Power Washing Your Data: Using SAS Data Quality Steps in SAS Studio Flows

Exercise Description

Update your data cleaning process with point-and-click data quality steps in SAS Studio Flows! In this hands-on workshop, you'll learn how to build a flow that deduplicates, parses, and standardizes your data with SAS Data Quality steps and code snippets.
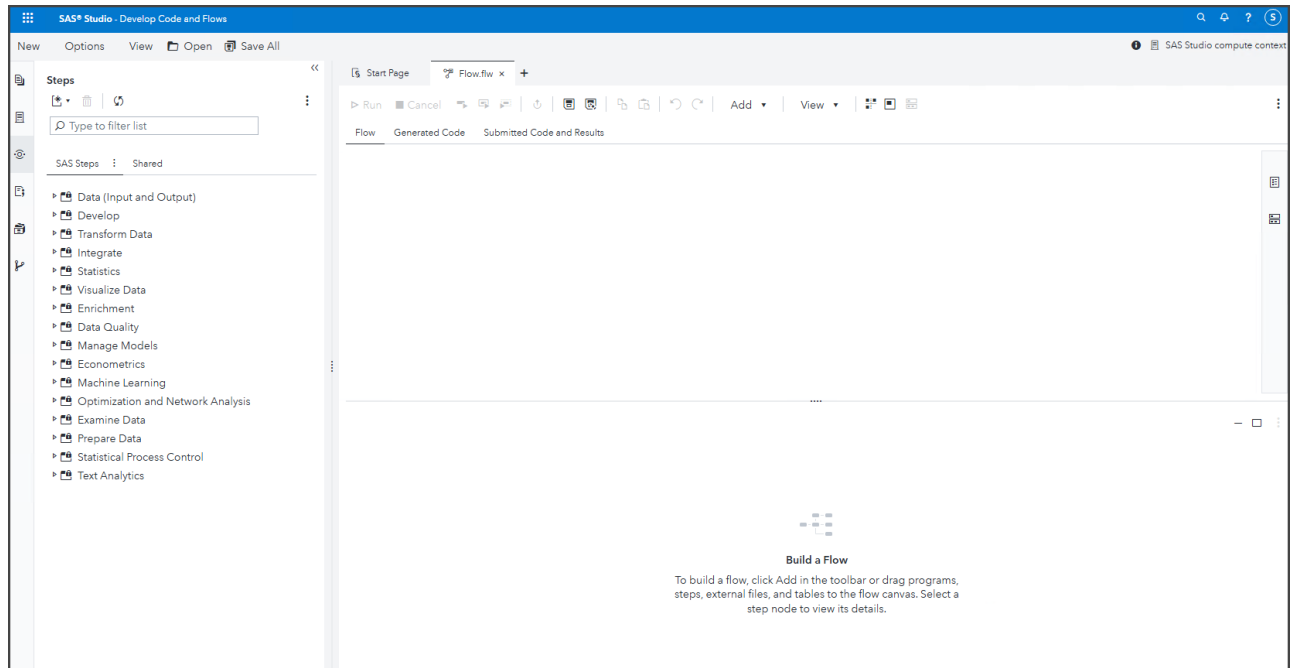
Log in to SAS Viya

Open a new window in the *Google Chrome* browser and select the **SAS Viya** bookmark.

- ID: **student**
- Password: **Metadata0**

Select **No** when prompted about accepting *Admin* privileges.

Create a SAS Studio Flow

1. Select ⊞ ➔ **Develop Code and Flows** to open *SAS Studio*.

2. Select **New ➔ Flow**.

3. Select 🔘 to view the **Steps** pane.

4. Expand the *Data (Input and Output)* section of the *Steps* pane. Double-click the **Table** step to add it to the flow canvas.

5. Select the **Table** node on the flow canvas. In the **Table Properties** section, select the following:

   - Library: **DQWS**
   - Table name: **CUSTOMERS**
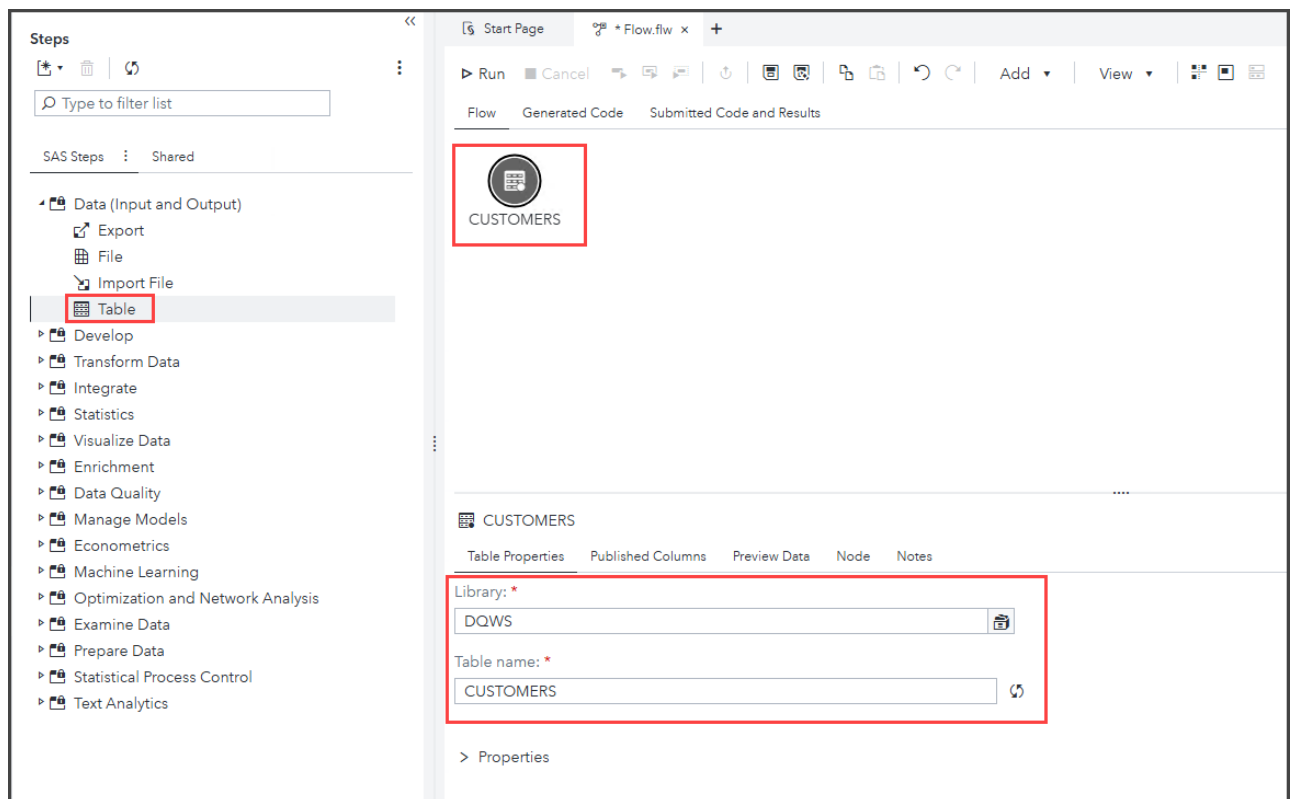
6. Click **Preview Data** to view a subset of rows from the table.

✎ If needed, click ⟪ to hide the Steps pane, or click ☐ to maximize the preview.

CUSTOMERS					Table rows: 60	Columns: 9 of 9	Rows 1 to 60		

	⊕ CustomerID	⚐ Name	⚐ Gender	⚐ Address1	🗓 Birthday	⚐ Occupation	⚐ Company	⚐ Email	⚐ Phone
1	.	AGNES R. YATES		4334 CHESTNUT ST,POLK CITY,FLORID…	.				
2	.	JAMES ATTWOOD		3238 HERITAGE RD,SAN JOAQUIN,CAL…	.				
3	100000014	Agnes R. Yates		4334 Chestnut Street,Polk City,FL,33868…	.			AgnesRYates@poo…	727-460-5745
4	100000012	Anna R. Christian		3323 Morgan Street,Fort Walton beach,…	.			AnnaChristi@dodg…	850-467-1006
5	100000010	Attwood, James	M	3238 Heritage Road,San Joaquin,CA,93…	1988-06-29	Information sy…	Thom McAn S…		
6	.	CALVIN WENZEL		1007 THOMPSON DR,DUBLIN,CALIFO…	.				
7	.	CATHERINE M. THOMAS		3005 BROOKSIDE DRIVE,GUIN,ALABA…	.				
8	.	CHRISTIAN, ANNA		3323 MORGAN ST,FORT WALTON BEA…	.				
9	.	CURRY, KRISTI R.		399 MASONIC HILL ROAD,LITTLE ROC…	.				
10	100000007	Calvin B. Wenzel		1007 Thompson drive,Dublin,CA,94568…	.			CalWenzel@mailin…	510-453-8297
11	100000018	Carolan, Kris J.	M	3935 Lakeland Park Driv,Duluth,GA,300…	1986-09-23	Real property …	Dun Rite Lawn…		
12	100000001	Catherine Thomas		3005 Brookside Drive,Guin,AL,35563,US	.			KatherineMThoma…	205-412-9602
13	100000013	Cheeks, Linda	F	3385 Rinehart Road,Miami,FL,33128,US	1999-09-30	Delivery servi…	Home Centers		
14	100000012	Christian, Anna R.	F	3323 Morgan Street,Fort Walton beach,…	1979-07-30	Mixing and bl…	Sambo's		
15	100000004	Curry, Kristi	F	399 Masonic Hill Road,Little Rock,AR,72…	1996-05-13	Industrial pro…	Giant		
16	.	DICK REYES		4819 KERRY WAY,LOS ANGELES,CALIF…	.				

CUSTOMERS has **60** rows and **9** columns. Note that:

- Each customer appears three times.
- All rows have *Name* and *Address* values.
- Most rows have a non-missing *CustomerID* value.
- Customer *Name* values differ in each appearance (casing, name order, inclusion of middle initial or use of nickname).
- Rows with non-missing *Gender* values also have non-missing *CustomerID*, *Birthday*, *Occupation*, and *Company* values.
- Rows with non-missing *Email* values also have non-missing *CustomerID* and *Phone* values.

We would like to deduplicate this data by combining unique values into one row per customer. First, we'll generate match codes to determine which rows represent the same customers.

Generate Match Codes

1. Restore your screen view if needed.

2. Expand the *Data Quality* section of the *Steps* pane. Double-click the **Match Codes** step to add it to the flow canvas.

3. Drag the *Match Codes* step to the right of *CUSTOMERS*. Use your mouse to draw an arrow connecting *Customers* to the *Match Codes* step.

4. (Optional) Click [icon] on the flow toolbar to arrange the nodes.
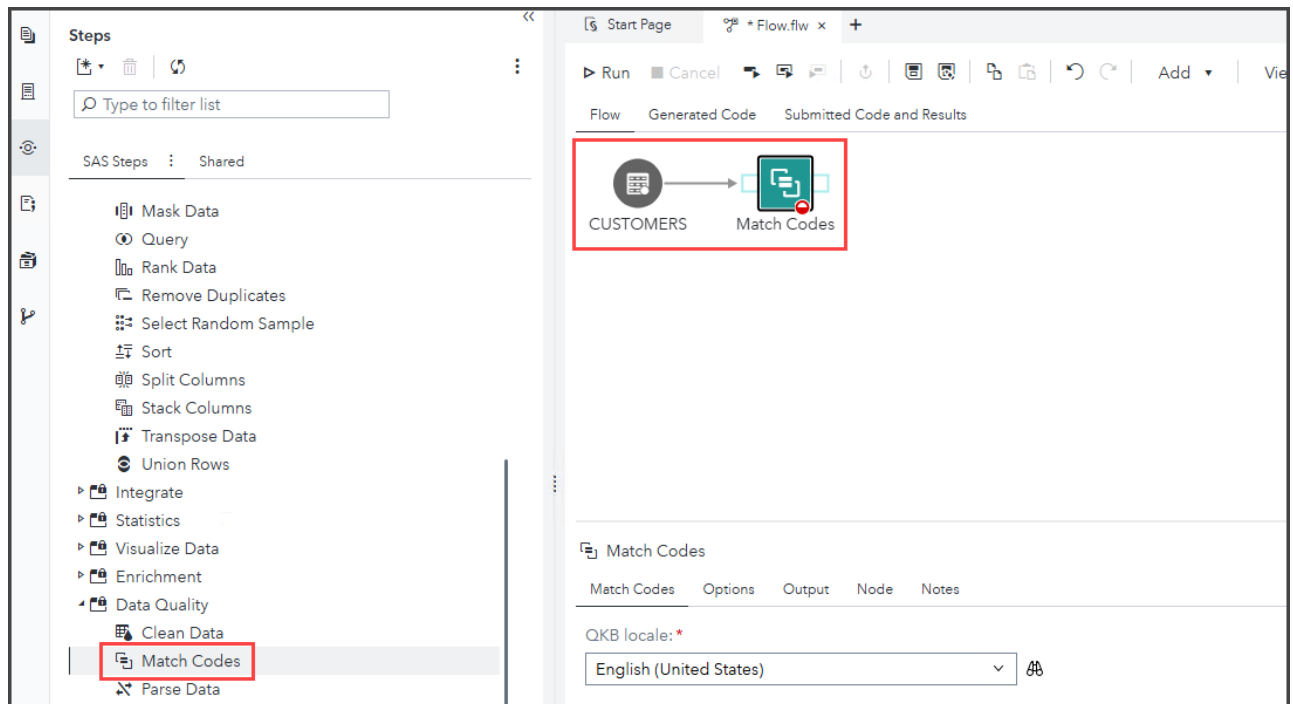
5. In the *Match Codes* tab for the *Match Codes* step, ensure that the default QKB locale (**English (United States)**) is selected.

6. Configure the *Match Codes* step as follows:

    ○ Match column: **Address1**
    ○ New match column name: **[Leave blank]**
    ○ Definition: **Address**
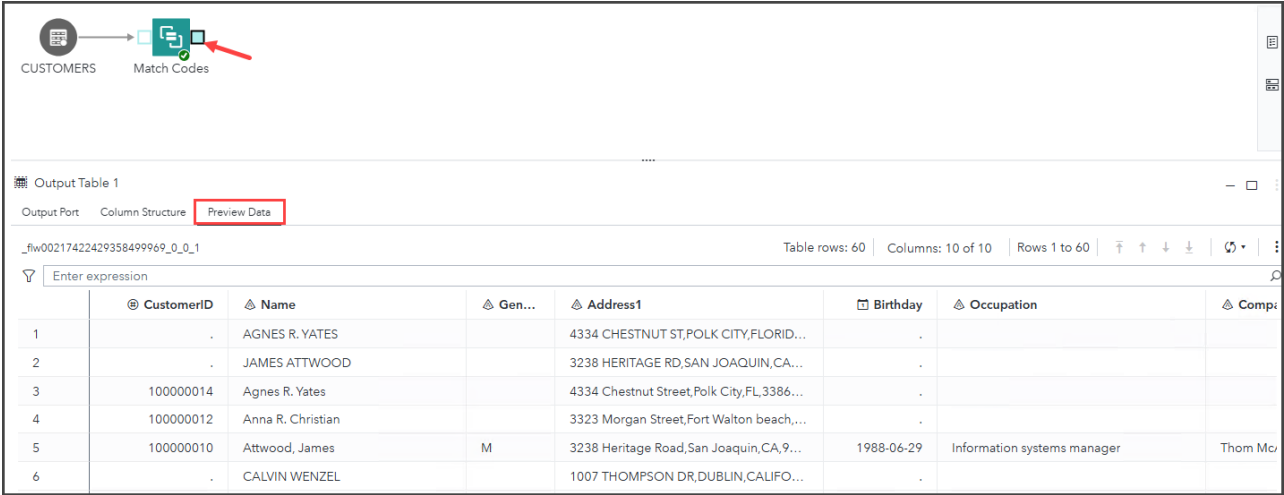   ○ Sensitivity: **85**

7. Select to save the flow.

8. Navigate to **SAS Content ➜ Public**. Save the file as **DQ_HOW.flw**.

9. Select on the flow toolbar to run the flow.

10. After the flow has run successfully, click the shaded **output port** on *Match Codes*.

11. Click **Preview Data** to review the output.

12. Click [icon] in the *Preview Data* window and select **Column Selection**.

13. In the *Column Selection* window, deselect all columns, then individually select **Name**, **Address1**, and **Address1_MC85**, then click **OK**.

14. Review the output on the *Preview Data* tab.

Notice that rows with equivalent *Address1* values generated the same match codes regardless of *Address1* format. We can use these match codes to combine customer data into one surviving record.

Remove Duplicates with Survivorship Snippet

1. Add the **SAS Program** step from the *Develop* section of the *Steps* pane to the flow canvas and connect it to the *Match Nodes* step.

2. **Arrange** the nodes and **save** the flow.

3. Select to open the **Snippets** pane.

4. Expand **Standard ➜ Data Quality** and double-click **Survivorship** to open the *Survivorship* snippet in a new tab.
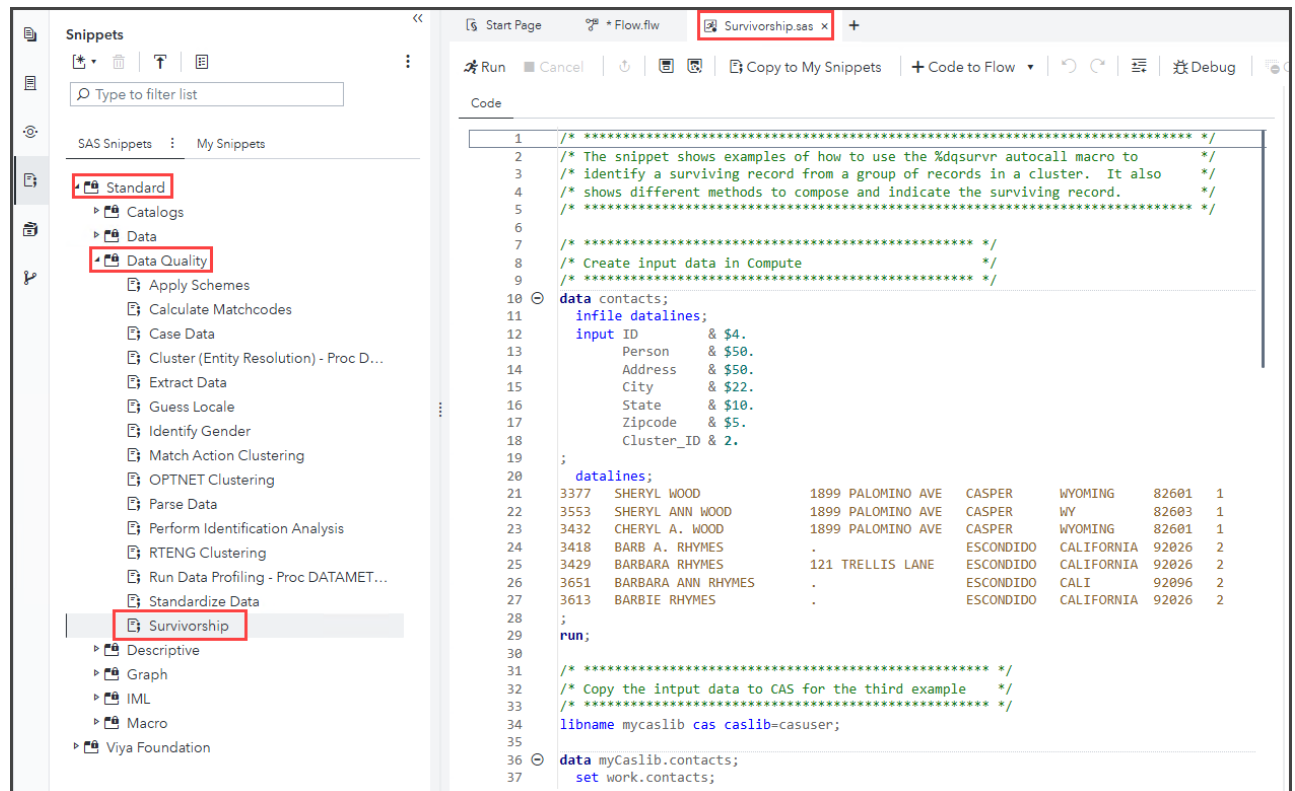
Per the snippet documentation: *The snippet shows examples of how to use the %dqsurvr autocall macro to identify a surviving record from a group of records in a cluster. It also shows different methods to compose and indicate the surviving record.*

5. Copy the first example that calls the %dqsurvr macro **(lines 52 - 59)** and paste it into the SAS Program node's *Code* tab.

▶ Details

Click to view or copy the required section of code.

```
%dqsurvr (inTable=contacts,
          outTable=work.contacts_out,
          clusterColumn=cluster_id,
          rowRule1=(max,ID),
          firstColumnRule1=(highocc,State),
          firstColumnRuleAppliedCols=(Zipcode),
          secondColumnRule1=(not_missing, Address),
          keepDuplicates=0);
```

```
SAS Program

  Code    Node    Notes

1    %dqsurvr (inTable=contacts,
2                outTable=work.contacts_out,
3                clusterColumn=cluster_id,
4                rowRule1=(max,ID),
5                firstColumnRule1=(highocc,State),
6                firstColumnRuleAppliedCols=(Zipcode),
7                secondColumnRule1=(not_missing, Address),
8                keepDuplicates=0);
9
```

6. Edit the macro parameters as follows:

  - inTable=**&_input1**
  - outTable=**&_output1**
  - clusterColumn=**Address1_MC85**
  - rowRule1=**(longest, Name)**

    > ✎ The surviving record will be chosen by the longest **Name** value, because this should be the fullest version of the customer's name.

  - firstColumnRule1=**(not_missing, Gender)**

    > ✎ The surviving record should store the first non-missing *Gender* value from the cluster records.

  - firstColumnRuleAppliedCols=**(Birthday, Occupation, Company)**

    > ✎ The surviving record should store the *Birthday*, *Occupation*, and *Company* values from the row satisfying *firstColumnRule1*.

  - secondColumnRule1=**(not_missing, Email)**

    > ✎ The surviving record should store the first non-missing *Email* value from the cluster records.

  - **Add secondColumnRuleAppliedCols=(CustomerID, Phone),** (including comma after option)

    > ✎ The surviving record should store the *CustomerID* and *Phone* values from the row satisfying *secondColumnRule1*.

  - **Add a comma after** keepDuplicates=**0**
  - **Add generateDistinctSurvivor=1** after keepDuplicates=**0**

▶ Details

Click to view or copy the edited macro call.

```
%dqsurvr (inTable=&_input1,
        outTable=&_output1,
        clusterColumn=Address1_MC85,
        rowRule1=(longest,Name),
        firstColumnRule1=(not_missing,Gender),
        firstColumnRuleAppliedCols=(Birthday,Occupation,Company),
        secondColumnRule1=(not_missing,Email),
        secondColumnRuleAppliedCols=(CustomerID,Phone),
        keepDuplicates=0,
        generateDistinctSurvivor=1);
```


7. On the **Node** tab, re-name the node to **Surviving Records**.
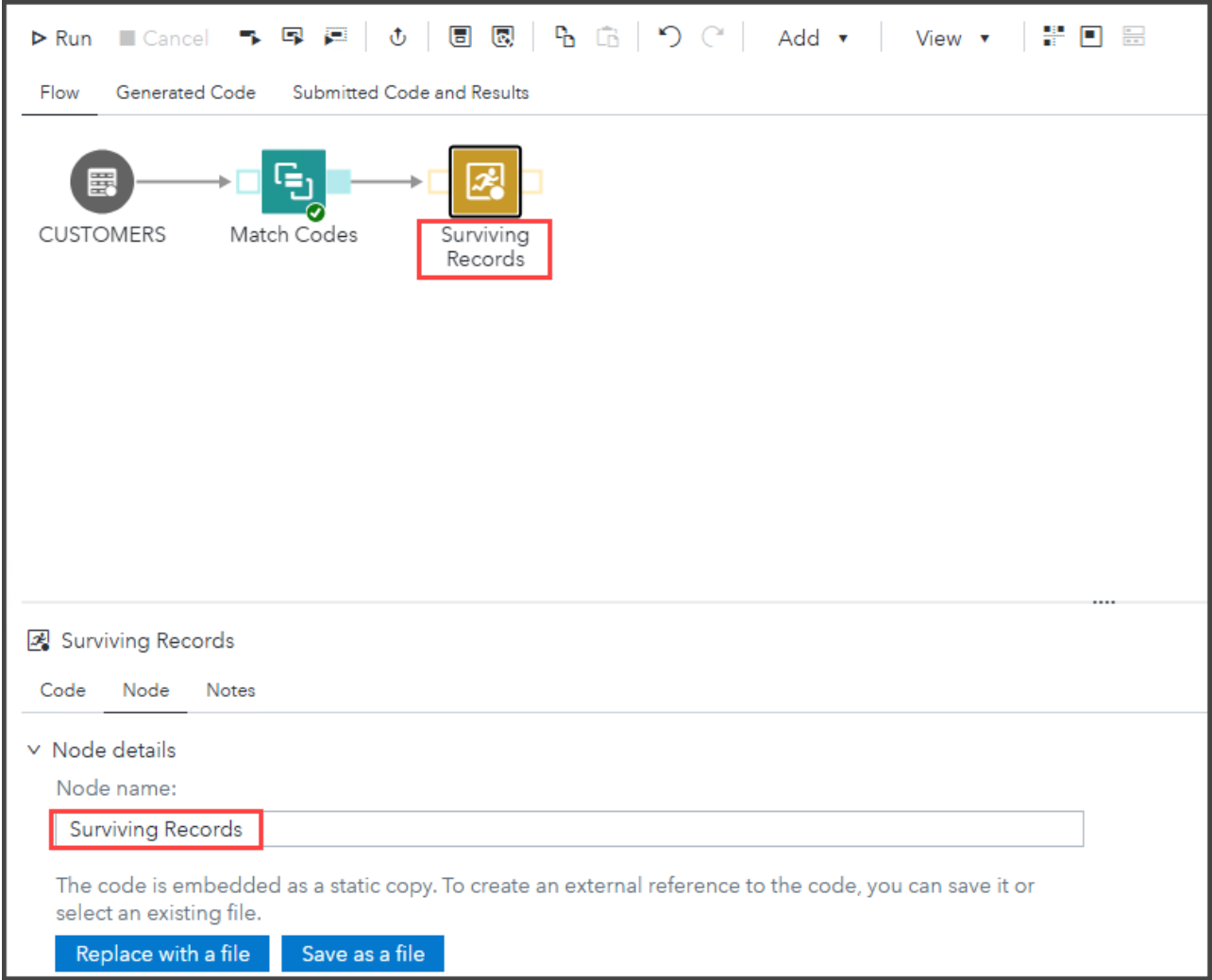
8. **Save** the changes to the flow.

9. **Right-click** *Surviving Records* and select **Run node**. Select the **output port** of *Surviving Records* and **preview** the results.

The output table has **20** rows and **5** columns. Duplicate records were accurately condensed to store all customer data in one record. Next, we'll parse *Address1* and store individual tokens like Street, City, and State in separate columns.

Parse Address Data

1. Select [icon] to return to the **Steps** pane.

2. Add the **Parse Data** step from the *Data Quality* section of the *Steps* pane to the flow canvas and connect it to the *SAS Program (Surviving Records)* step.

3. **Arrange** the nodes and **save** the flow.
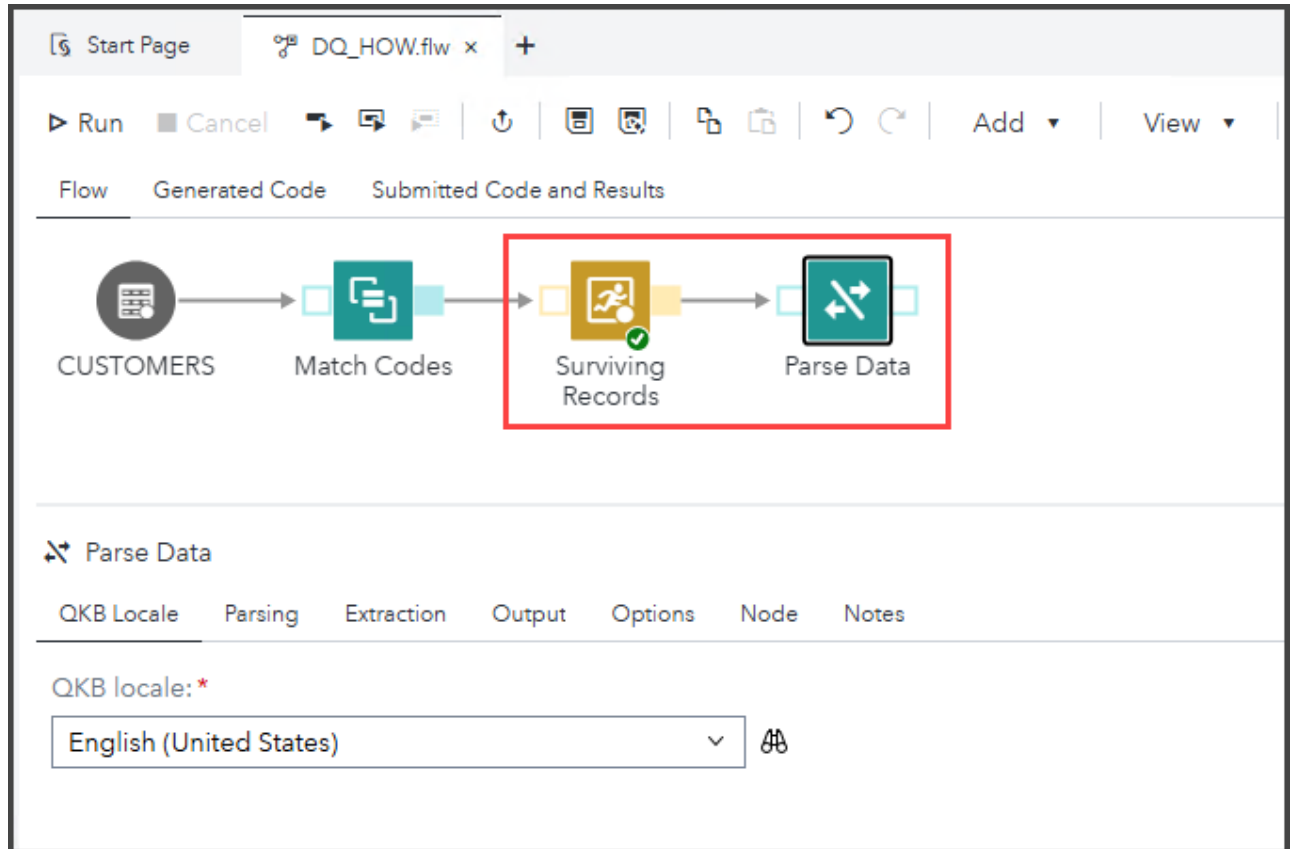
4. In the *QKB Locale* tab for the *Parse Data* step, ensure that the default QKB locale (**English (United States)**) is selected.
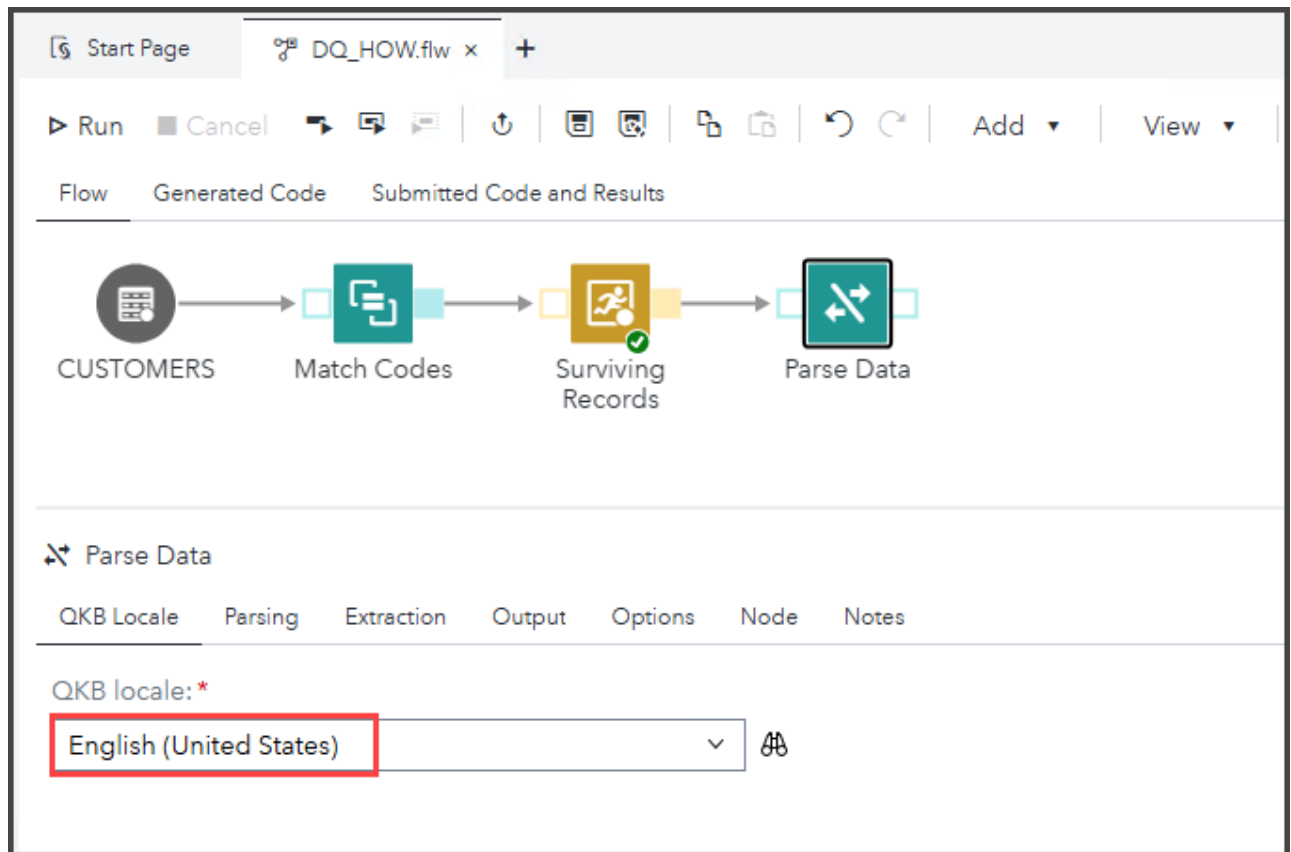
5. On the *Parsing* tab, check the box to **Enable parsing**.
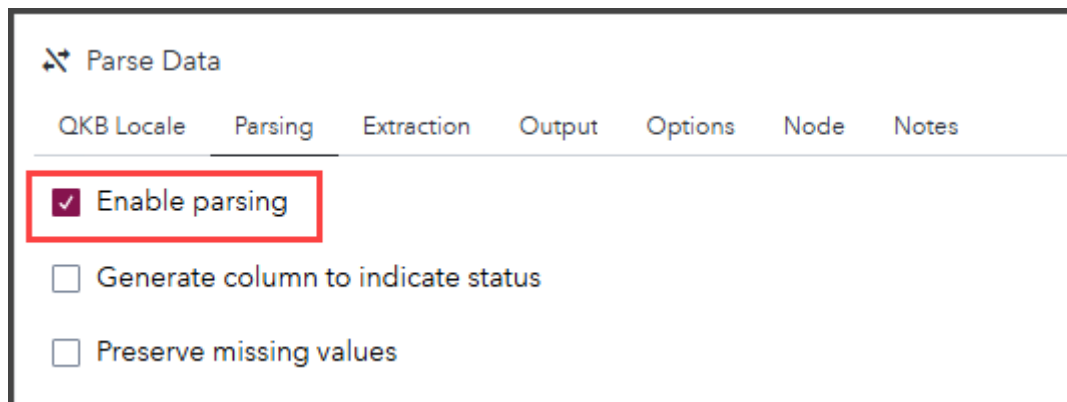
6. Configure the *Parse Data* step as follows:

- Select column: **Address1**
- Definition: **Address (Full)**
- Select tokens: **City**, **Country**, **Postal Code**, **State/Province**, **Street**
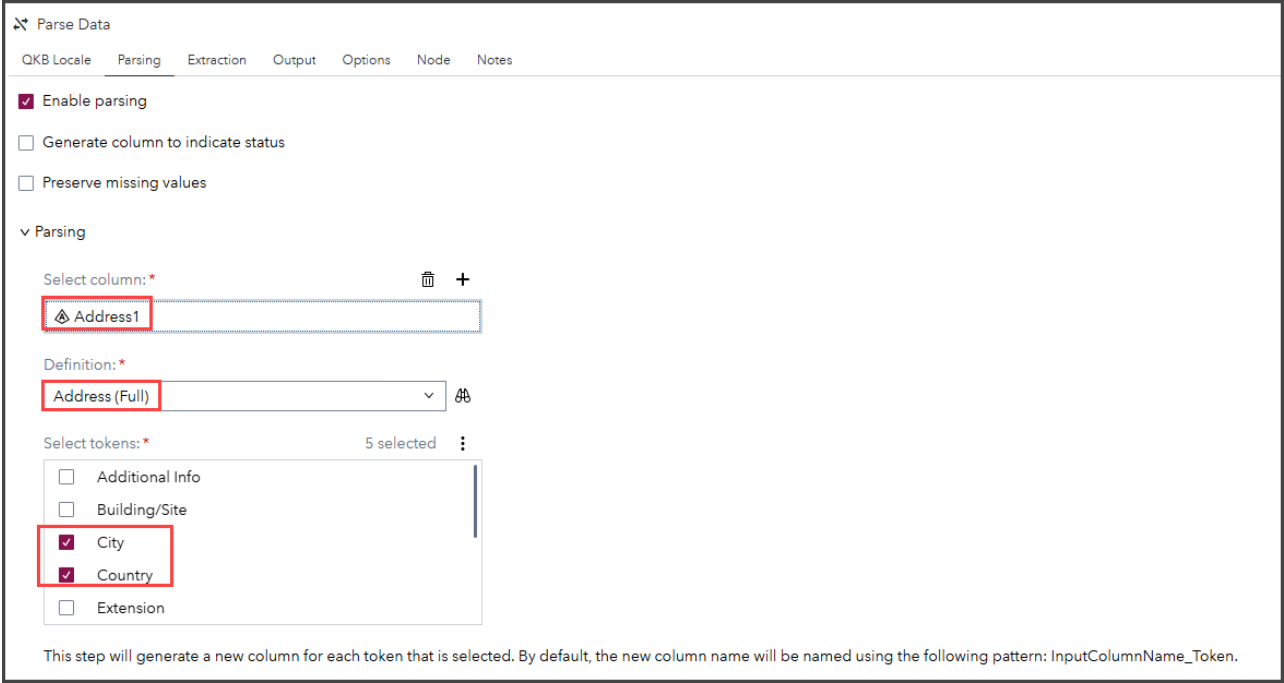
7. **Save** the changes to the flow.

8. **Right-click** *Parse Data* and select **Run node**. Select the **output port** of *Parse Data* and **preview** the results.

> ✎ Use the *Column Selection* option or scroll all the way to the right in the *Preview Data* window to see new columns.

The output table has **5** new columns storing the **City**, **Country**, **Postal Code**, **State/Province**, and **Street** from each *Address1* value. Note that:

- ○ **Address1_CITY** varies in casing. Some city names are written in uppercase, while others are in proper case.
- ○ **Address1_STATE_PROVINCE** varies in format. Some state names are written in full, while others are written in abbreviations.
- ○ **Address1_STREET** varies in casing and format. Specifically, some street addresses use full road names, while others use abbreviations.
- ○ **Address1_COUNTRY** uses a consistent country abbreviation standard, but we'd like to see the full country name instead. We'll correct these issues and others with the Clean Data step.

Standardize Data

1. Add the **Clean Data** step from the *Data Quality* section of the *Steps* pane to the flow canvas and connect it to the *Parse Data* step.

2. **Arrange** the nodes and **save** the flow.
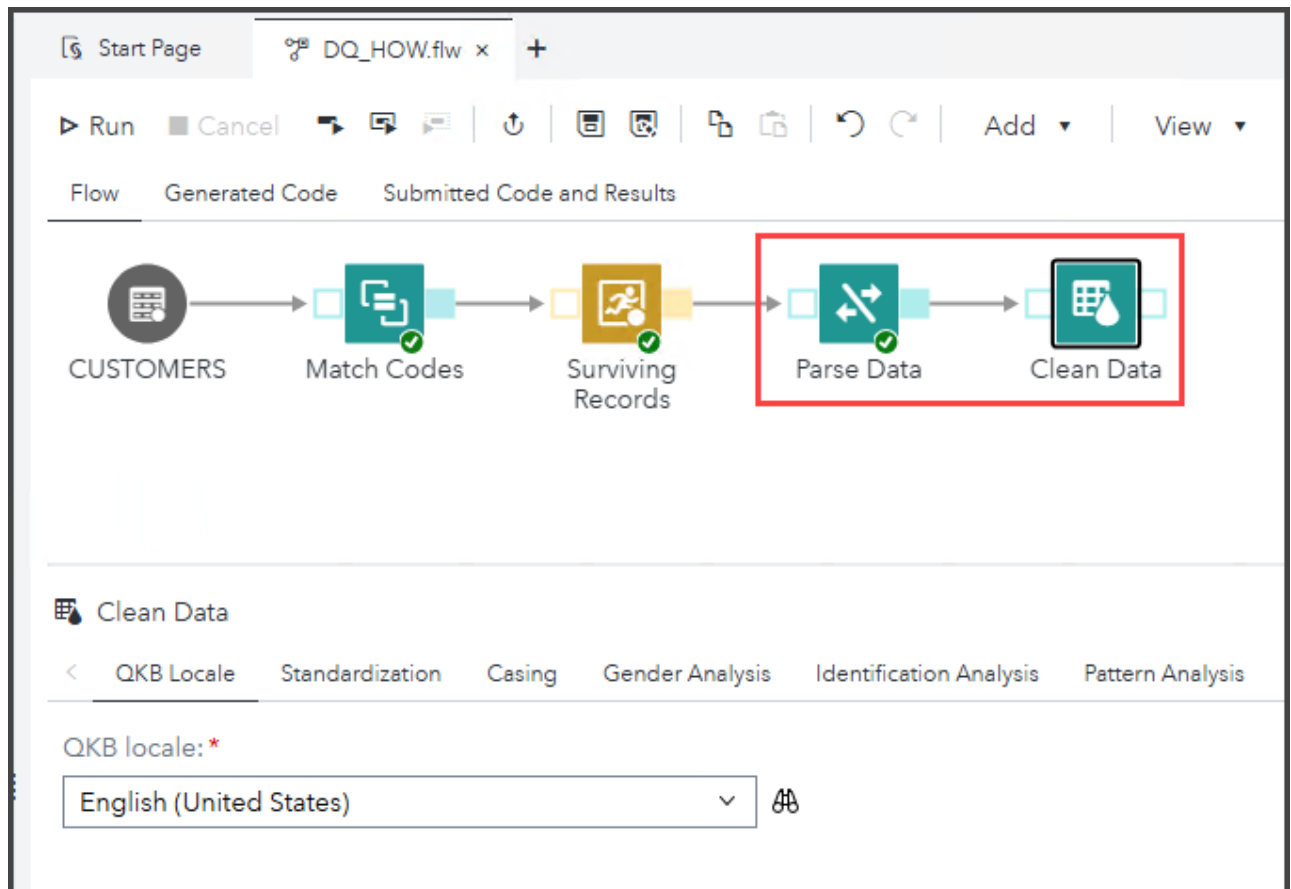
3. In the *QKB Locale* tab for the *Clean Data* step, ensure that the default QKB locale (**English (United States)**) is selected.

4. On the *Standardization* tab, check the box to **Enable standardization**.

5. Configure the *Clean Data* step as follows:

   ○ Select column: **Name**
   ○ Definition: **Name**
   ○ Column options: **Create new column**
   ○ New column: **[Leave blank]**

6. Under the *Additional Standardize* heading, repeat this process to standardize columns as specified below. **For all *Additional Standardize* columns, select *Create new column* and leave the new column name *blank*.**

Select Column	Definition
Address1_CITY	City
Address1_STATE_PROVINCE	State/Province (Full Name)
Address1_STREET	Address
Address1_COUNTRY	Country

7. **Save** the changes to the flow.

8. **Right-click** *Clean Data* and select **Run node**. Select the **output port** of *Clean Data* and **preview** the results.

> ✎ If the *Clean Data* step fails with *ERROR: Key not found* and *ERROR: Error in the FILENAME statement*, simply run the step again. This is a known bug which is resolved by executing again.

9. (Optional) use the *Column selection* option to limit the column view in the *Preview Data* window.

Standardized names are proper cased and written in **First MI Last** format.

Standardized city, state, and country names are proper cased and written in full instead of abbreviated.

Standardized street names are proper cased and road type names (Street, Drive, Avenue, etc) are shortened.

We'll finish by creating an output table to store the final results.

Create Output Table

1. Add the **Query** step (from the *Transform Data* section of the *Steps* pane) to the flow canvas and connect it to the *Clean Data* step.

2. Right-click the *Query* step output port and select **Add a table**.

3. **Arrange** the nodes and **save** the flow.

4. Select the **Table** node on the flow canvas. In the **Table Properties** section, type the following table attributes:

   ○ Library: **DQWS**
   ○ Table name: **CUSTOMERS_CLEAN**
   ○ Select **Create a physical table** (default selection)

5. Select the **Query** node on the flow canvas. In the *Options* tab, expand **t1 (Clean Data)** to view all source columns.

6. Select the following columns and update their attributes on the *Select* tab:

Source	Name	Label
CustomerID	CustomerID	**Customer ID**
Name_STND	**Name**	
Address1_STREET_STND	**Street**	

Source	Name	Label
Address1_CITY_STND	**City**	
Address1_STATE_PROVINCE_STND	**State**	
Address1_POSTALCODE	**PostalCode**	**Postal Code**
Address1_COUNTRY_STND	**Country**	
Gender	Gender	
Birthday	Birthday	
Occupation	Occupation	
Company	Company	
Email	Email	
Phone	Phone	

> ✏ Updates to column attributes are **bolded**.

7. Click the *Sort* tab.

8. Double-click *CustomerID* to add it to the *Sort* tab. Keep the default sort order selection (**Ascending**).

9. **Save** the changes to the flow.

10. **Right-click** *Query* and select **Run node**. Select the output table **CUSTOMERS_CLEAN** and **preview** the results.

> ✏ If needed, click [«] to hide the Steps pane, or click [□] to maximize the preview.

11. **Close** the *DQ_HOW.flw* file tab after reviewing the results.

Exercise Completed

You have completed the exercise on building a flow with SAS Data Quality!

THANK YOU FOR ATTENDING THIS WORKSHOP!