if it exists** - replaces the entire table with an empty table before the source data is loaded. This option does not preserve the existing table structure, metadata, constraints, and indexes.

- Load Data Using the Update Rows Technique

  You must specify at least one key column in order to use the Update rows technique.

  1 Click the **Options** tab and select **Update Rows**.

2 Click + and select one or more key columns. You must select at least one key column that uniquely identifies each row in the source table. Do not select all of the columns in the list. Click **OK**.

Note: For some databases, strict matching rules apply to key columns. Depending on the data type of the target table, you might not be able to select a column as a key column if the case of the source and target columns does not match. The Select Key Columns window indicates whether the case of the source and target columns matches. Use the Column Resolution tab to view the source and target column names.

Note: If you are using the Update rows technique to load data into a Snowflake or Singlestore table, a primary key must be defined for the target table. For more information, see "Understanding Snowflake Update and Delete Rules" in *SAS/ACCESS for Relational Databases: Reference*.

- Load Data Using the Upsert Rows Technique

  You must specify at least one key column in order to use the Upsert rows technique.

  1 Click the **Options** tab and select **Upsert Rows**.

  2 If the physical target table does not exist, you can automatically create the table by selecting **If a physical table does not exist, create a table**. If no metadata is found for the target table that you specified, you must either select a new target table or use the Column Structure tab to specify the column structure. You can copy the column structure from the source table or load the structure from a structure definition file. For more information, see "Specifying the Column Structure for the Target Table" on page 114.

  If the physical target table does not exist and you do not select this option, the step cannot run successfully.

  3 Click + and select one or more key columns. You must select at least one key column that uniquely identifies each row in the source table. Do not select all of the columns in the list. Click **OK**.

  For some databases, strict matching rules apply to key columns. Depending on the data type of the target table, you might not be able to select a column as a key column if the case of the source and target columns does not match. The Select Key Columns window indicates whether the case of the source and target columns matches. Use the Column Resolution tab to view the source and target column names.

  Note: If you are using the Upsert rows technique to load data into a Snowflake or Singlestore table, a primary key must be defined for the target table. For more information, see "Understanding Snowflake Update and Delete Rules" in *SAS/ACCESS for Relational Databases: Reference*.

(Optional) Step 4: Specify Output Table Options

You can specify options to optimize how the data is loaded into the target table.

1 Click the **Options** tab and expand **Output Table Options**.

2 If you are loading data into an Oracle, Snowflake, Singlestore, Azure Synapse, or Teradata target table, you can select one of the following options:

- **Use default table options** - uses the default SAS/ACCESS table options that are associated with the data type of the target table.
- **Bulk load** - loads the rows of data to the target table using the native bulk load facility that is associated with the data type of the target table.

  Note: If you are loading data into an Azure Synapse table, authentication to Azure Data Lake Storage is required. Single sign-on (SSO) is preferred. For more information, see "Authentication for Bulk Loading to Azure" in *SAS/ACCESS for Relational Databases: Reference*. If you need to specify any of the required authentication options as data set options instead of LIBNAME options, click **Advanced table options** and enter each option on a separate line.

- **Commit all rows in a single transaction** - changes to the target table are committed after all the rows have been written. This option uses the DBCOMMIT = 0 data set option for Oracle, Teradata, Snowflake, Singlestore, and Azure Synapse target tables.
- **Commit every *n* rows** - changes to the target table are committed after the specified number of rows is written. The default value is 100,000 rows. This option uses the DBCOMMIT = n data set option for Oracle, Teradata, Snowflake, Singlestore, and Azure Synapse target tables.

3 If necessary, you can specify advanced output table options by clicking **Advanced table options**. Enter each option on a separate line. Advanced table options can include any DATA step options that are associated with inserting and updating data in a table, such as options to specify the number of rows to process prior to committing the data or to specify whether to use a DBMS-specific bulk-load facility. You must include any quotation marks that are required for string values.

Step 5: Run the Flow and Load the Data

To load the source data into the target table, click ▷ Run.

Specifying the Column Structure for the Target Table

If no metadata is found for the target table that you have specified, you can use the Column Structure tab in the node details to specify the column structure. You can copy the column structure from the source table or load the structure from a structure definition file. You can also use the Column Structure tab to create a structure definition file that you can use to define the metadata for other target tables.

To copy the structure from the source table:

- Click the **Column Structure** tab and select **Structure** ⇨ **Copy structure from source**.

To load the column structure from an external file:

1. Click the **Column Structure** tab.
2. Select **Structure** ⇨ **Load structure definition**, and then select the metadata definition file that you want to use.

To create a structure definition file that is based on the current structure of the target table:

1. Click the **Column Structure** tab and select **Structure** ⇨ **Generate structure definition**.
2. In the Generate a Column Structure File window, specify the location and name of the file. You can save the file as a comma-delimited (*.csv) or text (*.txt) file. The column structure metadata file is always saved with UTF-8 encoding.

Merge Table: Updating and Inserting Rows in a Target Table

About the Merge Table Step

You can use the Merge Table step to make changes to columns in a target table based on values from a source table. Rows in the source and target columns are matched using one or more key columns. The Merge Table step enables you to combine update and insert operations in one step.

The Merge Table step works with DBMS tables that support ANSI SQL standards, including Oracle, Teradata, Snowflake, and SQL Server. Both the source and target tables must reside on the same database server.

Note: This step is available only if your site licenses SAS Studio Engineer.

There are six main steps to use the Merge Table step:

1. Add the Merge Table step to your flow and connect the source table.
2. Specify the library and name of the target table.
3. Specify one or more key columns that are used to match the rows between the source and target tables.
4. (Optional) Specify the columns in the target table that should be updated with values from the source table.
5. (Optional) Specify the columns in the target table into which new values should be inserted when no matching row is found.
6. Run the flow to merge the tables.

Note: To run the flow, you must specify at least one column to update or insert new values.

Node Connection Requirements

The Merge Table step includes only one input port. The target table is specified in the node details rather than in an output port. In order to run the Merge Table step, you must create a connection to the input port of the node as indicated here:

Input Port	Output Port
Table node. The table must reside in a database that supports ANSI SQL, including Oracle, Teradata, Snowflake, and SQL Server.	Not applicable.

Merge Table: Step-by-Step Instructions

Step 1: Add the Merge Table Step and Connect a Source Table

1. In the **Steps** section of the navigation pane, expand the **Integrate** folder, and double-click **Merge Table**.
2. Connect the input port of the Merge Table node to a data source by using a Table node. The table must reside in a database that supports ANSI SQL, such

as Oracle, Teradata, Snowflake, and SQL Server. For more information, see “Connecting Nodes” on page 12.

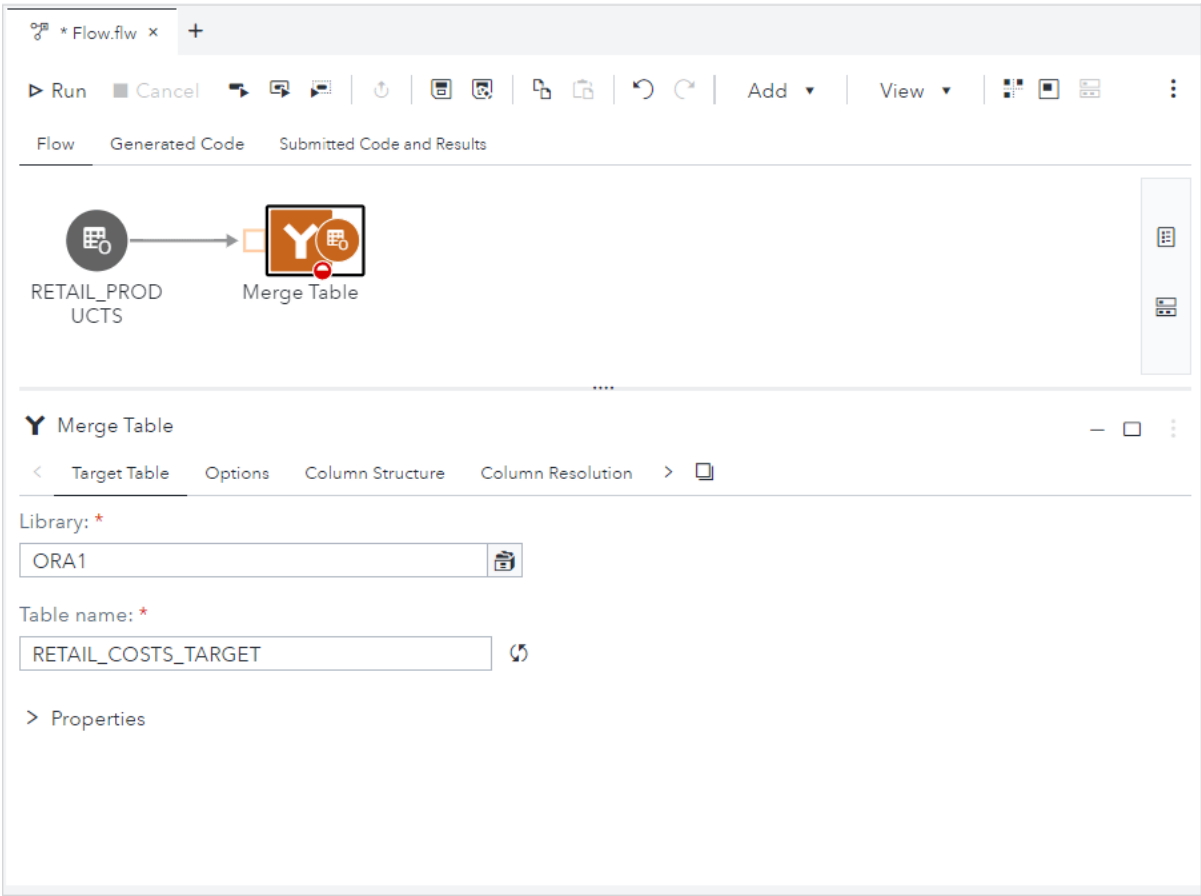
Step 2: Specify the Target Table

The source and target tables must reside on the same database server.

1 Click the **Merge Table** node, and then click the **Target Table** tab.

2 Use the **Library** and **Table name** boxes to specify the target table.

Note: The Merge Table node does not have an output port. You must specify the target table in the Merge Table node details. However, you can connect the Merge Table node to another operational node that requires a table as input.



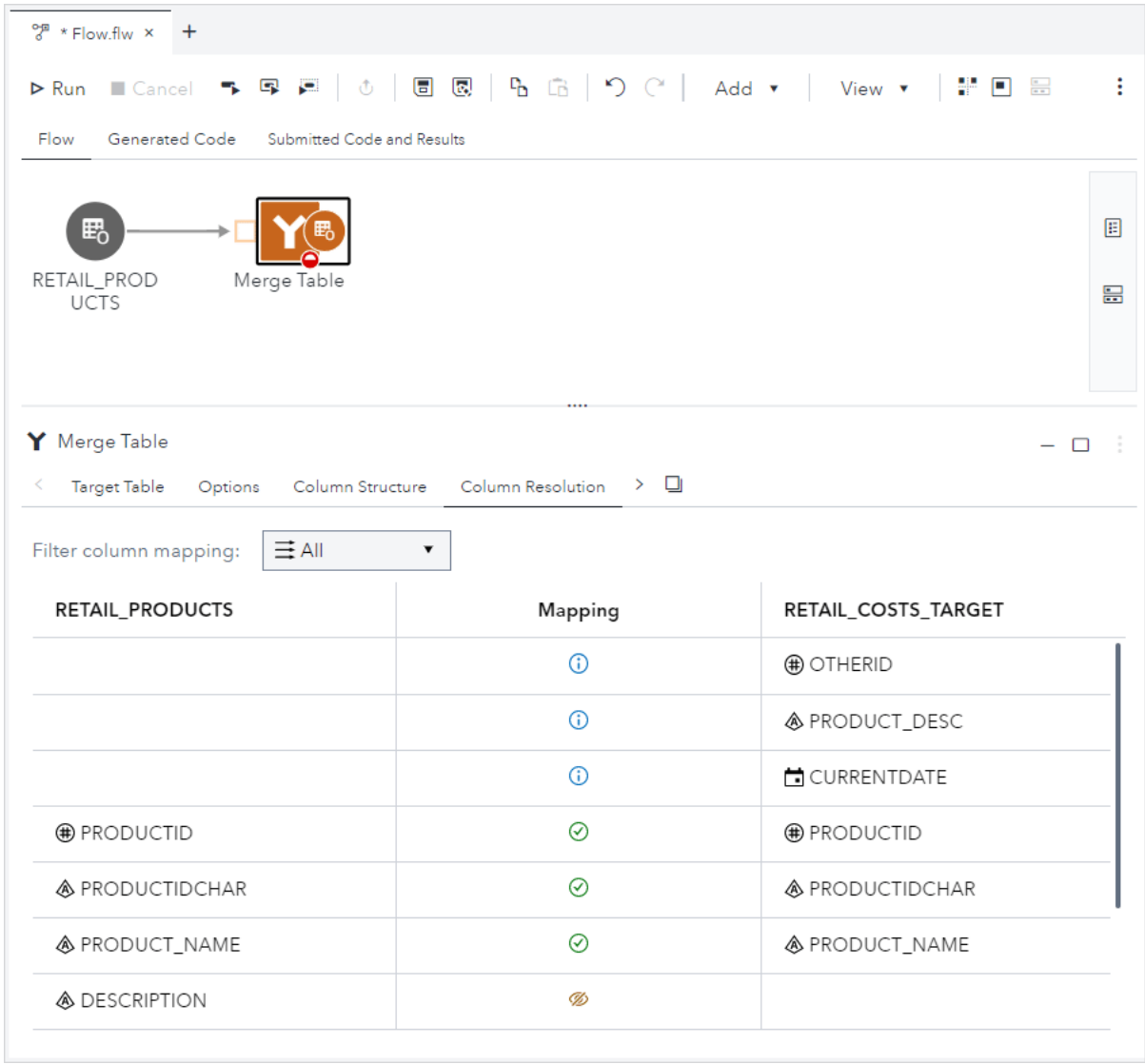
3 To view the structure of the target table, click the **Column Structure** tab.

To view the column mapping between the source and target tables, click the **Column Resolution** tab.

Note: On the Column Resolution tab, you can 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 source column is successfully mapped to a column in the target table.
- Ignored (⊘) - the source column does not have a corresponding column in the target table. Data for this column is ignored in the target table.
- Information (ⓘ) - a source column is not mapped to the column in the target table. An unresolved column can occur when SAS Studio cannot automatically map columns based on name and data type.

In this example, there are three columns in the target table that do not have corresponding columns in the source table, and one column in the source table that does not have a corresponding column in the target table.



Step 3: Specify One or More Key Columns

You must specify at least one key column to run the Merge Table step.

Note: To use the SQL editor to create SQL code for the Merge Table expression, click **SQL editor**. If you have used the Options tab to specify one or more key columns and at least one other column to update or insert, SAS Studio automatically generates the SQL code for those options. The SQL code is passed to the target database for processing by using explicit pass-through, so you should use the SQL syntax of your data source. The expression must include the code to specify the key columns, as well as any target columns to update and target column values to insert. When you select this option, you cannot revert to using the graphical expression builder.

To preview the merge expression that is generated by SAS Studio, click **Preview expression**. You can also copy the code to use in a SAS program. This option is available only when you have specified one or more key columns and at least one column to update or insert.

1. Click the **Options** tab and select **Key columns**.
2. To add a key column, click **Add columns**.
3. In the Select a Column window, select one or more key columns. You must select at least one key column that uniquely identifies each row in the source table. Do not select all of the columns in the list. Click **OK**.
4. SAS Studio attempts to match key columns in the source and target tables based on column name, regardless of case and data type. If a match is not found, click to select the column that you want to use from the target table. Click **OK**.

(Optional) Step 4: Specify the Target Columns to Update

You can specify columns in the target table to update with values from the corresponding source columns for each row that is matched by a key column. This step is optional if you specify a column to insert values.

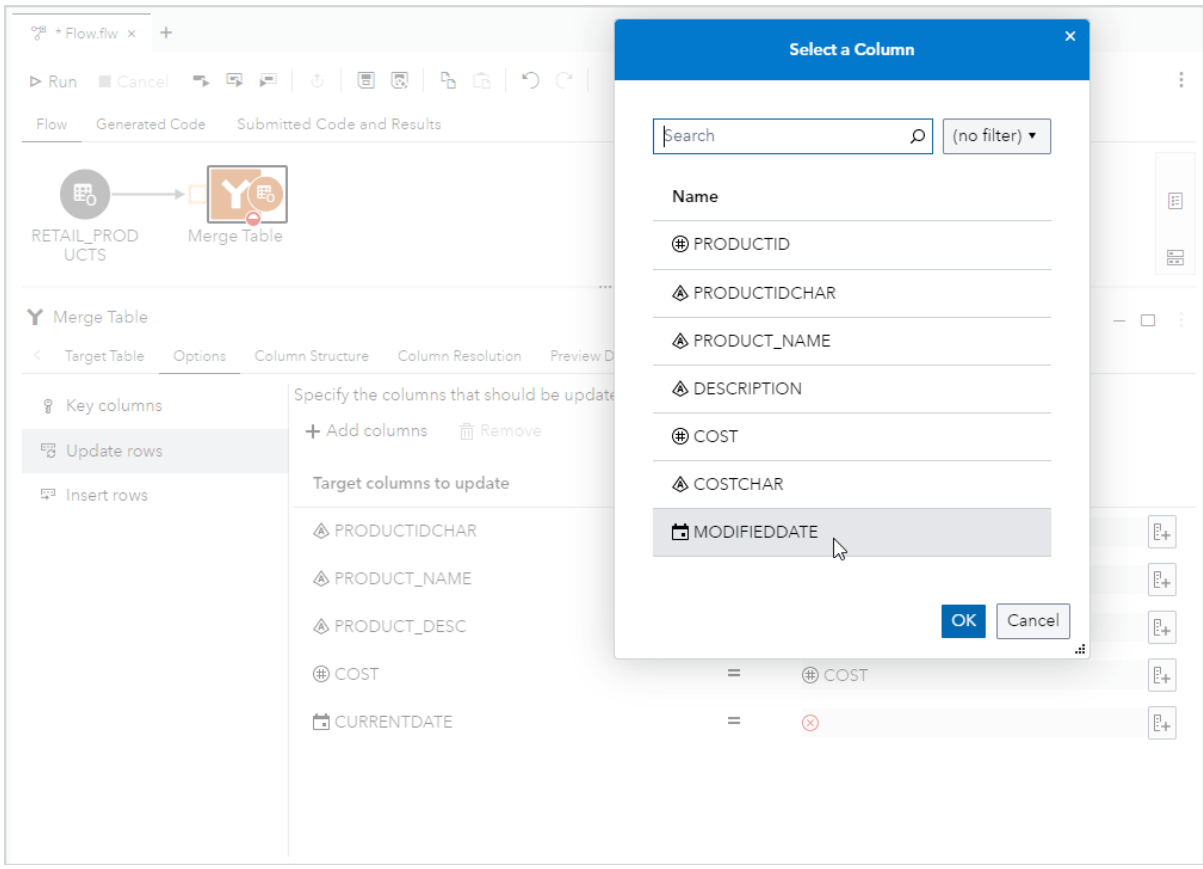
Note: Columns that are used as key columns cannot be updated.

1. Click the **Options** tab and select **Update rows**.
2. To select a column to update, click **Add columns**.

   Note: You can add all of the columns in the target table by clicking **Add all columns**.

3. In the Select a Column window, select one or more columns.
4. SAS Studio attempts to match a column in the source table to the column that you selected from the target table based on column name, regardless of case and data type. If a match is not found, click to select the column that you want to use from the source table. Click **OK**.

In this example, the MODIFIEDDATE column is selected to map to the CURRENTDATE column.



(Optional) Step 5: Specify Target Column Values to Insert

Specify the columns where new values should be inserted when no matching row is found. This step is optional if you specify a column to update.

1. Click the **Options** tab and select **Insert rows**.
2. To select a column to insert, click **Add columns**.
3. In the Select a Column window, select one or more columns.
4. SAS Studio attempts to match a column in the source table to the column that you selected from the target table based on column name, regardless of case and data type. If a match is not found, click to select the column that you want to use from the source table. Click **OK**.

Step 6: Run the Flow and Merge the Tables

To merge the source and target tables, click ▷ Run.

7

Enriching Your Data

Geocode Data Step

About the Geocode Data Step

Geocoding is the process of adding geographic coordinates (latitude and longitude values) to an address. This process provides a way to convert address data into

map locations. The geographic coordinates typically represent the center of a ZIP code, a city, an address, or any geographic region. After geocoding, the coordinates can be used to display a point on a map or to calculate distances. Geocoding also enables you to add attribute values such as census blocks to the geocoded address.

The Geocode Data step uses the SAS GEOCODE procedure. The GEOCODE procedure uses two types of input data:

- an input table with columns that relate to specific geographic locations. For example, a table might contain mailing address variables such as ZIP codes and street addresses, or custom geographic variables such as sales regions.
- lookup tables that contain reference variables and geographic coordinates. For example, a lookup data table for the ZIP method contains ZIP codes and the geographic coordinates that are associated with the ZIP codes. Some geocoding methods require multiple lookup data tables. This data is essential to transform address data into location information that can be viewed on a map.

By default, the GEOCODE procedure produces an output data set that contains all of the variables from the input table. The data set also contains the X, Y or LONG, LAT, and _MATCHED_ variables. The X or LONG and Y or LAT coordinates must be in the same coordinate system as the lookup data set.

The Geocode step supports seven geocode methods:

- U.S. City
- World City
- Street
- Zip
- Plus4
- Custom
- Internet Protocol (IP) addresses or other ranges

Note: The process of adding coordinates for IP addresses is usually called *geolocating*. IP data is a form of range data and was not designed to be geographic.

For more information, see "GEOCODE Procedure" in *SAS/GRAPH and Base SAS: Mapping Reference*.

Node Connection Requirements

The Geocode Data step includes an input port and an output port. In order to run the Geocode Data step, you must create the input port of the node as indicated here:

Input Port	Output Port
■ Table node or	No connection is required. **Note:** By default, the output data is written to a temporary table in the Work library. You can specify

Input Port	Output Port
■ Operational node that creates an output table, such as a Query node, a SAS Program node, or an Import node	the library and name of the output table by connecting the output port to a Table node. For more information, see “Adding a Table from a SAS Library to a Flow” on page 34.

TIP Depending on the geocode method, you might need more than one input port on the Geocode Data node. To view the labels for the ports on the Geocode Data node, right-click the Geocode Data node and select **Expand ports**.

Using the U.S. City Geocode Method

About the U.S. City Geocode Method

For U.S. city geocoding, each observation in your input data must contain a city and a state. The recommended state value is the two-character state abbreviation, but the complete state name can also be used.

U.S. City Geocoding: Step-by-Step Instructions

Step 1: Add the Geocode Data Step and Connect a Source Table

1 In the **Steps** section of the navigation pane, expand the **Enrichment** folder and double-click **Geocode Data** to add the step to your flow.

2 Connect the input port of the Geocode Data 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 12.

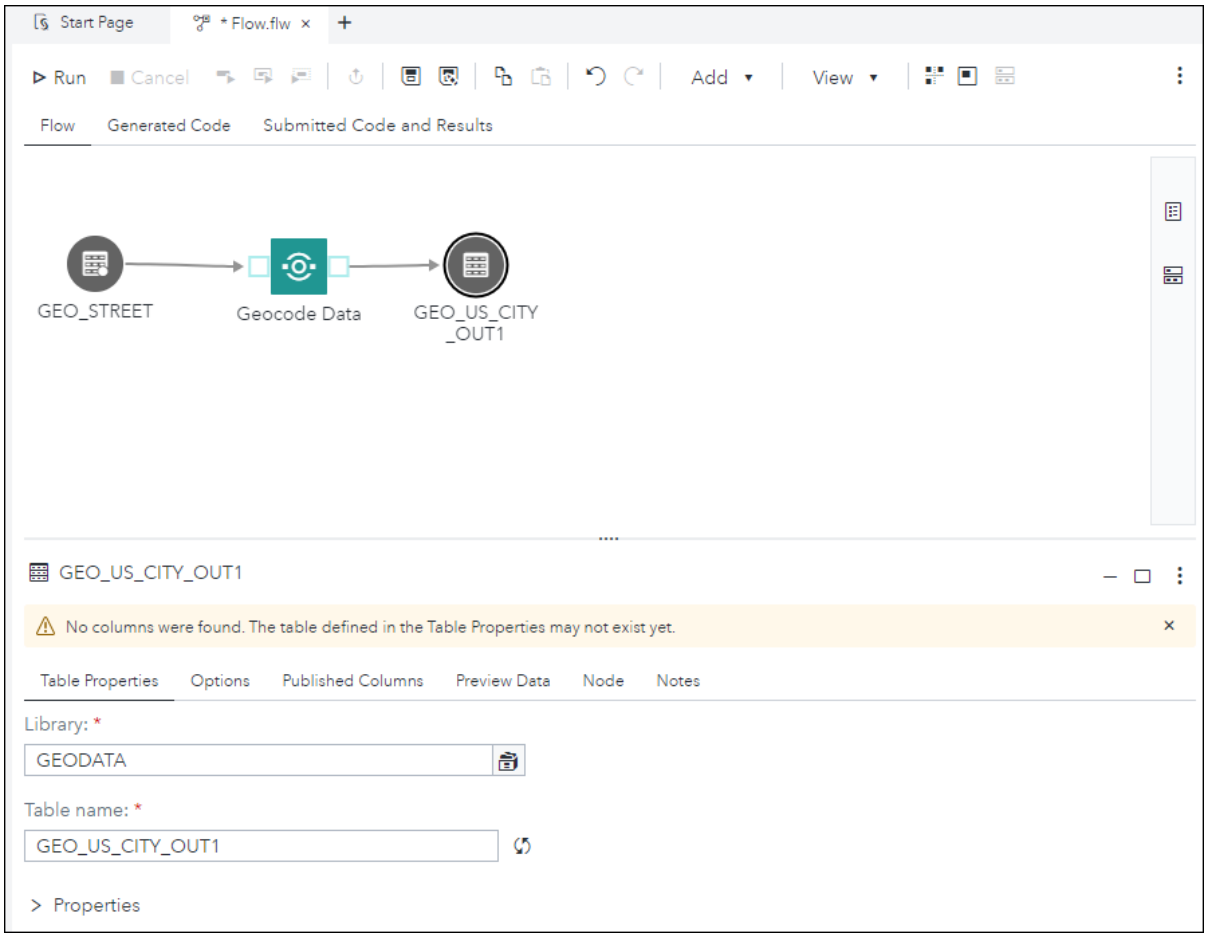
Note: The input data must contain columns for the city and state.

3 To save the results from the Geocode Data step to a permanent output table, select **Add** ⇨ **Table**. A table node is added to the flow. Specify the library and

name for the output table. (In this example, the output table is named Geo_US_City_Out1.)

Next, connect the output port of the Geocode Data node to the table node.

Note: The output table does not have any columns until you run the flow.



Step 2: Select U.S. City as the Geocoding Method

1. Select **World city** as the geocoding method.
2. Select the **U.S. city** check box.
3. Map these fields for geocoding:
   - **City**
   - **State/Province**
   - **Postal code**

(Optional) Step 3: Set Additional Options

1. To convert the data to use standard two-letter abbreviation, select **Perform state two-letter abbreviation before processing**.
2. Specify any optional arguments from the GEOCODE procedure.

   For example, add `LOOKUPCITY=SASHELP.ZIPCODE` to specify the lookup table for associating coordinates with addresses.

(Optional) Step 4: Set the Debugging Options

To add extra SAS debugging for macros, select **Debug SAS macros**.

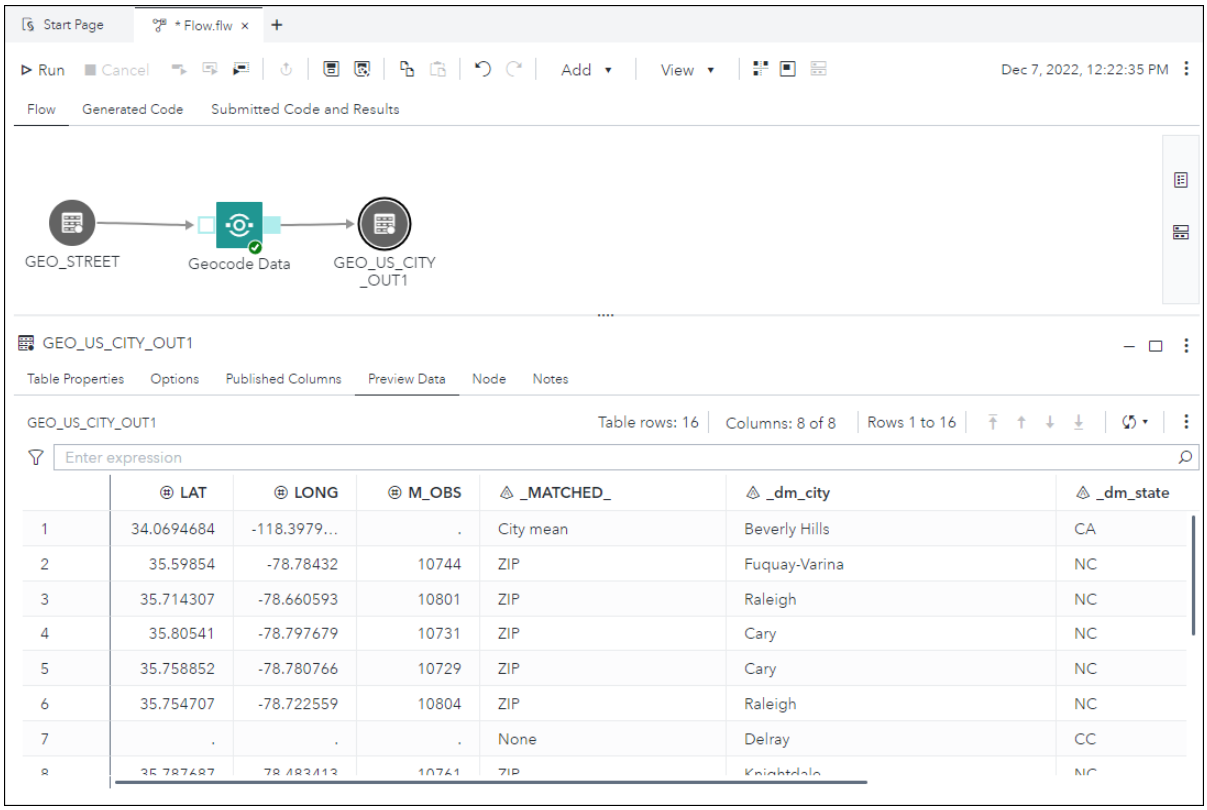
Step 5: Run the Flow and View the Output Table

To run the flow, click ▷ Run.

To view the output data, click the table node, and then click the **Preview Data** tab.

Items to note in the output table:

- Columns from the SAS data source are prefixed with _dm.
- The GEOCODE procedure generates a _MATCHED_ column that indicates how the coordinates were found.



	LAT	LONG	M_OBS	_MATCHED_	_dm_city	_dm_state
1	34.0694684	-118.3979...	.	City mean	Beverly Hills	CA
2	35.59854	-78.78432	10744	ZIP	Fuquay-Varina	NC
3	35.714307	-78.660593	10801	ZIP	Raleigh	NC
4	35.80541	-78.797679	10731	ZIP	Cary	NC
5	35.758852	-78.780766	10729	ZIP	Cary	NC
6	35.754707	-78.722559	10804	ZIP	Raleigh	NC
7	.	.	.	None	Delray	CC
8	35.787687	78.483413	10761	ZIP	Knightdale	NC

Using the World City Geocode Method

About the World City Geocode Method

For worldwide geocoding, each observation in your input address data must include the city name and the country. The country value can be the complete country name or its two- or three-character country abbreviation. An example is GB or GBR for the United Kingdom. An optional state, province, or region name can also be specified.

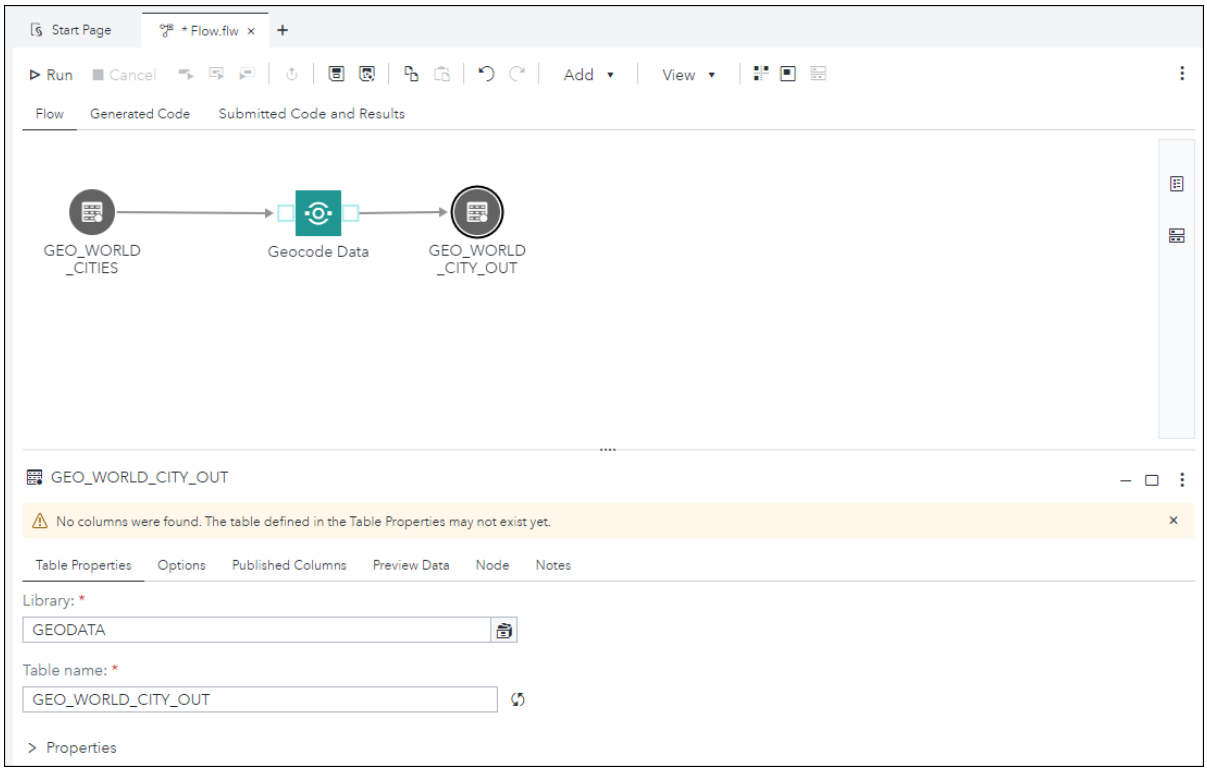
World City Geocoding: Step-by-Step Instructions

Step 1: Add the Geocode Data Step and Connect a Source Table

1. In the **Steps** section of the navigation pane, expand the **Enrichment** folder and double-click **Geocode Data** to add the step to your flow.
2. Connect the input port of the Geocode Data 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 12.
3. To save the results from the Geocode Data step to a permanent output table, select **Add** ⇨ **Table**. A table node is added to the flow. Specify the library and name for the output table. (In this example, the output table is named Geo_World_City_Out.)

   Next, connect the output port of the Geocode Data node to the table node.

   Note: The output table does not have any columns until you run the flow.



Step 2: Select World City as the Geocoding Method

1 In the Geocode Data step, select **World city** as the geocoding method.

2 Map these fields for geocoding:

- **City**
- **Country**

Step 3: Specify Lookup Table

1 To view the labels for the ports on the Geocode Data node, right-click the Geocode Data node in the flow and select **Expand ports**. In the flow, the input port for the input data source (in this example, GEO_WORLD_CITIES) is labeled Input Table 1.

2 Add the input port and data for the city lookup table.

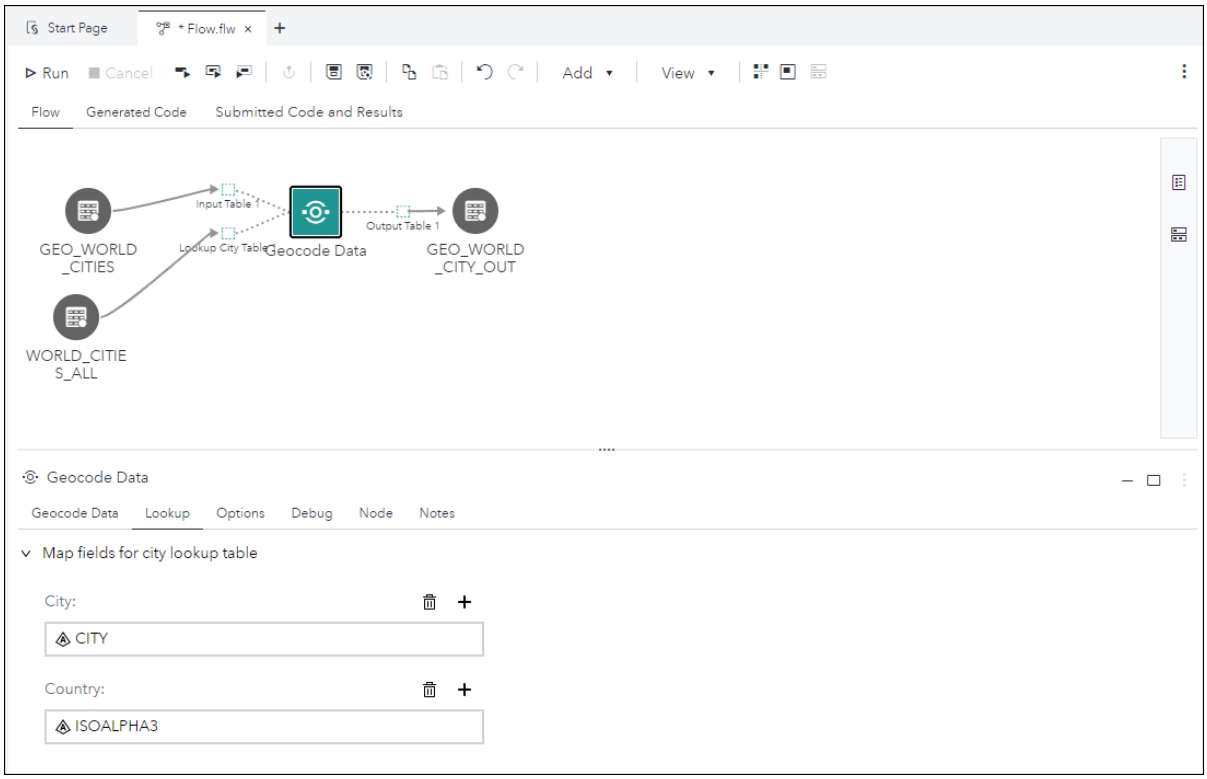
a Right-click the Geocode Data node and select **Add input port** ⇨ **Lookup City Table**.

b Add the lookup table to the flow. (In this example, the lookup table is World_Cities_All.) Connect the lookup table node to the Lookup City Table port.

c On the **Lookup** tab in the Geocode Data step, map these fields for the city lookup table:

- **City**

- **Country**



(Optional) Step 4: Set Additional Options

1. To convert the data to use the three-letter ISO standard for the country, select **Perform country-three letter ISO standardization before processing**.
2. Specify any optional arguments from the GEOCODE procedure.

   Here is an example: `attributevar=(isoname mapidname1) addressstatevar=province`

(Optional) Step 5: Set the Debugging Options

To add extra SAS debugging for macros, select **Debug SAS macros**.

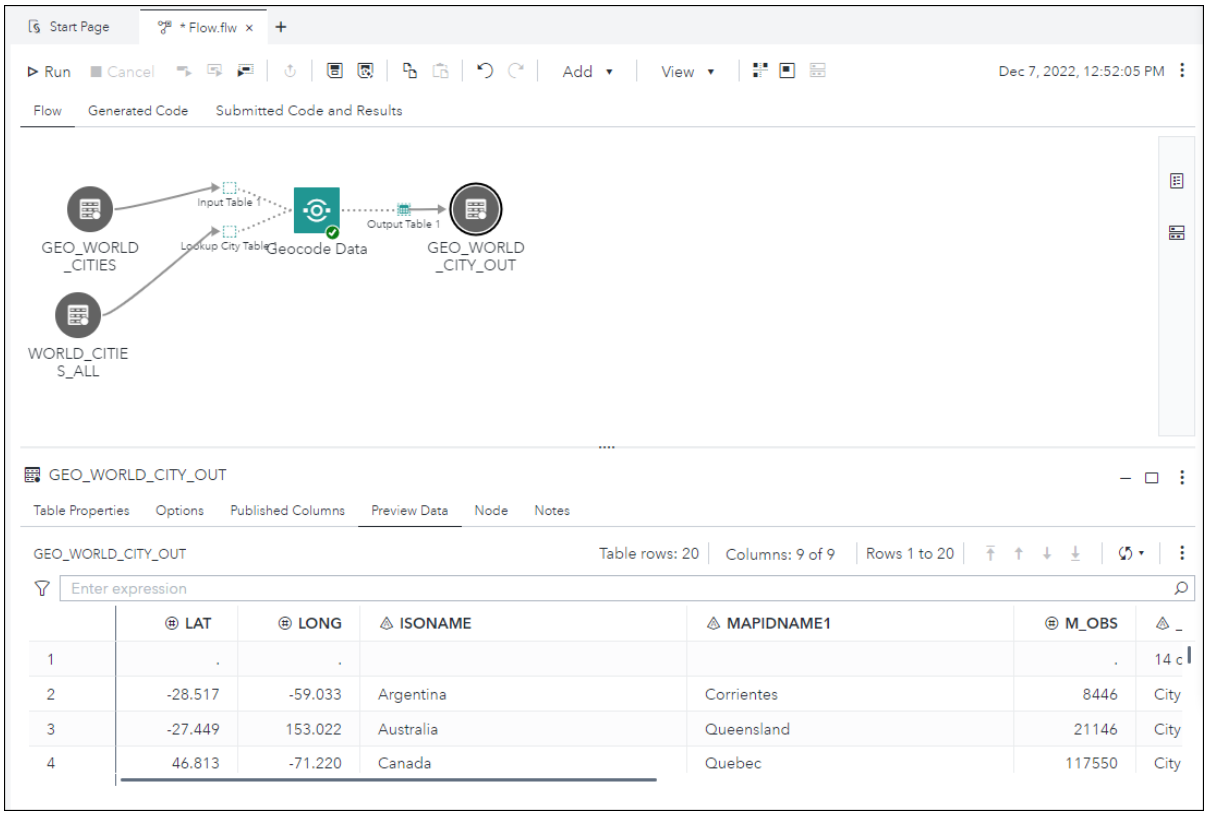
Step 6: Run the Flow and View the Output Table

To run the flow, click ▷ Run.

To view the output data, click the table node, and then click the **Preview Data** tab.

Items to note in the output table:

- Columns from the original data source are prefixed with _dm.
- The GEOCODE procedure generates a _MATCHED_ column that indicates how the coordinates were found.



Using the Street Geocode Method

About the Street Geocode Method

The street geocode method computes geographical coordinates for specified U.S. or Canadian street addresses. This method converts a full street address that includes a house or building number, street name, city, state (or province), and ZIP (or postal) code to a map location.

Street Geocoding: Step-by-Step Instructions

Step 1: Add the Geocode Data Step and Connect a Source Table

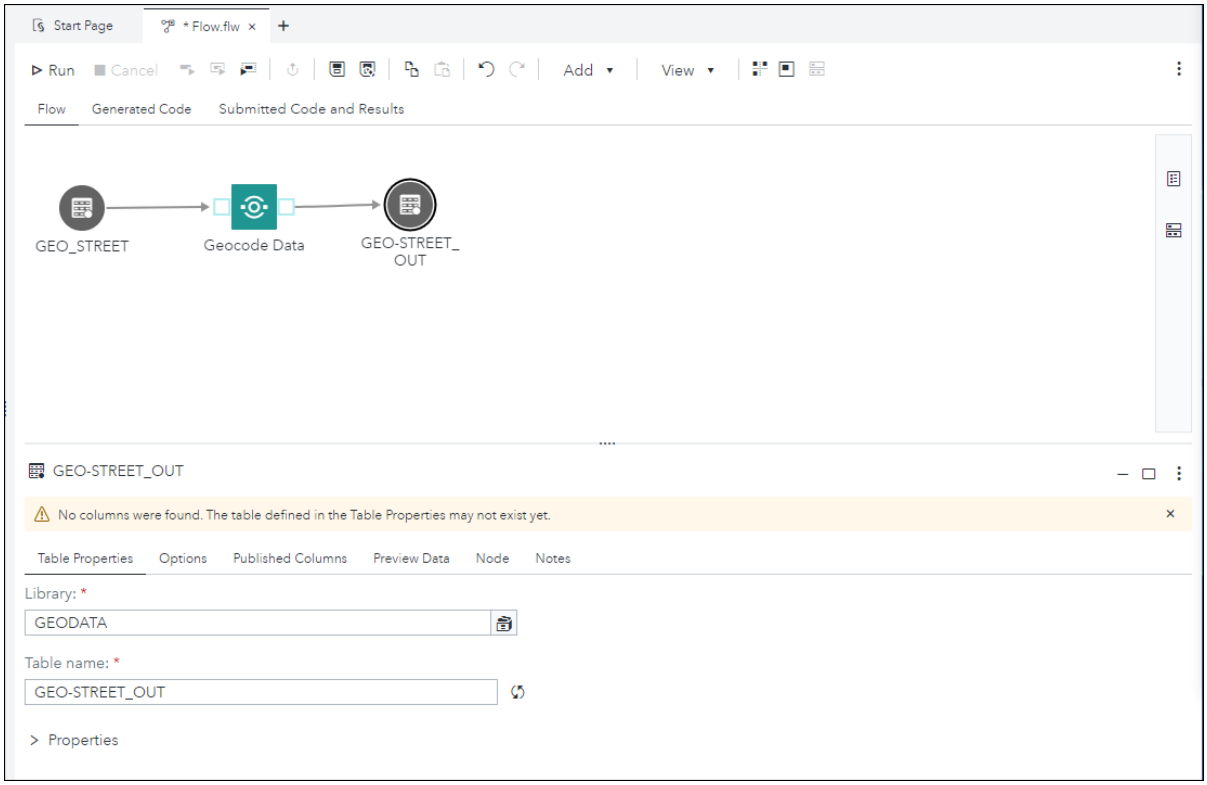
1 In the **Steps** section of the navigation pane, expand the **Enrichment** folder and double-click **Geocode Data** to add the step to your flow.

2 Connect the input port of the Geocode Data 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 12.

3 To save the results from the Geocode Data step to a permanent output table, select **Add** ⇨ **Table**. A table node is added to the flow. Specify the library and name for the output table. (In this example, the output table is named Geo_Street_Out.)

Next, connect the output port of the Geocode Data node to the table node.

Note: The output table does not have any columns until you run the flow.



Step 2: Select Street as the Geocoding Method

1 In the Geocode Data step, select **Street** as the geocoding method.

2 Select the street method:

- **Default**
- **No city**
- **No zip**

3 Map these fields for geocoding:

- **Address**
- **City**

Note: This field is not available when **No city** is selected as the street method.

- **State/Province**

  Note: This field is not available when **No city** is selected as the street method.

- **Postal code**

  Note: This field is not available when **No zip** is selected as the street method.

Step 3: Specify Lookup Table

By default, the Geocode Data step uses SasHelp.USM as the lookup table. You do not need to map any fields for the city lookup table or the lookup table. You must specify the Lookup library where all the lookup reference data is stored.

(Optional) Step 4: Set Additional Options

Specify any optional arguments from the GEOCODE procedure.

(Optional) Step 5: Set the Debugging Options

To add extra SAS debugging for macros, select **Debug SAS macros**.

Step 6: Run the Flow and View the Output Table

To run the flow, click ▷ Run.

To view the output data, click the table node, and then click the **Preview Data** tab.

Items to note in the output table:

- Columns from the original data source are prefixed with _dm.
- The GEOCODE procedure generates a _MATCHED_ column that indicates how the coordinates were found.

Using the Plus4 Geocode Method

About the Plus4 Geocode Method

With Plus4 geocoding, the GEOCODE procedure attempts to match the five-digit ZIP code and ZIP+4 extension from your address data set with the lookup data set.

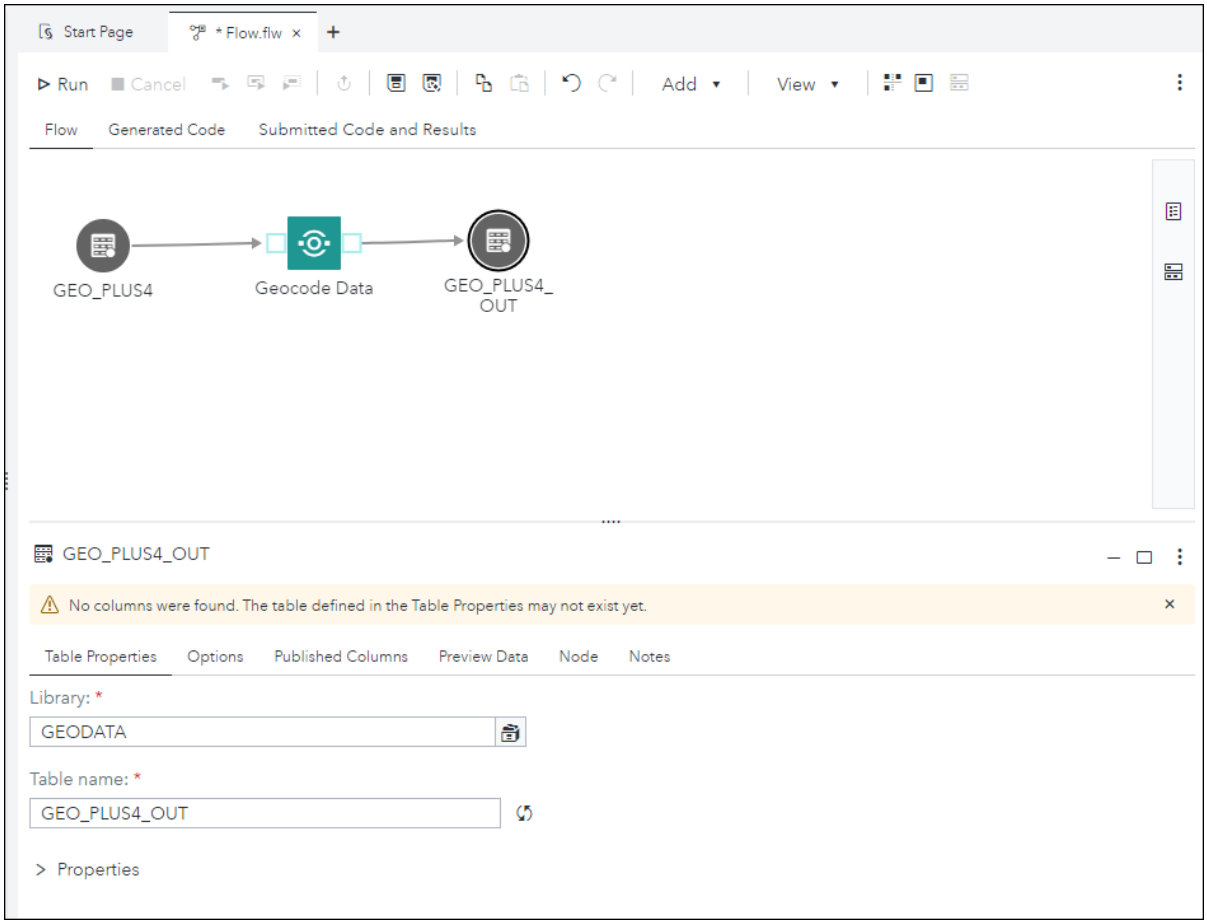
Plus4 Geocoding: Step-by-Step Instructions

Step 1: Add the Geocode Data Step and Connect a Source Table

1. In the **Steps** section of the navigation pane, expand the **Enrichment** folder and double-click **Geocode Data** to add the step to your flow.
2. Connect the input port of the Geocode Data 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 12.
3. To save the results from the Geocode Data step to a permanent output table, select **Add** ⇨ **Table**. A table node is added to the flow. Specify the library and name for the output table. (In this example, the output table is named Geo_Plus4_Out.)

   Next, connect the output port of the Geocode Data node to the table node.

   Note: The output table does not have any columns until you run the flow.



Step 2: Select Plus4 as the Geocoding Method

1 In the Geocode Data step, select **Plus4** as the geocoding method.

2 Map these fields for geocoding:

- **Postal code**
- **Zip+4**

Step 3: Specify Lookup Table

1 To view the labels for the ports on the Geocode Data node, right-click the Geocode Data node and select **Expand ports**. In the flow, the input port for the input data source (in this example, GEO_PLUS4) is labeled Input Table 1.

2 To add the input port and data for the lookup table:

a Right-click the Geocode Data node and select **Add input port** ⇨ **Lookup Table**.

b Add the lookup table to the flow. Connect the lookup table node to the Lookup Table 1 port.

c On the **Lookup** tab in the Geocode Data step, map these fields for the lookup table:
   - **Postal code**
   - **Zip+4**

(Optional) Step 4: Set Additional Options

Specify any optional arguments from the GEOCODE procedure.

(Optional) Step 5: Set the Debugging Options

To add extra SAS debugging for macros, select **Debug SAS macros**.

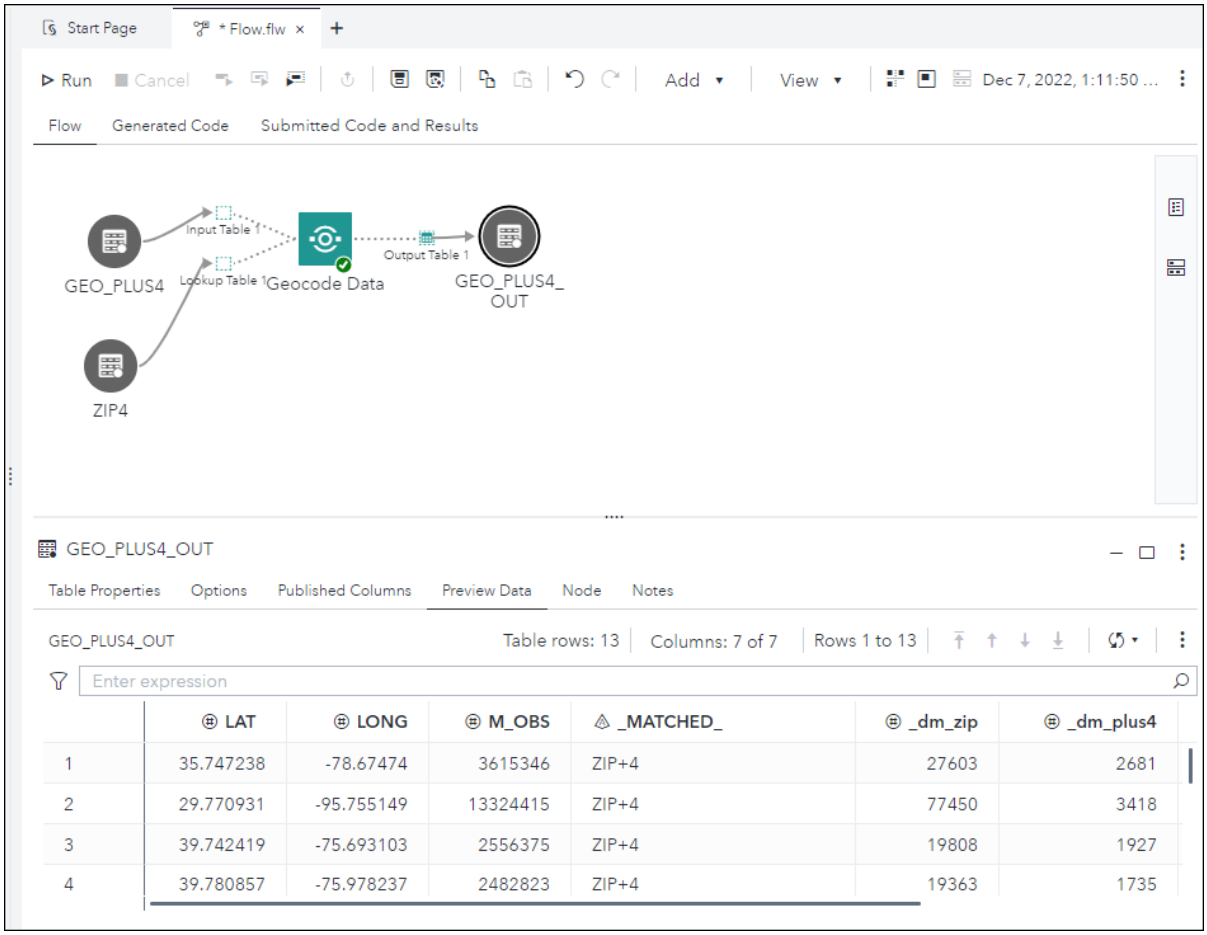
Step 6: Run the Flow and View the Output Table

To run the flow, click ▷ Run.

To view the output data, click the table node, and then click the **Preview Data** tab.

Items to note in the output table:

- Columns from the original data source are prefixed with _dm.
- The GEOCODE procedure generates a _MATCHED_ column that indicates how the coordinates were found.



Using the Zip Geocode Method

About the Zip Geocode Method

Your input data must contain a valid ZIP or postal code for each observation. If a ZIP code is not found, then the GEOCODE procedure attempts to find the city center location. If you are interested in the ZIP code location only, you can turn off this cascading behavior by selecting the **No city** option.

Zip Geocoding: Step-by-Step Instructions

Step 1: Add the Geocode Data Step and Connect a Source Table

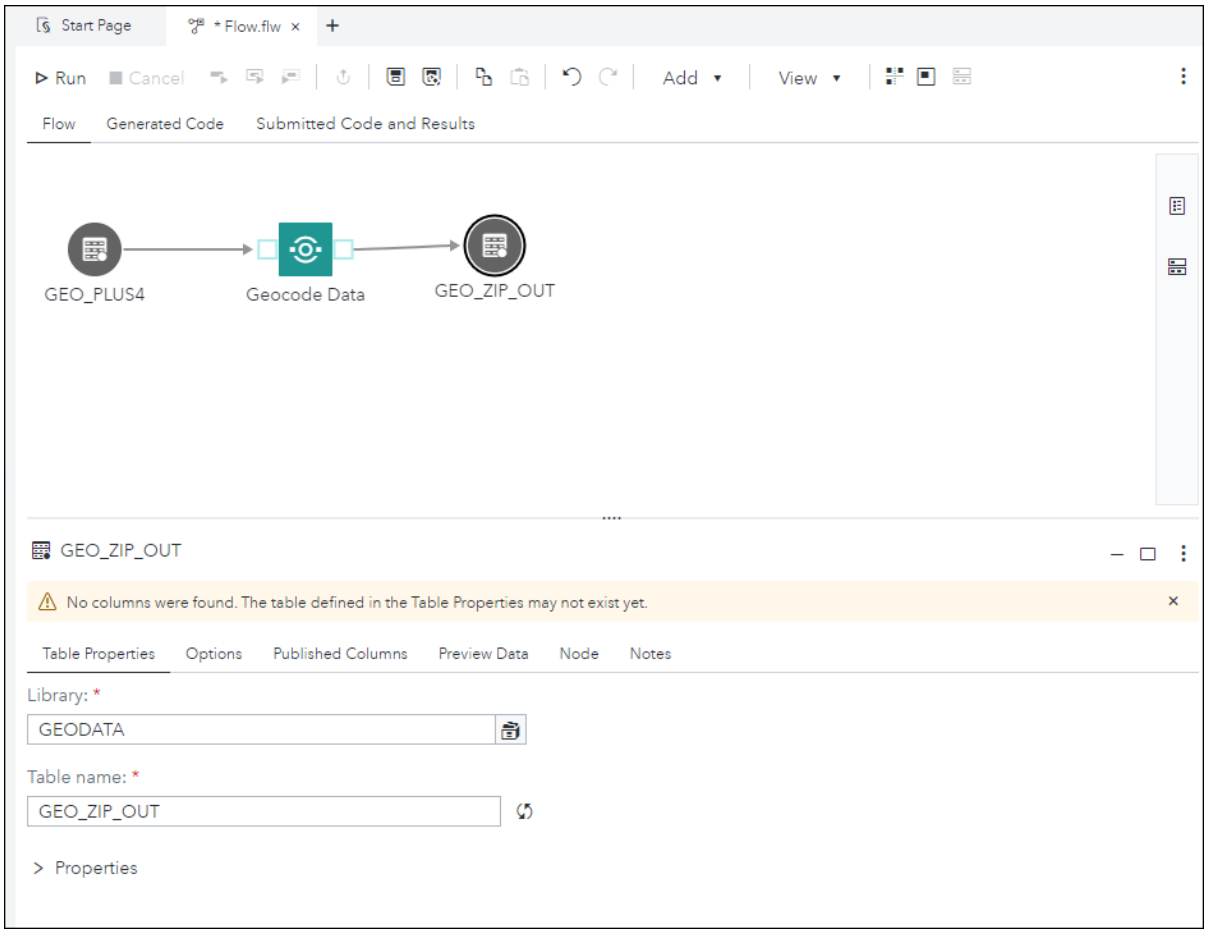
1 In the **Steps** section of the navigation pane, expand the **Enrichment** folder and double-click **Geocode Data** to add the step to your flow.

2 Connect the input port of the Geocode Data 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 12.

3 To save the results from the Geocode Data step to a permanent output table, select **Add** ⇨ **Table**. A table node is added to the flow. Specify the library and name for the output table. (In this example, the output table is named Geo_Zip_Out.)

Next, connect the output port of the Geocode Data node to the table node.

Note: The output table does not have any columns until you run the flow.



Step 2: Select Zip as the Geocoding Method

1. In the **Geocode Data** step, select **Zip** as the geocoding method.
2. Select the **No city** check box if your input data does not include a column for city mappings.
3. Map these fields for geocoding:
   - **City**

     Note: This field is not available if you select the **No city** check box.
   - **State/Province**

     Note: This field is not available if you select the **No city** check box.
   - **Postal code**

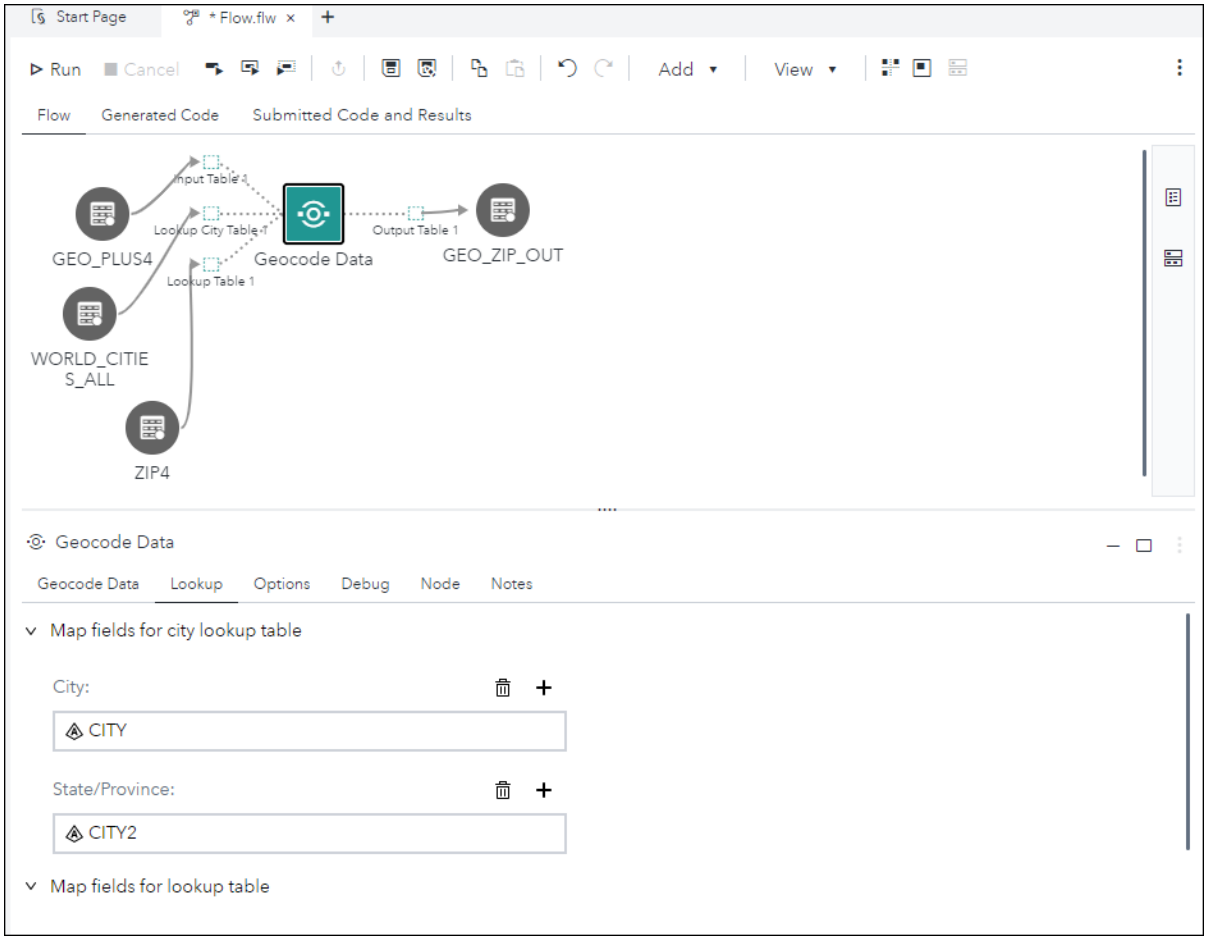
Step 3: Specify Lookup Table

For the Zip geocode method, you must specify a lookup table. If you did not select the **No city** check box on the **Geocode Data** tab, you also must add a city lookup table.

1. To view the labels for the ports on the Geocode Data node, right-click the Geocode Data node and select **Expand ports**. In the flow, the input port for the input data source (in this example, GEO_PLUS4) is labeled Input Table 1.
2. (Optional) Add the input port and data for the city lookup table.
   a. Right-click the Geocode node and select **Add input port** ⇨ **Lookup City Table**.
   b. Add the lookup table to the flow. (In this example, the lookup table is World_Cities_All.) Connect the lookup table node to the Lookup City Table 1 port.
   c. On the **Lookup** tab in the Geocode Data step, map these fields for the lookup table:
      - **City**
      - **State/Province**

   Note: This step is required only if you selected the **No city** check box on the Geocode Data tab.

3. To add the input port and data for the lookup table:
   a. Right-click the Geocode Data node again and select **Add input port** ⇨ **Lookup Table**.
   b. Add the lookup table to the flow. (In this example, the lookup table is ZIPCODE.) Connect the lookup table node to the Lookup Table 1 port.
   c. Map the **Postal code** field for the lookup table.



(Optional) Step 4: Set Additional Options

Specify any optional arguments from the GEOCODE procedure.

(Optional) Step 5: Set the Debugging Options

To add extra SAS debugging for macros, select **Debug SAS macros**.

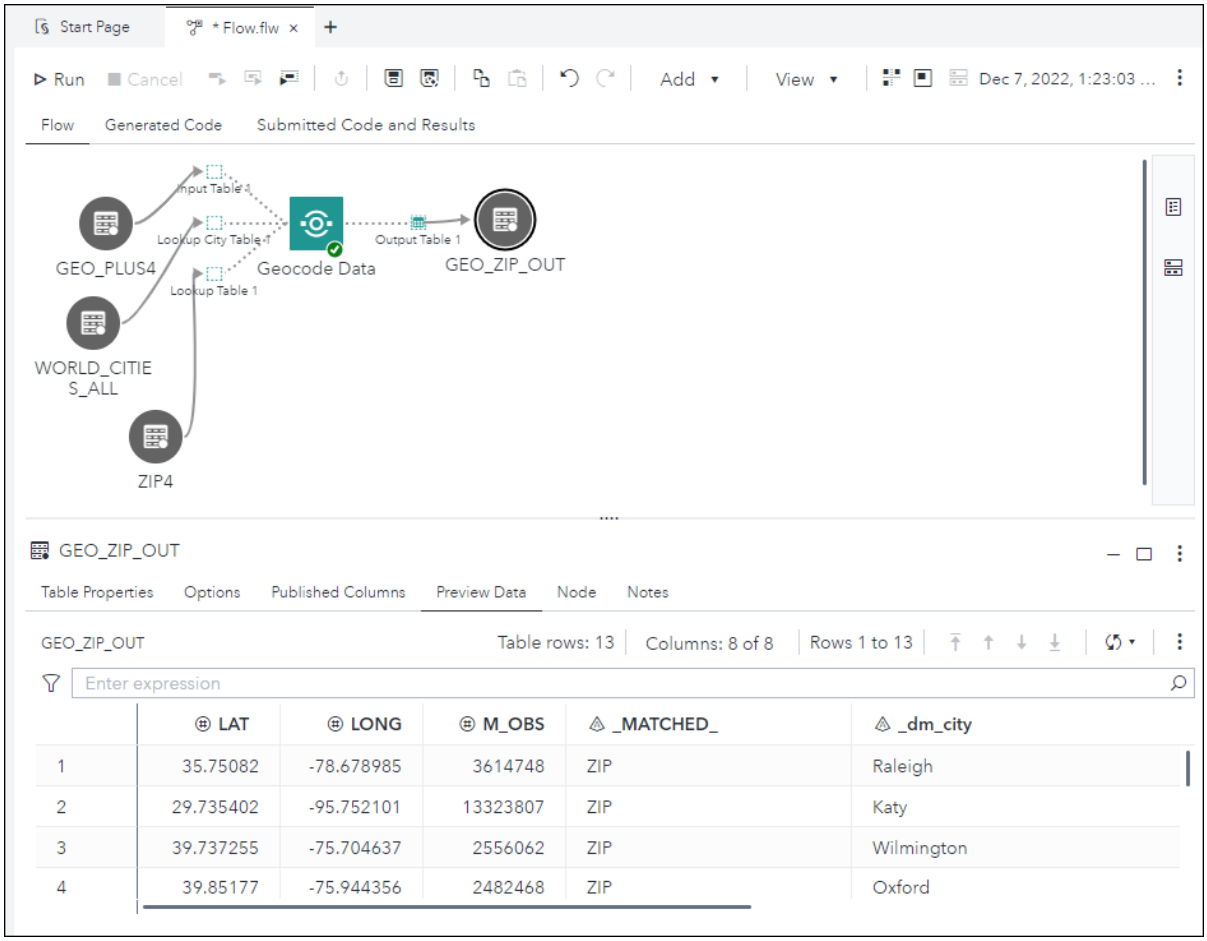
Step 6: Run the Flow and View the Output Table

To run the flow, click ▷ Run.

To view the output data, click the table node, and then click the **Preview Data** tab.

Items to note in the output table:

- Columns from the original data source are prefixed with _dm.
- The GEOCODE procedure generates a _MATCHED_ column that indicates how the coordinates were found.



Using the Range of IP Addresses Geocode Method

About the Range of IP Addresses Geocode Method

Range geocoding matches individual address values to a lookup data set that contains a range of values. IP address data is a form of range data. IP data was not designed to be geographic like street addresses. For this reason, the process of adding coordinates to IP addresses is usually called geolocating rather than geocoding.

Generally, IP addresses are collected from visitors to websites and indicate the connection the visitor used. IP address lookup data contains information that matches ranges of IP addresses to particular geographic locations. A range of IP addresses usually belongs to a company or an internet provider. The location found is not at the street- or even ZIP-code level, but might indicate the city, state, or country where the IP address is registered.

Range geocoding with IPv4 addresses requires a lookup data set and an additional range data set. Two data sets are required because a range of addresses can map to more than one location. Range geocoding with IPv6 addresses is possible and requires a single lookup data set that contains all geographic coordinates and the ranges of IPv6 addresses.

- The lookup data set contains geographic coordinates (latitude and longitude).
- The range data set identifies the ranges (of IPv4 addresses or of other items).

A location ID key variable links the two data sets. Both data sets must contain this variable in order to identify locations for each IPv4 range. Internally, the proper range is found, and then the key value is used to access the lookup data set to find the latitude and longitude for that key. This key variable can map a range of addresses in the range data set to more than one location in the lookup data set.

Range of IP Addresses Geocoding: Step-by-Step Instructions

Step 1: Add the Geocode Data Step and Connect a Source Table

1 In the **Steps** section of the navigation pane, expand the **Enrichment** folder and double-click **Geocode Data** to add the step to your flow.

2 Connect the input port of the Geocode Data 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 12.

3 To save the results from the Geocode Data step to a permanent output table, select **Add** ⇨ **Table**. A table node is added to the flow. Specify the library and name for the output table. (In this example, the output table is named Geo_Plus4_Out.)

   Next, connect the output port of the Geocode Data node to the table node.

   Note: The output table does not have any columns until you run the flow.

Step 2: Select Range of IP Addresses as the Geocoding Method

1 Select **Range of IP Addresses** as the geocoding method.

2 Map these fields for geocoding:

   - **Address**
   - **Begin range**

- **End range**

Step 3: Specify Lookup Table

For the Range of IP Addresses geocode method, you must specify a lookup table.

1. To view the labels for the ports on the Geocode Data node, right-click the Geocode node and select **Expand ports**. In the flow, the input port for the input data source is now labeled Input Table 1.
2. Right-click the Geocode Data node again and select **Add input port** ⇨ **Lookup Table**.
3. Add a table to the flow for the lookup table. Connect this table to the Lookup Table 1 port of the Geocode Data step.
4. On the **Lookup** tab in the Geocode Data step, map these fields for the lookup table:
   - **Key**
   - **Longitude**
5. Right-click the Geocode Data node again and select **Add input port** ⇨ **Range Table**.
6. Add a table to the flow for the range table. Connect this table to the Range Table 1 port of the Geocode Data step.
7. Map the key value.

(Optional) Step 4: Set Additional Options

Specify any optional arguments from the GEOCODE procedure.

(Optional) Step 5: Set the Debugging Options

To add extra SAS debugging for macros, select **Debug SAS macros**.

Step 6: Run the Flow and View the Output Table

To run the flow, click ▷ Run.

To view the output data, click the table node, and then click the **Preview Data** tab.

Items to note in the output table:

- Columns from the original data source are prefixed with _dm.
- The GEOCODE procedure generates a _MATCHED_ column that indicates how the coordinates were found.

Using the Custom Geocode Method

About the Custom Geocode Method

Any data can be used as lookup data with the Custom method of geocoding. The only requirement is that you have at least three variables. The variables must be the projected or unprojected coordinate values (X and Y or LAT and LONG) and include a key variable to look up.

Custom Geocoding: Step-by-Step Instructions

Step 1: Add the Geocode Data Step and Connect a Source Table

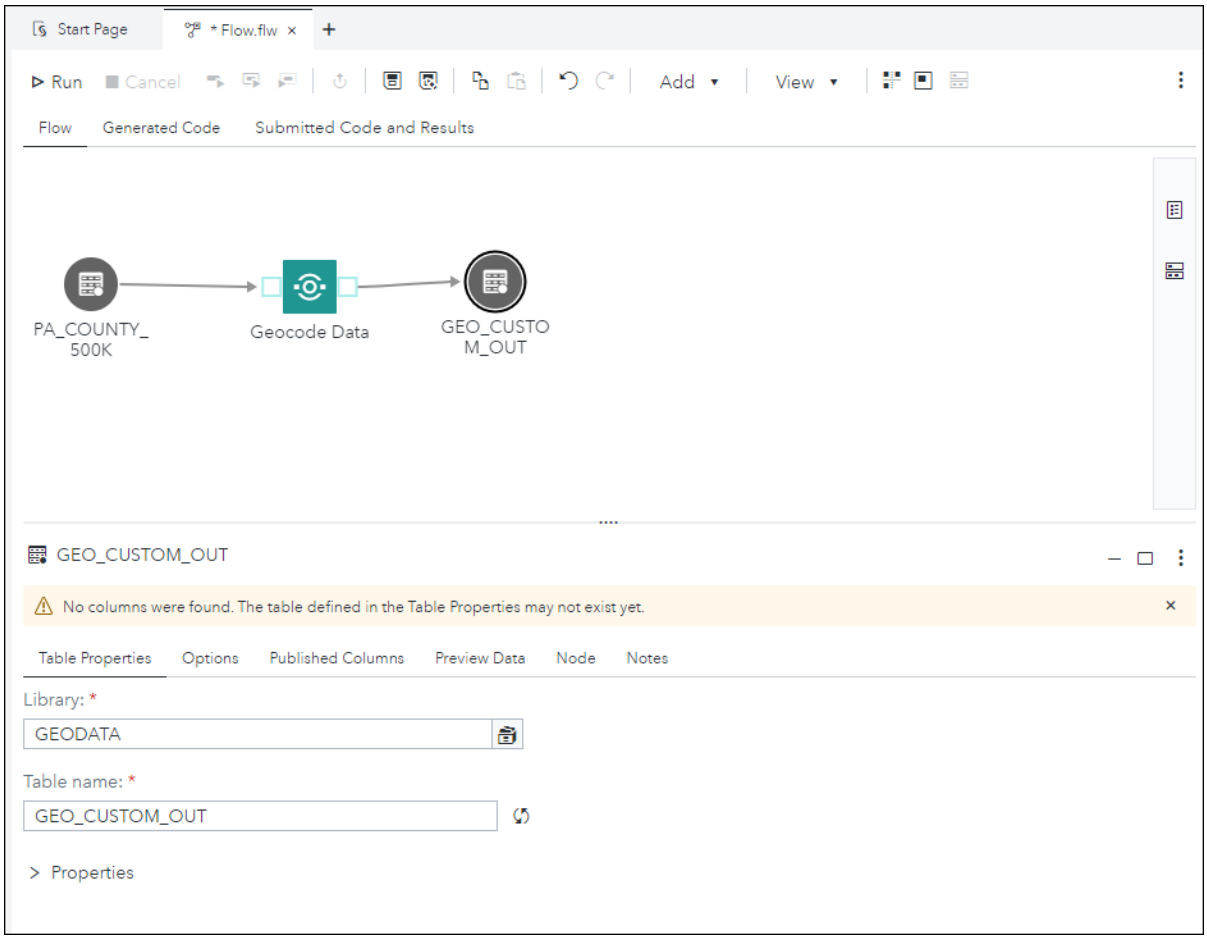
1 In the **Steps** section of the navigation pane, expand the **Enrichment** folder and double-click **Geocode Data** to add the step to your flow.

2 Connect the input port of the Geocode Data 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 12.

3 To save the results from the Geocode Data step to a permanent output table, select **Add** ⇨ **Table**. A table node is added to the flow. Specify the library and name for the output table. (In this example, the output table is named Geo_Custom_Out.)

Next, connect the output port of the Geocode Data node to the table node.

Note: The output table does not have any columns until you run the flow.



Step 2: Select Custom as the Geocoding Method

1 Select **Custom** as the geocoding method.

2 Map the **Address** field.

Step 3: Specify Lookup Table

For the Custom geocode method, you must specify a lookup table.

1 To view the labels for the ports on the Geocode Data node, right-click the Geocode node and select **Expand ports**. In the flow, the input port for the input data source is now labeled Input Table 1.

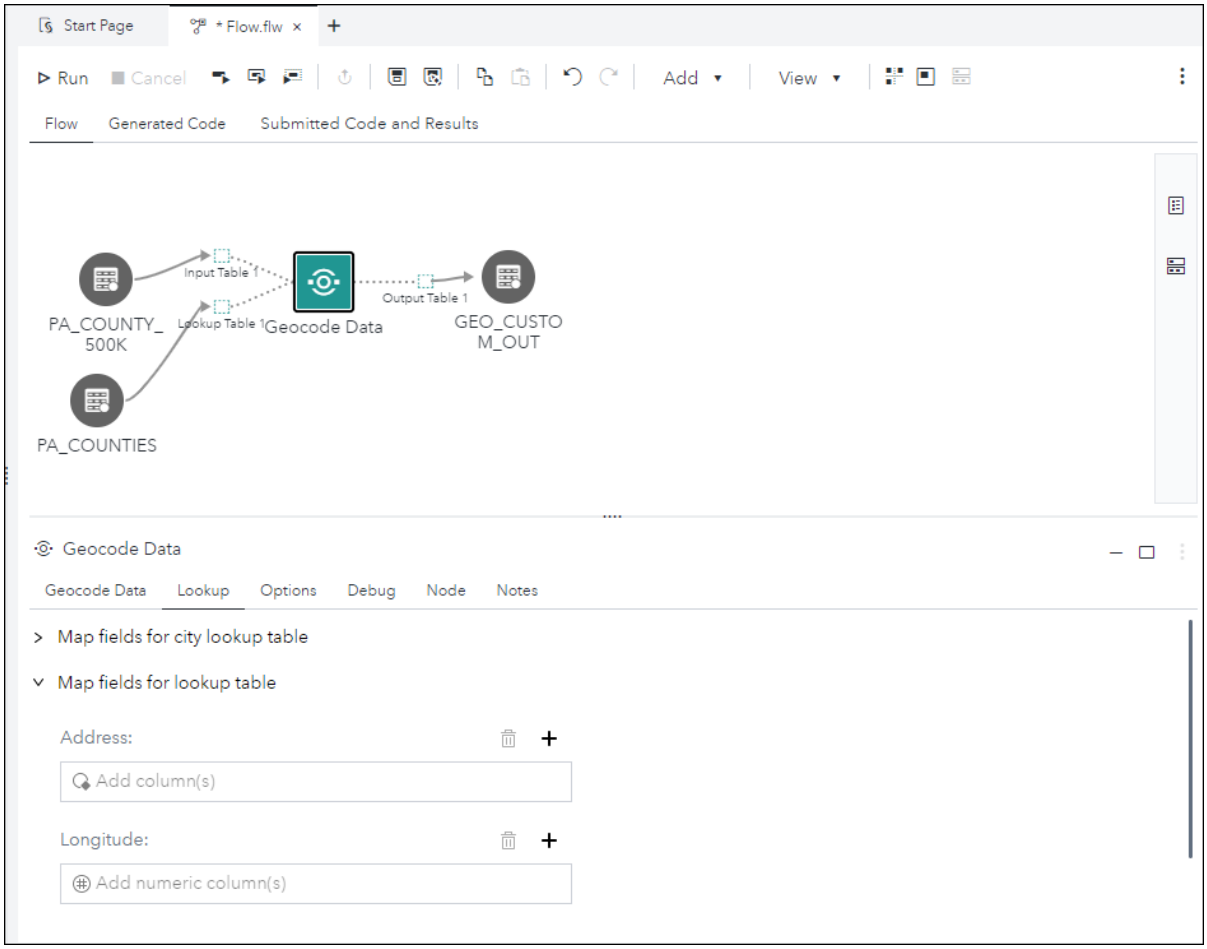
2 Right-click the Geocode node again and select **Add input port** ⇨ **Lookup Table**.

3 Add the lookup table to the flow. Connect the lookup table node to the Lookup Table 1 port.

4 On the **Lookup** tab in the Geocode Data step, map these fields for the lookup table:

- **Address**

- **Longitude**
- **Latitude**



(Optional) Step 4: Set Additional Options

Specify any optional arguments from the GEOCODE procedure.

(Optional) Step 5: Set the Debugging Options

To add extra SAS debugging for macros, select **Debug SAS macros**.

Step 6: Run the Flow and View the Output Table

To run the flow, click ▷ Run.

To view the output data, click the table node, and then click the **Preview Data** tab.

Items to note in the output table:

- Columns from the original data source are prefixed with _dm.
- The GEOCODE procedure generates a _MATCHED_ column that indicates how the coordinates were found.

Verifying Addresses

About the Verify & Geocode Addresses - Loqate Step

The Verify & Geocode Addresses - Loqate step enables you to verify address information and obtain geocode coordinates from the Loqate Verify Address API. Loqate is a third-party provider that specializes in address verification. The step uses PROC HTTP to connect to the Loqate Verify Address API. The Loqate API returns JSON code to SAS with the enriched data.

Note: Loqate associates a cost for each verification. For more information, see the Loqate website.

Node Connection Requirements

The Verify & Geocode Address step includes an input port and an output port. In order to run the Verify & Geocode Addresses - Loqate step, you must create the input port 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	No connection is required. **Note:** By default, the output data is written to a temporary table in the Work library. You can specify the library and name of the output table by connecting the output port to a Table node. For more information, see “Adding a Table from a SAS Library to a Flow” on page 34.

Verify & Geocode Addresses - Loqate: Step-by-Step Instructions

Step 1: Add the Verify & Geocode Addresses - Loqate Step and Connect a Source Table

To add the Verify & Geocode Addresses - Loqate step to your flow:

1. In the **Steps** section of the navigation pane, expand the **Enrichment** folder and double-click **Verify & Geocode Addresses - Loqate**.
2. Connect the input port of the Verify & Geocode Addresses - Loqate 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 12.

Note: The data source must include address and country information.

Here is an example of the sample_addresses table.

Start Page | sample_addresses × | * Flow.flw | +

sample_addresses — Table rows: 19 | Columns: 6 of 6 | Rows 1 to 19

Enter expression

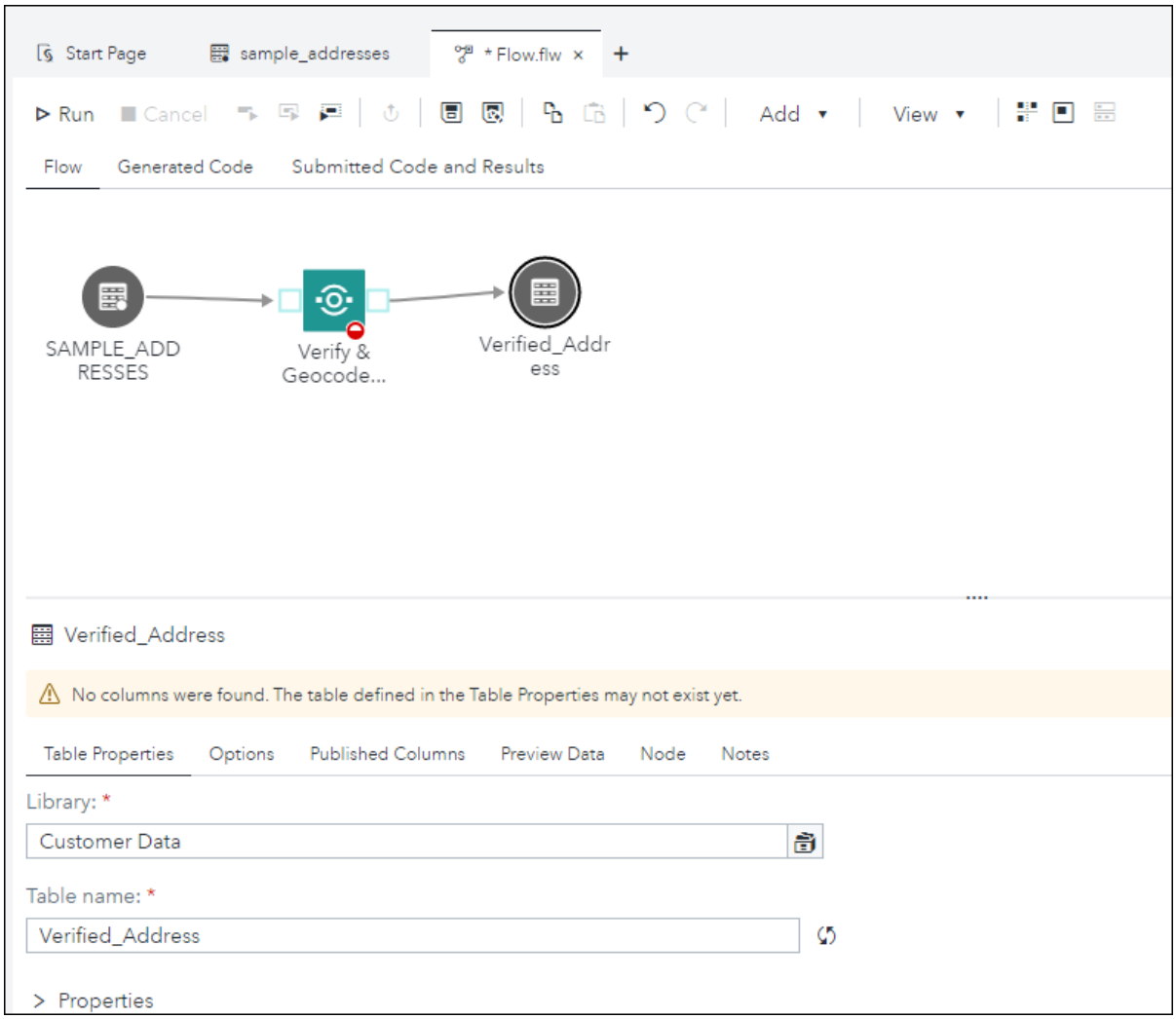
	id	address	pc	city	state	country
1	1	100 SAS Campus Dr	27513		NC	United States
2	2	385 Bourke St	VIC 3004	Melbourne	VIC	AUS
3	3	1 Eagle Street	4000	Brisbane	QLD	AUS
4	4	10188 Telesis Court Suite 200		San Diego		USA
5	5	Domaine de Grégy Grégy-sur-Yerres	77257			FRA
6	6	787 Seventh Ave.	10019			United States of America
7	7	1530 Wilson Blvd. Suite 800		Arlington		US
8	8	Piazza della Repubblica 68		Roma		ITA
9	9	One PPG Place Suite 2950	15222	Pittsburgh	Pennsylvania	US
10	10	Via Confienza 10	10121			Italia
11	11	Monroe Park Towers 101 N. Monroe St. …		Tallahassee	FL	U.S.A.
12	12	121 W. Trade St.	28202			US
13	13	Tour Ariane 27ème étage 5 place de la …	92800	PARIS LA DEFE…		France
14	14	Via Darwin 20/22	20143			Italy
15	15	111 Rockville Pike Suite 900	20850			U.S.A.
16	16	300 Burns Bay Road	2066	Lane Cove		Australia
17	17	Two Prudential Plaza 180 N. Stetson St. …	60601			US

3. To save the results from the Verify & Geocode Addresses - Loqate step to a permanent output table, select **Add** ⇨ **Table**. A table node is added to the flow.

Specify the library and name for the output table. (In this example, the output table is named Verified_Address.)

Next, connect the output port of the Verify & Geocode Addresses - Loqate node to the table node.

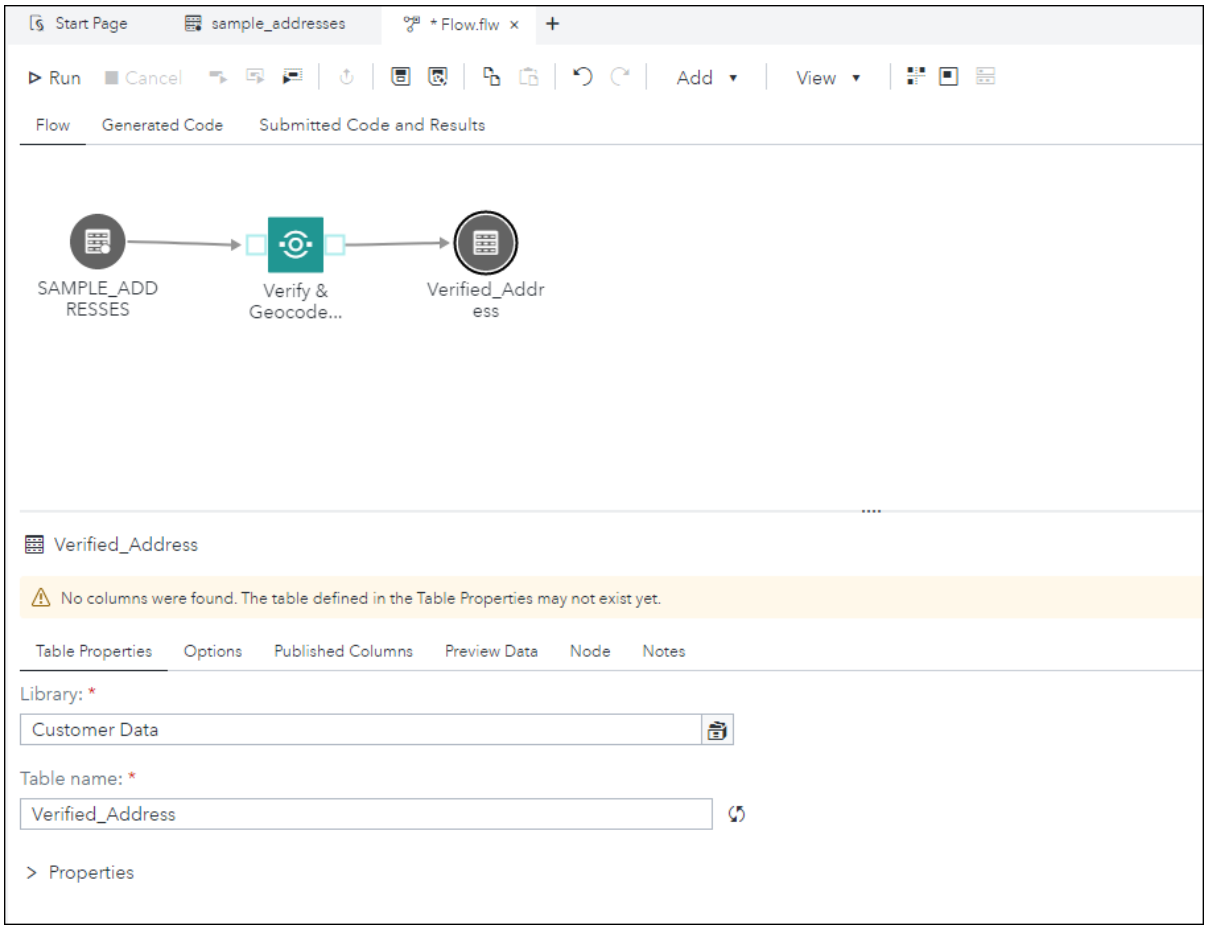
Note: The output table does not have any columns until you run the flow.



Step 2: Map the Fields for Address Verification

Use the options to map the columns in your data source to the corresponding fields in the Loqate database. Depending on your data, you might not specify values for all of these fields. The **Country** and **Address 1** fields are required.

For the sample_addresses data source, the country column is mapped to the **Country** field, and the address column is mapped to the **Address 1** field.



(Optional) Step 3: Set Additional Options

1. To retrieve the latitude and longitude coordinates for the address from Loqate, select **Geocode the address**.
2. To standardize the country code in your data source, select **Perform country ISO standardization before processing**.

   Note: The Loqate Verify Address API requires that country name or code be in an ISO 3166 two-character country code or ISO 3166 three-character country code. When you select the **Perform country ISO standardization before processing** option, SAS Studio standardizes the country value to an ISO 3166 three-character country code. If this option is not selected, the Loqate Verify Address API processes only country values in the ISO two-character code or ISO three-character code format. In the sample_addresses data source, some countries are in an ISO format (such as AUS, which is in the ISO 3166 three-character country code format), but other values (such as United States of America) are not. In order for the Loqate Verify Address API to process all these country values correctly, you need to select the **Perform country ISO standardization before processing** option.

3 Specify the number of records to send to Loqate at one time. The maximum value is 500.

4 Specify whether to replace the existing output table.

Step 4: Set the Debugging Options

Each call to the Loqate database incurs a cost. Therefore, you might want to set these debugging and logging options to identify and quickly troubleshoot problems in the flow before running against the Loqate Verify Address API.

1 To show the request from SAS and the response from Loqate in the log, select **Show API input and output JSON in the log** .

2 To test your flow without making the call to Loqate, select **Test mode**.

> **Note:** Because each call to the Loqate API incurs a cost, the **Test mode** option is selected by default. When you are ready to run your flow against the Loqate API, deselect this check box.

3 To add extra SAS debugging for macros, select **Debug SAS macros**.

Step 5: Specify the Loqate Key

A Loqate key is required to run this step. You can get this key from the Loqate website.

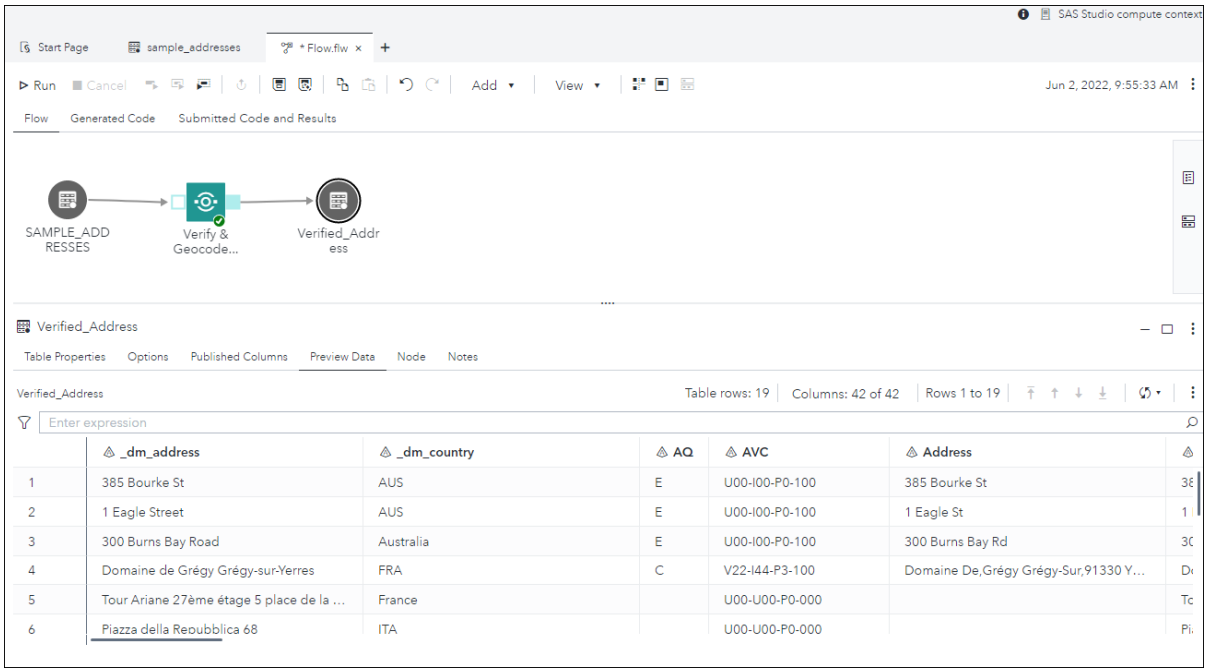
Step 6: Run the Flow and View the Output Table

To run the flow, click ▷ Run.

To view the output data, click the table node, and then click the **Preview Data** tab.

Items to note in the output table:

- Columns from the SAS data source that were mapped to Loqate fields are prefixed with _dm.
- The remaining columns are from the Loqate Verify Address API. To understand the codes in your results, see the Loqate documentation.



Verifying Email Addresses

About the Verify Email Addresses - Loqate Step

The Verify Email Addresses - Loqate step enables you to verify email addresses that use the Loqate Verify Email Addresses API. Loqate is a third-party provider that specializes in email address verification. The step uses PROC HTTP to connect to the Loqate Verify Email Addresses API. The Loqate API returns a CSV file to SAS with the enriched data.

Note: Loqate associates a cost for each verification. For more information, see the Loqate website.

Node Connection Requirements

The Verify Email Addresses - Loqate step includes an input port and an output port. In order to run the Verify Email Addresses - Loqate step, you must create the input port 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	No connection is required. **Note:** By default, the output data is written to a temporary table in the Work library. You can specify the library and name of the output table by connecting the output port to a Table node. For more information, see “Adding a Table from a SAS Library to a Flow” on page 34.

Verify Email Addresses - Loqate: Step-by-Step Instructions

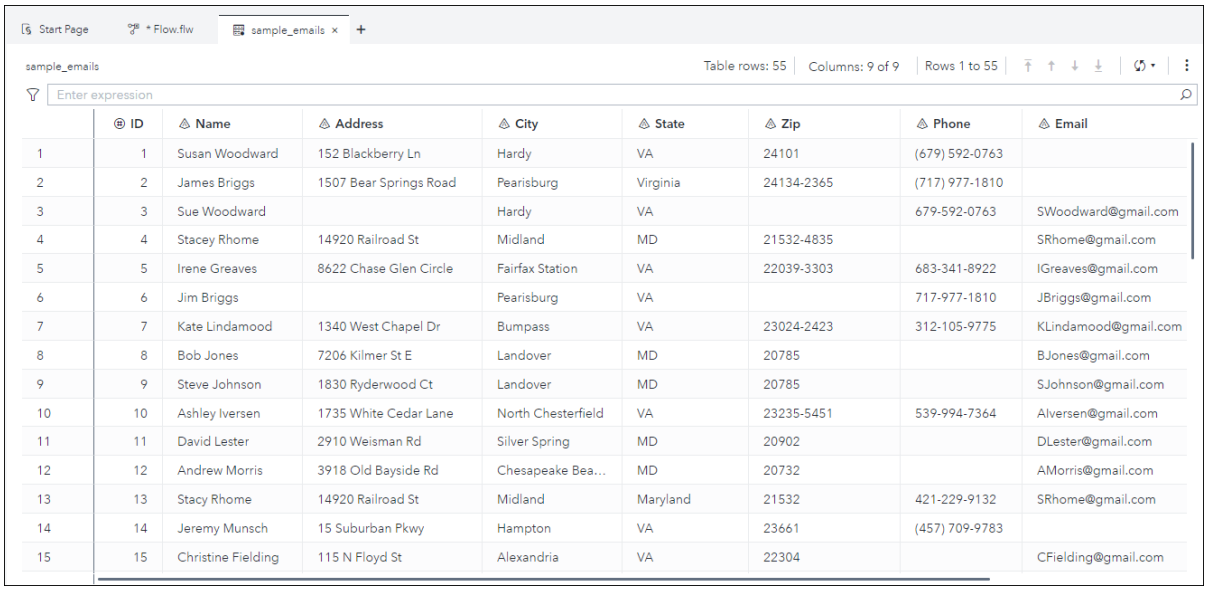
Step 1: Add the Verify Email Addresses - Loqate Step and Connect a Source Table

1 In the **Steps** section of the navigation pane, expand the **Enrichment** folder and double-click **Verify Email Addresses - Loqate** to add the step to your flow.

2 Connect the input port of the Verify Email Addresses - Loqate 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 12.

Note: The data source must include email addresses.

Here is an example of the sample_emails table.



Start Page | * Flow.flw | sample_emails ×

sample_emails — Table rows: 55 | Columns: 9 of 9 | Rows 1 to 55

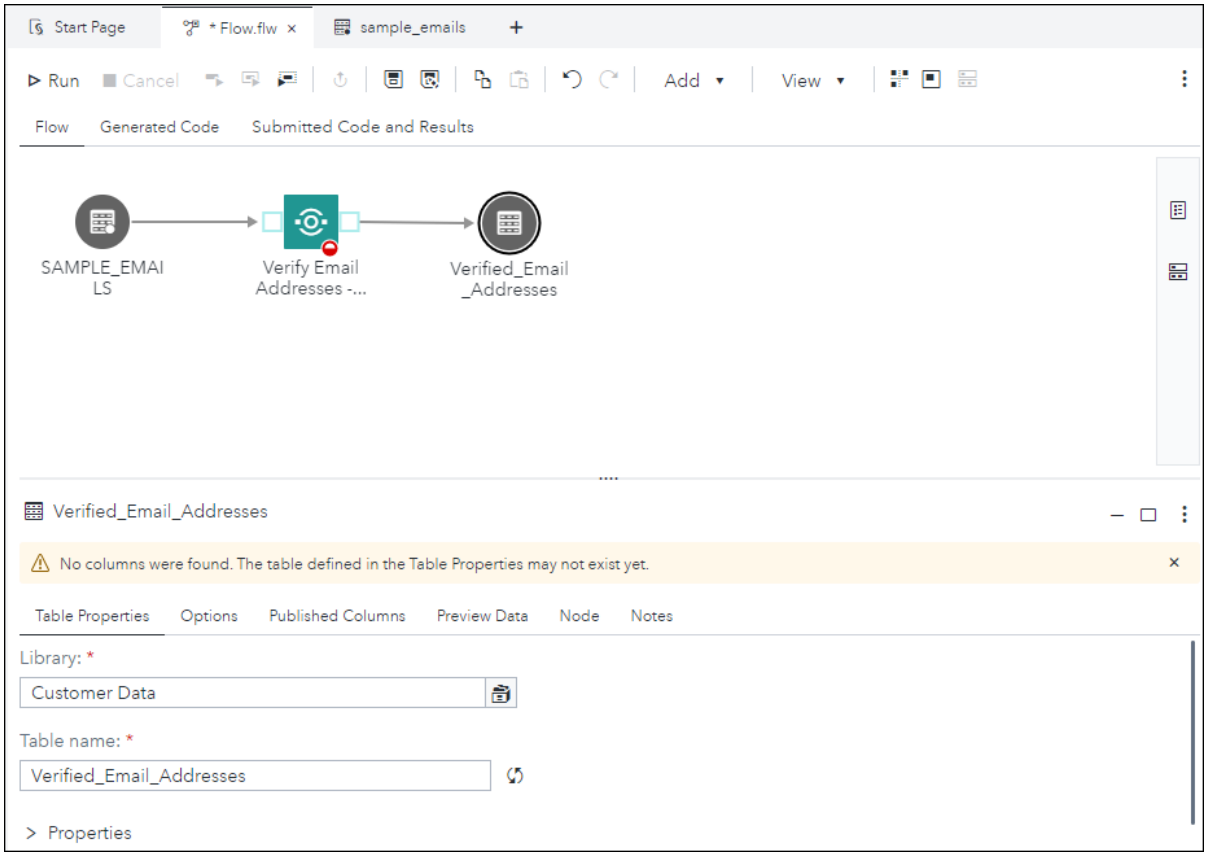
Enter expression

	ID	Name	Address	City	State	Zip	Phone	Email
1	1	Susan Woodward	152 Blackberry Ln	Hardy	VA	24101	(679) 592-0763	
2	2	James Briggs	1507 Bear Springs Road	Pearisburg	Virginia	24134-2365	(717) 977-1810	
3	3	Sue Woodward		Hardy	VA		679-592-0763	SWoodward@gmail.com
4	4	Stacey Rhome	14920 Railroad St	Midland	MD	21532-4835		SRhome@gmail.com
5	5	Irene Greaves	8622 Chase Glen Circle	Fairfax Station	VA	22039-3303	683-341-8922	IGreaves@gmail.com
6	6	Jim Briggs		Pearisburg	VA		717-977-1810	JBriggs@gmail.com
7	7	Kate Lindamood	1340 West Chapel Dr	Bumpass	VA	23024-2423	312-105-9775	KLindamood@gmail.com
8	8	Bob Jones	7206 Kilmer St E	Landover	MD	20785		BJones@gmail.com
9	9	Steve Johnson	1830 Ryderwood Ct	Landover	MD	20785		SJohnson@gmail.com
10	10	Ashley Iversen	1735 White Cedar Lane	North Chesterfield	VA	23235-5451	539-994-7364	AIversen@gmail.com
11	11	David Lester	2910 Weisman Rd	Silver Spring	MD	20902		DLester@gmail.com
12	12	Andrew Morris	3918 Old Bayside Rd	Chesapeake Bea...	MD	20732		AMorris@gmail.com
13	13	Stacy Rhome	14920 Railroad St	Midland	Maryland	21532	421-229-9132	SRhome@gmail.com
14	14	Jeremy Munsch	15 Suburban Pkwy	Hampton	VA	23661	(457) 709-9783	
15	15	Christine Fielding	115 N Floyd St	Alexandria	VA	22304		CFielding@gmail.com

3 To save the results from the Verify Email Addresses - Loqate step to a permanent output table, select **Add** ⇨ **Table**. A table node is added to the flow. Specify the library and name for the output table. (In this example, the output table is named Verified_Email_Addresses.)

Next, connect the output port of the Verify Email Addresses - Loqate node to the table node.

Note: The output table does not have any columns until you run the flow.



Step 2: Map the Fields for Email Address Verification

Use the options to map the columns in your data source to the corresponding fields in the Loqate database. The **Email address** field is required. Only English characters are supported.

For the sample_emails data source, the Email column is mapped to the **Email** field.

(Optional) Step 3: Set Additional Options

1. In the **Batch size** field, specify the number of records to send to Loqate at one time. The maximum value is 100.
2. Specify whether to replace the existing output table.

Step 4: Set the Debugging Options

Each call to the Loqate database incurs a cost. Therefore, you might want to set these debugging and logging options to identify and quickly troubleshoot problems in the flow before running against the Loqate Verify Email Address API.

1. To show the request from SAS and the response from Loqate in the log, select **Show API Input/Output CSV in the log** .
2. To test your flow without making the call to Loqate, select **Test mode**.

   Note: Because each call to the Loqate API incurs a cost, the **Test mode** option is selected by default. When you are ready to run your flow against the Loqate API, deselect this check box.
3. To add extra SAS debugging for macros, select **Debug SAS macros**.

Step 5: Specify the Loqate Key

A Loqate key is required to run this step. You can get this key from the Loqate website.

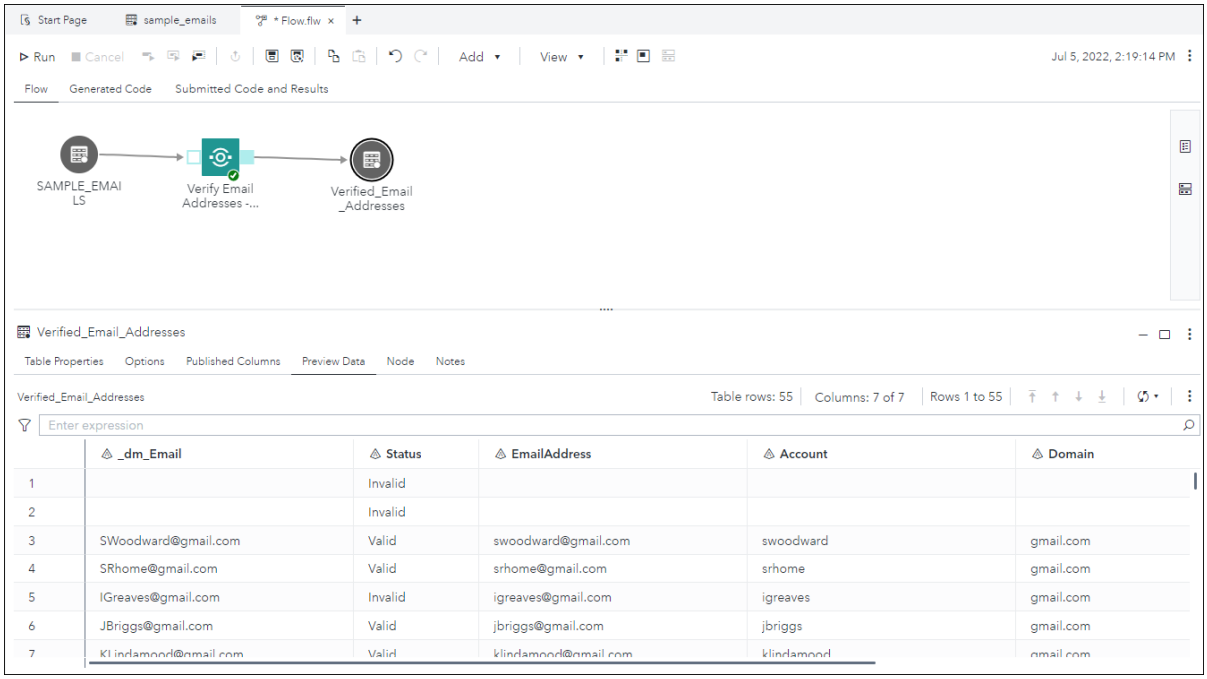
Step 6: Run the Flow and View the Output Table

To run the flow, click ▷ Run.

To view the output data, click the table node, and then click the **Preview Data** tab.

Items to note in the output table:

- Columns from the SAS data source that were mapped to Loqate fields are prefixed with _dm.
- The remaining columns are from the Loqate API. To understand the codes in your results, see the Loqate documentation.



	_dm_Email	Status	EmailAddress	Account	Domain
1		Invalid			
2		Invalid			
3	SWoodward@gmail.com	Valid	swoodward@gmail.com	swoodward	gmail.com
4	SRhome@gmail.com	Valid	srhome@gmail.com	srhome	gmail.com
5	IGreaves@gmail.com	Invalid	igreaves@gmail.com	igreaves	gmail.com
6	JBriggs@gmail.com	Valid	jbriggs@gmail.com	jbriggs	gmail.com
7	KLindamood@gmail.com	Valid	klindamood@gmail.com	klindamood	gmail.com

Verifying Phone Numbers

About the Verify Phone Numbers - Loqate Step

The Verify Phone Numbers - Loqate step enables you to verify phone numbers that use the Loqate Verify Phone Numbers API. Loqate is a third-party provider that specializes in phone number verification. The step uses PROC HTTP to connect to

the Loqate Verify Phone Numbers API. The Loqate API returns a CSV code to SAS with the enriched data.

Note: Loqate associates a cost for each verification. For more information, see the Loqate website.

Node Connection Requirements

The Verify Phone Numbers - Loqate step includes an input port and an output port. In order to run the Verify Phone Numbers - Loqate step, you must create the input port 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	No connection is required. **Note:** By default, the output data is written to a temporary table in the Work library. You can specify the library and name of the output table by connecting the output port to a Table node. For more information, see “Adding a Table from a SAS Library to a Flow” on page 34.

Verify Phone Numbers - Loqate: Step-by-Step Instructions

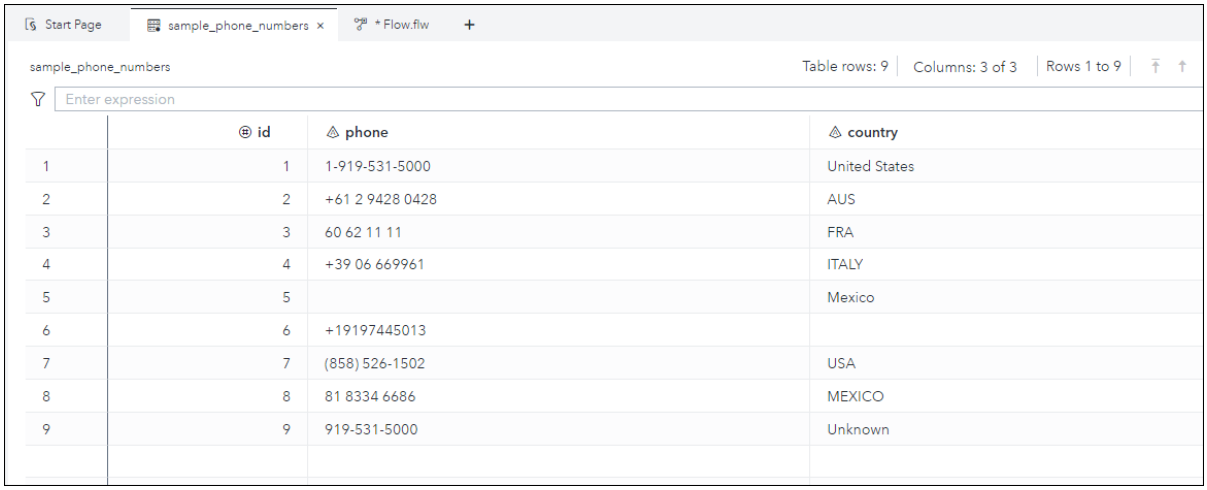
Step 1: Add the Verify Phone Numbers - Loqate Step and Connect a Source Table

To add the Verify Phone Numbers - Loqate step to your flow:

1. In the **Steps** section of the navigation pane, expand the **Enrichment** folder and double-click **Verify Phone Numbers - Loqate**.
2. Connect the input port of the Verify Phone Numbers - Loqate 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 12.

   Note: The data source must include phone information.

Here is an example of the sample_phone_numbers table.


Start Page | sample_phone_numbers × | * Flow.flw | +

sample_phone_numbers — Table rows: 9 | Columns: 3 of 3 | Rows 1 to 9

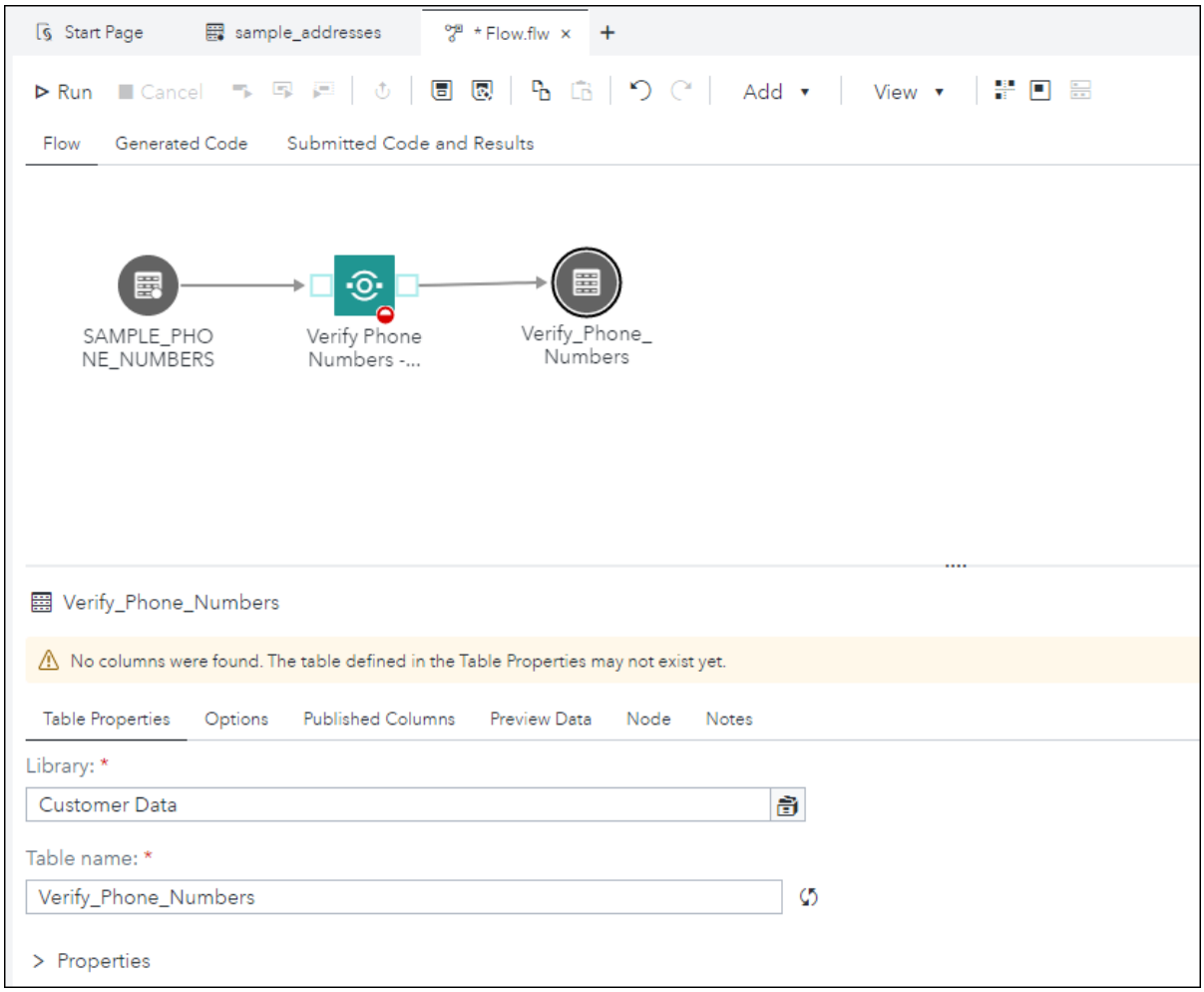
Enter expression

	id	phone	country
1	1	1-919-531-5000	United States
2	2	+61 2 9428 0428	AUS
3	3	60 62 11 11	FRA
4	4	+39 06 669961	ITALY
5	5		Mexico
6	6	+19197445013	
7	7	(858) 526-1502	USA
8	8	81 8334 6686	MEXICO
9	9	919-531-5000	Unknown

3 To save the results from the Verify Phone Numbers - Loqate step to a permanent output table, select **Add** ⇨ **Table**. A table node is added to the flow. Specify the library and name for the output table. (In this example, the output table is named Verified_Phone_Numbers.)

Next, connect the output port of the Verify Phone Numbers - Loqate node to the table node.

Note: The output table does not have any columns until you run the flow.



Step 2: Map the Fields for Phone Verification

Use the options to map the columns in your data source to the corresponding fields in the Loqate database. The **Phone number** field is required.

For the sample_phone_numbers data source, the phone column is mapped to the **Phone number** field, and the country column is mapped to the **Country** field.

(Optional) Step 3: Set Additional Options

The Loqate Verify Phone API requires that country name be in an ISO 3166 two-character country code format. When the input data source contains a country field and you select the **Perform country ISO standardization before processing** option, SAS Studio standardizes the country value to an ISO 3166 two-character country code. If this option is not selected, the Loqate Verify Phone Numbers API processes only country values in the ISO two-character code format.

In the sample_phone_numbers data source, none of the country values are in an ISO 3166 two-character country code format. In order for the Loqate Verify Phone Numbers API to process all these country values correctly, you need to select the **Perform country ISO standardization before processing** option.

Step 4: Set the Debugging Options

Each call to the Loqate database incurs a cost. Therefore, you might want to set these debugging and logging options to identify and quickly troubleshoot problems in the flow before running against the Loqate Verify Phone Numbers API.

1. To show the request from SAS and the response from Loqate in the log, select **Show API input and output CSV in the log** .
2. To test your flow without making the call to Loqate, select **Test mode**.

   Note: Because each call to the Loqate API incurs a cost, the **Test mode** option is selected by default. When you are ready to run your flow against the Loqate API, deselect this check box.

3. To add extra SAS debugging for macros, select **Debug SAS macros**.

Step 5: Specify the Loqate Key

A Loqate key is required to run this step. You can get this key from the Loqate website.

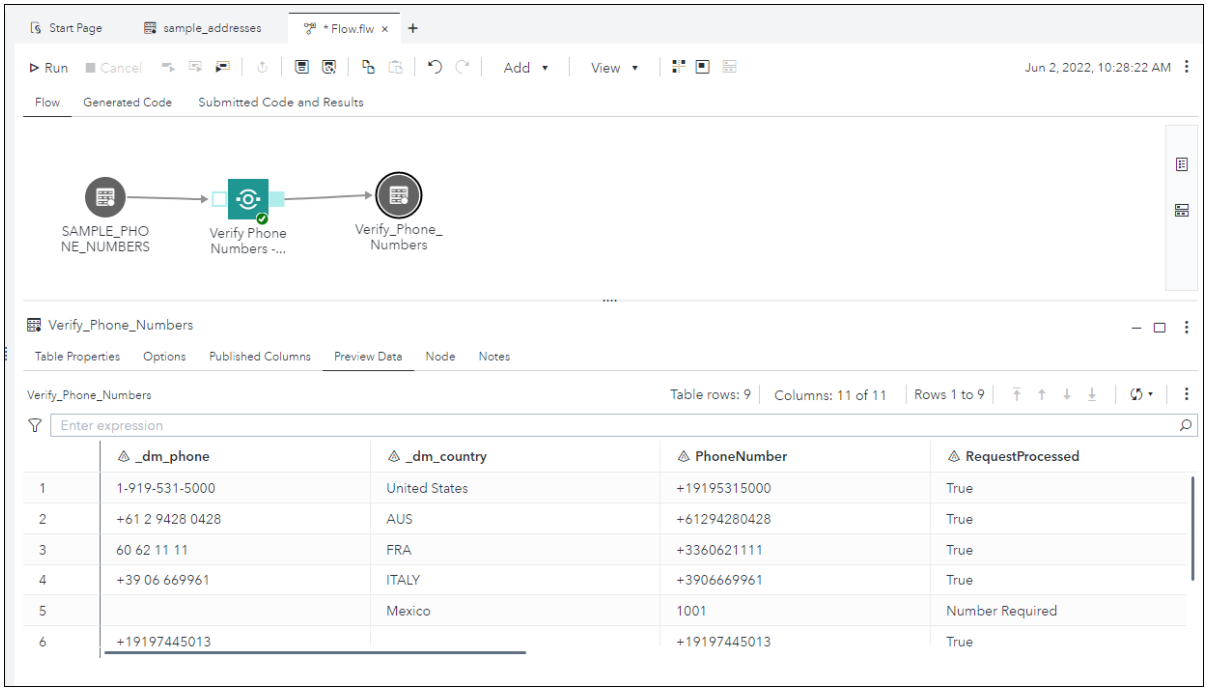
Step 6: Run the Flow and View the Output Table

To run the flow, click ▷ Run.

To view the output data, click the table node, and then click the **Preview Data** tab.

Items to note in the output table:

- Columns from the SAS data source that were mapped to Loqate fields are prefixed with _dm.
- The remaining columns are from the Loqate API. To understand the codes in your results, see the Loqate documentation.


Start Page
sample_addresses
* Flow.flw
Run
Cancel
Add
View
Jun 2, 2022, 10:28:22 AM
Flow
Generated Code
Submitted Code and Results
SAMPLE_PHONE_NUMBERS
Verify Phone Numbers -...
Verify_Phone_Numbers
Verify_Phone_Numbers
Table Properties
Options
Published Columns
Preview Data
Node
Notes
Verify_Phone_Numbers
Table rows: 9
Columns: 11 of 11
Rows 1 to 9
Enter expression
_dm_phone
_dm_country
PhoneNumber
RequestProcessed
1 1-919-531-5000 United States +19195315000 True
2 +61 2 9428 0428 AUS +61294280428 True
3 60 62 11 11 FRA +3360621111 True
4 +39 06 669961 ITALY +3906669961 True
5 Mexico 1001 Number Required
6 +19197445013 +19197445013 True

8

Generating Statistics

One-Way Frequencies

About the One-Way Frequencies Step

The One-Way Frequencies step generates frequency tables from your data. You can also use this step to perform binomial and chi-square tests.

You might want to use this step to analyze the efficiency of a new drug. For example, suppose a group of medical researchers are interested in evaluating the efficacy of a new treatment for a skin condition. Dermatologists from participating clinics are trained to conduct the study and to evaluate the condition. After the training, two dermatologists examine patients with the skin condition from a pilot study and rate the same patients. The One-Way Frequencies step can be used to evaluate the agreement of the diagnoses.

Node Connection Requirements

In order to run the One-Way Frequencies step, you must create the input port of the node as indicated here:

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

One-Way Frequencies: Assigning Data to Roles

To run the One-Way Frequencies step, you must assign a column to the **Analysis variables** role.

To filter the input data source, type the filter expression in the **Filter** field.

Table 8.1 *Roles in the One-Way Frequencies Step*

Role	Description
Roles	
Analysis variables	Specifies the variables to be analyzed. For each variable that you assign to this role, the task creates a one-way frequency table. You must assign at least one variable to this role.
Additional Roles	
Frequency count	Lists a numeric variable whose value represents the frequency of the observation. If you assign a variable to this role, the task assumes that each observation represents n observations, where *n* is the value of the frequency variable. If *n* is not an integer, SAS truncates it. If *n* is less than 1 or is missing, the observation is excluded from the analysis. The sum of the frequency variable represents the total number of observations.
Group analysis by	Enables you to obtain separate analyses of observations for each unique group.

One-Way Frequencies: Setting Options

On the **Options** tab, you can set these options.

Table 8.2 *Frequencies and Percentages*

Role	Description
Frequency table	Specifies whether to create the frequencies table.
Include percentages	Creates a table that contains the cumulative frequencies and percentages and the percentages of total frequencies for each value of the analysis variable.
Include cumulative frequencies and percentages	Creates a table that contains the frequencies and cumulative frequencies for each value of the analysis variable.
Row value order	Specifies the order of the data in the results. You can choose from these options: ■ **Unformatted value**—orders values in ascending order by their unformatted values. This is the default. ■ **Descending frequency**—orders values by descending frequency count. ■ **Formatted value**—orders values in ascending order by their formatted values. ■ **Order of appearance in data set**—orders values according to their order in the input data set.

Table 8.3 *Statistics*

Option Name	Description
Binomial Proportion	Specifies whether to perform an asymptotic test. For binomial proportions, specify a null hypothesis proportion and a confidence level. To compute the exact *p*-values, select **Exact test**.

Option Name	Description
Chi-square Goodness-of-Fit	Specifies whether to perform an asymptotic test. For binomial proportions, specify a null hypothesis proportion and a confidence level. To compute the Monte Carlo estimates of the exact *p*-values instead of directly computing the exact *p*-values, select **Use Monte Carlo estimation**. Monte Carlo estimation can be useful for large problems that require a great amount of time and memory for exact computations but for which asymptotic approximations might be insufficient. Select **Limit computation time** to specify the time limit (in seconds) for the computation of each *p*-value for each crosstabulation table. The default is 300 seconds (or 5 minutes).

Table 8.4 *Plots and Missing Values*

Option Name	Description
Suppress plots	Suppresses the plots from the results.
Include in frequency table	Includes missing values in the frequency tables.
Include in percentages and statistics	Includes the frequencies of missing values in binomial or chi-square tests and in the calculations of percentages.

Example: One-Way Frequencies of Unit Sales

In this example, you want to analyze unit sales for each sales region.

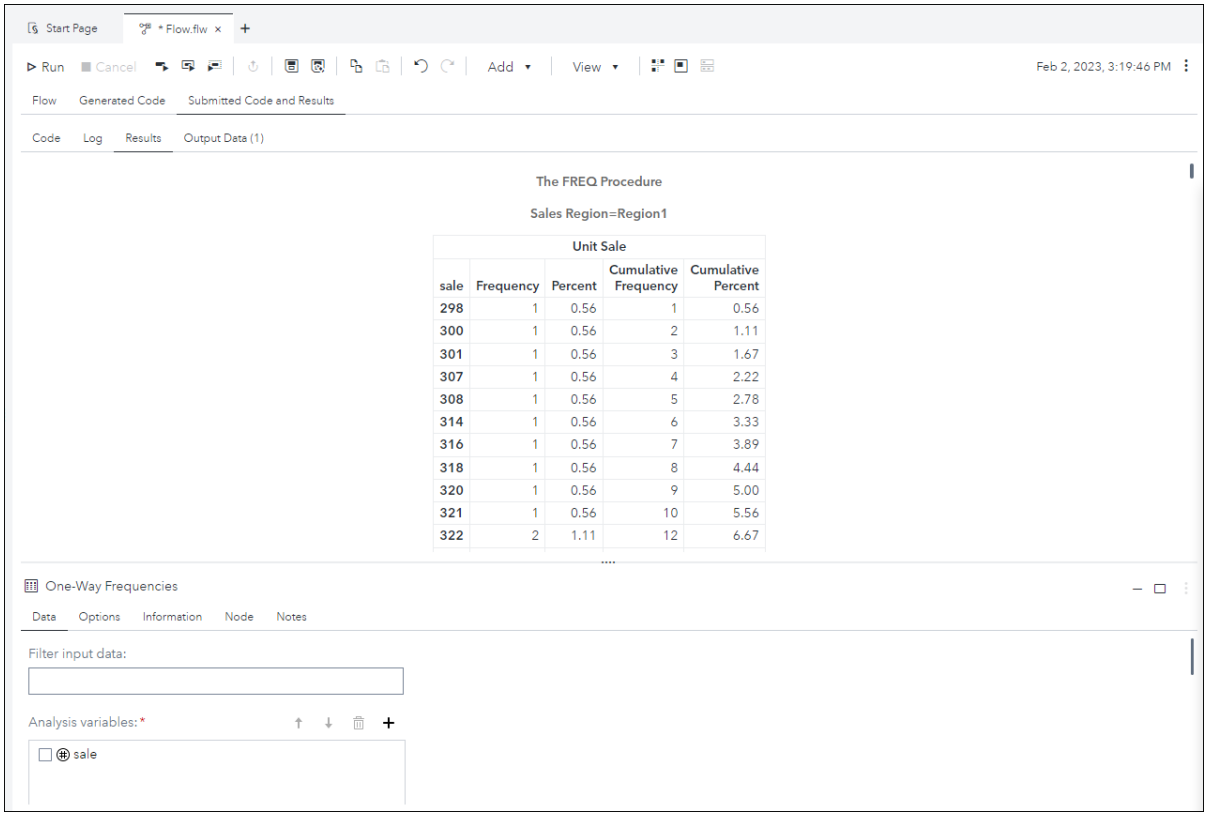
To create this example:

1. In the **Steps** section of the navigation pane, expand the **Statistics** folder and double-click **One-Way Frequencies** to add the step to your flow.
2. Add the Sashelp.PriceData table to your flow. Connect the input port of the One-Way Frequencies node to the data source. For more information, see “Connecting Nodes” on page 12.
3. To set the options for the One-Way Frequencies step, click the One-Way Frequencies node.

4 On the **Data** tab, assign columns to these roles.
   - Assign the **sale** column to the **Analysis variables** role.
   - Expand the **Additional Roles** heading. Assign the **regionName** column to the **Group analysis by** role.

5 Run the flow.

Open the **Submitted Code and Results** tab to view the results.

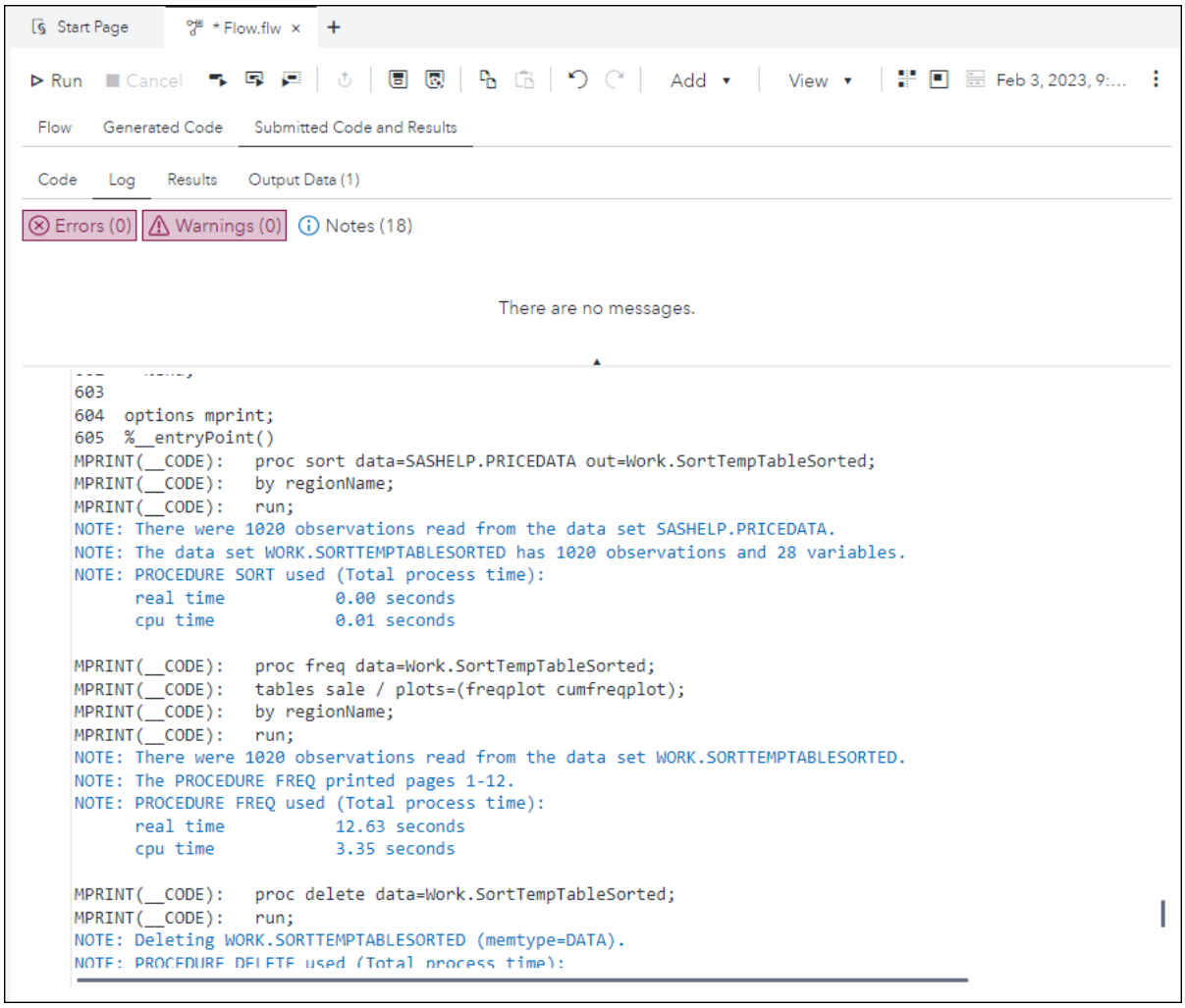


To view the generated SAS code:

1 Open the **Log** tab.

2 Search for MPRINT_CODE.

In this example, here is the generated SAS code.


Start Page
* Flow.flw
Run
Cancel
Add
View
Feb 3, 2023, 9:...
Flow
Generated Code
Submitted Code and Results
Code
Log
Results
Output Data (1)
Errors (0)
Warnings (0)
Notes (18)
There are no messages.
603
604 options mprint;
605 %__entryPoint()
MPRINT(__CODE): proc sort data=SASHELP.PRICEDATA out=Work.SortTempTableSorted;
MPRINT(__CODE): by regionName;
MPRINT(__CODE): run;
NOTE: There were 1020 observations read from the data set SASHELP.PRICEDATA.
NOTE: The data set WORK.SORTTEMPTABLESORTED has 1020 observations and 28 variables.
NOTE: PROCEDURE SORT used (Total process time):
real time 0.00 seconds
cpu time 0.01 seconds
MPRINT(__CODE): proc freq data=Work.SortTempTableSorted;
MPRINT(__CODE): tables sale / plots=(freqplot cumfreqplot);
MPRINT(__CODE): by regionName;
MPRINT(__CODE): run;
NOTE: There were 1020 observations read from the data set WORK.SORTTEMPTABLESORTED.
NOTE: The PROCEDURE FREQ printed pages 1-12.
NOTE: PROCEDURE FREQ used (Total process time):
real time 12.63 seconds
cpu time 3.35 seconds
MPRINT(__CODE): proc delete data=Work.SortTempTableSorted;
MPRINT(__CODE): run;
NOTE: Deleting WORK.SORTTEMPTABLESORTED (memtype=DATA).

9

Optimizing and Running a Flow

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.

The following step combinations can be optimized:

- Query node connected to an Insert Rows node
- Filter Rows node connected to a Sort node

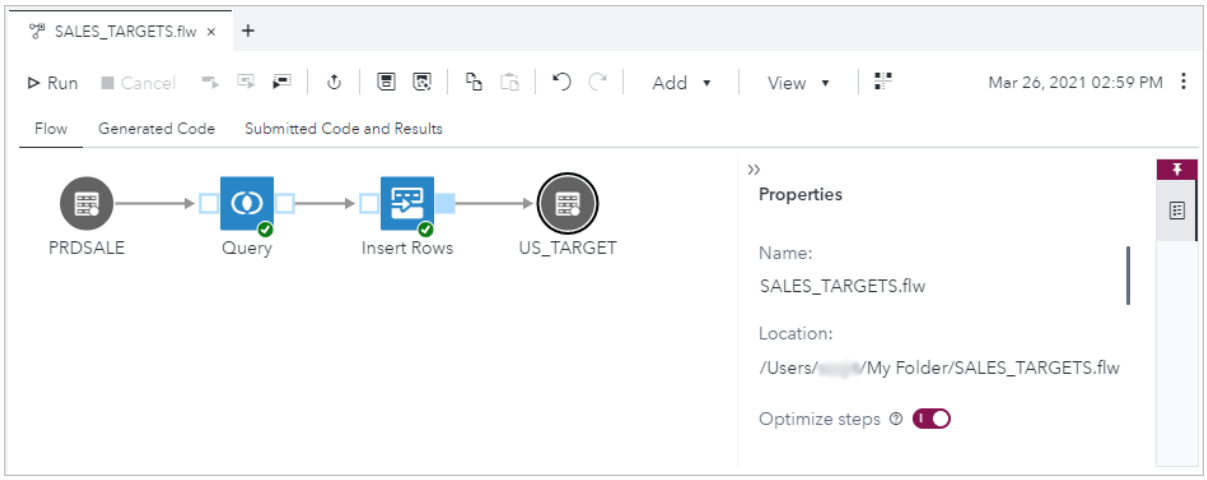
Note: The optimization option works only when the nodes that can be optimized are run at the same time.

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.

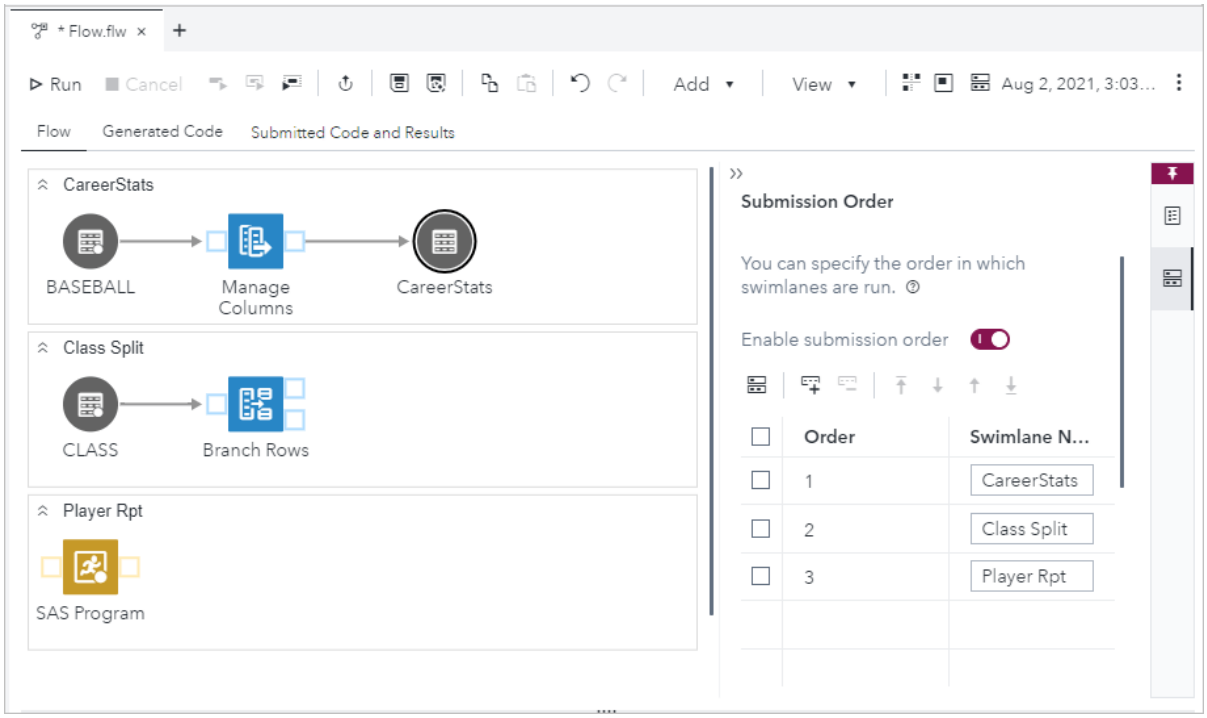


Controlling the Submission Order of a Flow

By default, the order of the nodes on the flow canvas determines the order in which the nodes run. You can control the submission order by grouping connected nodes into swimlanes, and then specifying the order of the swimlanes.

To specify the submission order:

1. On the flow canvas, expand the Submission Order pane by clicking on the right side of the canvas.
2. Click to turn on the **Enable submission order** option. Each group of connected nodes is organized into a separate swimlane.



3 You can perform the following actions in the Submission Order pane:

- To change the order in which the swimlanes are run, use the buttons on the toolbar to move the swimlanes up and down the list. You can also drag swimlanes to the flow canvas.
- To move one or more nodes from one swimlane to another, select the nodes and drag them to the appropriate swimlane. You can also cut, copy, and paste nodes between swimlanes.
- To add a new swimlane to the flow, click on the Submission Order toolbar.
- To delete a swimlane and the associated nodes, select one or more swimlanes from the list of swimlanes and click on the Submission Order toolbar.
- To rename a swimlane, enter the new name in the Swimlane name column. The name is updated on the flow canvas.
- To refresh the arrangement of the nodes within swimlanes and remove empty swimlanes, click on the toolbar.

Note: When you disable the submission order feature, any changes that you have made to swimlane names and order are not saved.

Running a Flow

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

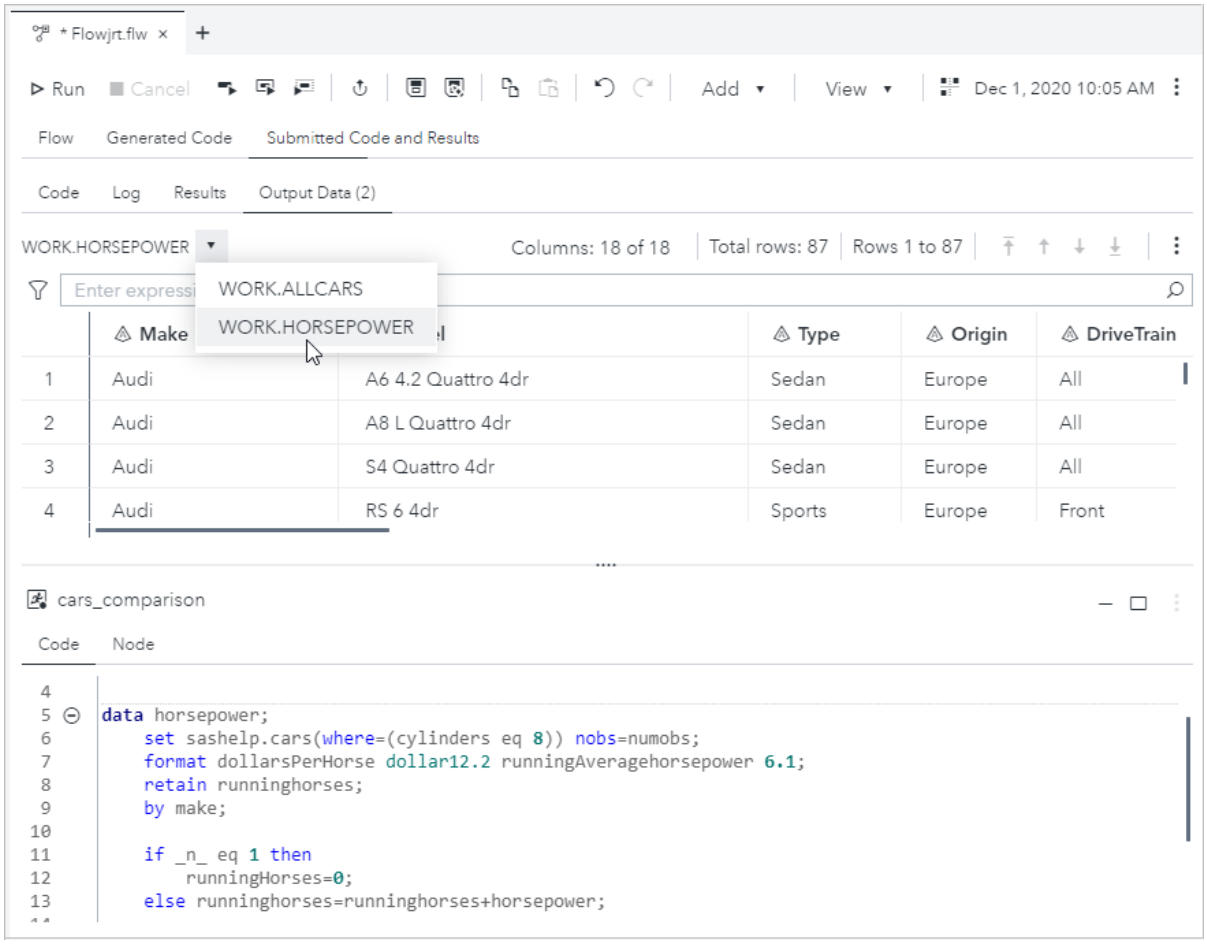
You can also run a subset of the flow by using the following buttons on the flow toolbar:

- - 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 **Submitted Code and Results** tabs on the flow canvas. The **Submitted Code and Results** 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 last submitted code** or **Go to last submitted 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" in *SAS Studio: User's Guide*.

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

10

Creating a Job from a Flow

Creating a Job from a Flow

To create a job from a saved flow, open the Explorer pane. Navigate to the location of the saved FLW file. Right-click the name and select **Create job**. The new job opens as a job definition in the workspace. For more information, see "Create a New Job Definition" in *SAS Studio Developer's Guide: Working with Jobs*.

You can also schedule a job from the Explorer pane. Right-click the file name of the flow and select **Schedule a job**. For more information, see "Schedule a Job" in *SAS Studio Developer's Guide: Working with Jobs*.

11

SAS Information Catalog and SAS Lineage Viewer Integration

SAS Information Catalog and SAS Lineage Viewer Integration

SAS Studio flows are automatically indexed in SAS Information Catalog and SAS Lineage Viewer. SAS Information Catalog enables you to capture and enrich metadata for files, tables, and other information assets. This metadata is stored in a catalog. You can search the catalog to find SAS Studio flows and other assets that you need to meet your business goals.

To access SAS Information Catalog, select **Discover Information Assets** from the Applications menu in the top left of the application window. For more information, see SAS Information Catalog: User's Guide.

SAS Lineage Viewer enables you to better understand the relationships between objects in your applications on the SAS Viya platform. These objects include SAS Studio flows, data, transformation processes, reports, and visualizations.

To access SAS Lineage Viewer, select **Explore Lineage** from the Applications menu in the top left of the application window. For more information, see SAS Lineage: User's Guide.