Chapter 9: Creating Prep Flows in Various Business Scenarios

So far in this book, we've learned a wide variety of capabilities offered by Tableau Prep. In this chapter, we're going to cover creating an end-to-end flow in Tableau Prep. Each recipe will allow you to prepare for a realistic business scenario in which you may use Tableau Prep. In the first recipe, we'll build a data flow that prepares transaction data for a chain of stores, for the purpose of downstream analysis. During this process, we'll prepare the data and perform cleanup actions so that the downstream analysis can leverage a comprehensive and clean dataset. In the second recipe, we'll use Tableau Prep to answer questions. That is, we'll transform the data to answer a specific business question. Both of these recipes mimic real-world scenarios that you are likely to encounter, no matter the industry you work in.

In this chapter, we will cover the following recipes:

- Creating a flow for transaction analytics
- Creating a call center flow for instant analysis

Technical requirements

To follow along with the recipes in this chapter, you will require Tableau Prep Builder.

The recipes in this chapter use sample data files that you can download from the book's GitHub repository at https://github.com/PacktPublishing/Tableau-Prep-Cookbook.

Creating a flow for transaction analytics

In this recipe, we'll create a data pipeline, or flow, for analytics. In this scenario, we'll assume that we are an analyst for a fictive department store with multiple physical stores, as well as an online store front. We will be presented with multiple data sources that need to be combined, cleaned, and transformed so that we can output a clean and reliable dataset of all transactions that occurred in the first six months of 2020. This is a common scenario in most industries and is the perfect use case for Tableau Prep.

Getting ready

To follow along with this recipe, download the **Sample Files 9.1** folder from this book's GitHub repository. In here you'll find various data files. Several of these files originate from disparate systems and we'll need to employ Tableau Prep to provide a single, holistic output of all transactions.

The contents of the files are as follows:

- Files starting with **OnlineSales** contain sales information for transactions made through the company website. There is one file per calendar month, and so we must combine six files to get the full dataset we need for the first six months of 2020.
- **STORE_SALES_EXPORT.xlsx** contains sales data from physical stores. The stores sell the same products as the online storefront. However, the data format is different as the stores use a different point-of-sale system. This data export contains all store sales for the six-month period we need, from January to June 2020.
- **Products.csv** contains descriptive product information, such as the product name and category. We will need to join this to the sales data so that the new dataset is easier to understand, as the sales data only includes product IDs.
- **ShippingData.hyper** is a Tableau Hyper extract prepared by our analyst colleague who works in the shipping department. The data contains product shipping information for those products that were sold online. The company does not provide a delivery service for products bought in their physical stores.
- **CustomerList.csv** contains our customer information for those customers who have created an account with the company. Let's assume that creating a customer account for online transactions is mandatory. However, in-store transactions only have a customer ID if the customer uses their optional loyalty card.
- **returns_h1_2020.csv** contains product return information.

In this recipe, we're going to combine all of these datasets using a number of techniques we've learned in this book. The output of our flow will be a comprehensive dataset that can be easily understood and used for downstream analysis purposes.

How to do it...

Start by opening up Tableau Prep and connect to the **OnlineSales_2020_01.csv** file from the **Sample Files 9.1** folder in Tableau Prep. Then, perform the following steps:

1. This dataset contains data for a single month. Specifically, the month of January, as indicated by the last two numbers in the filename. The format of all files starting with **OnlineSales** are the same, and so we can combine these files using the **UNION** functionality with the input step. To do this, select the **Multiple Files** tab in the **Input** settings and select **Wildcard union**. Then, set the matching pattern to **OnlineSales***. This will instruct Tableau Prep to union all files starting with **OnlineSales**. Make sure to click **Applied** to save your settings:

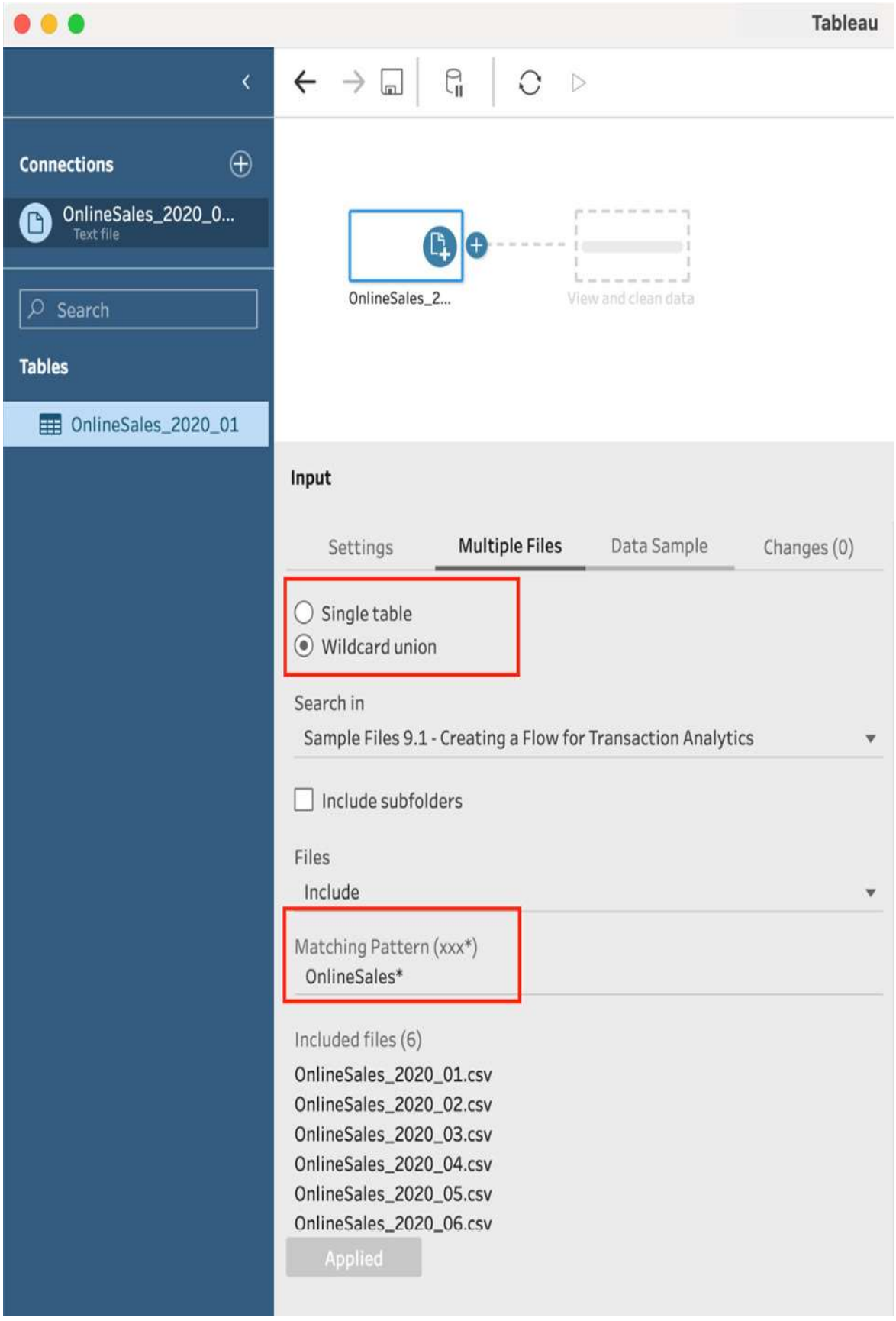
Tableau
Connections
OnlineSales_2020_0...
Text file
Search
Tables
OnlineSales_2020_01
OnlineSales_2...
View and clean data
Input
Settings
Multiple Files
Data Sample
Changes (0)
Single table
Wildcard union
Search in
Sample Files 9.1 - Creating a Flow for Transaction Analytics
Include subfolders
Files
Include
Matching Pattern (xxx*)
OnlineSales*
Included files (6)
OnlineSales_2020_01.csv
OnlineSales_2020_02.csv
OnlineSales_2020_03.csv
OnlineSales_2020_04.csv
OnlineSales_2020_05.csv
OnlineSales_2020_06.csv
Applied

Figure 9.1 – Selecting Wildcard union and specifying the matching pattern

2. As a result of our union action, Tableau Prep has automatically added the **File Paths** field, to indicate where each row of data originated. As we won't require this information for any type of analysis, we can remove it here simply by unchecking the box in the field list:

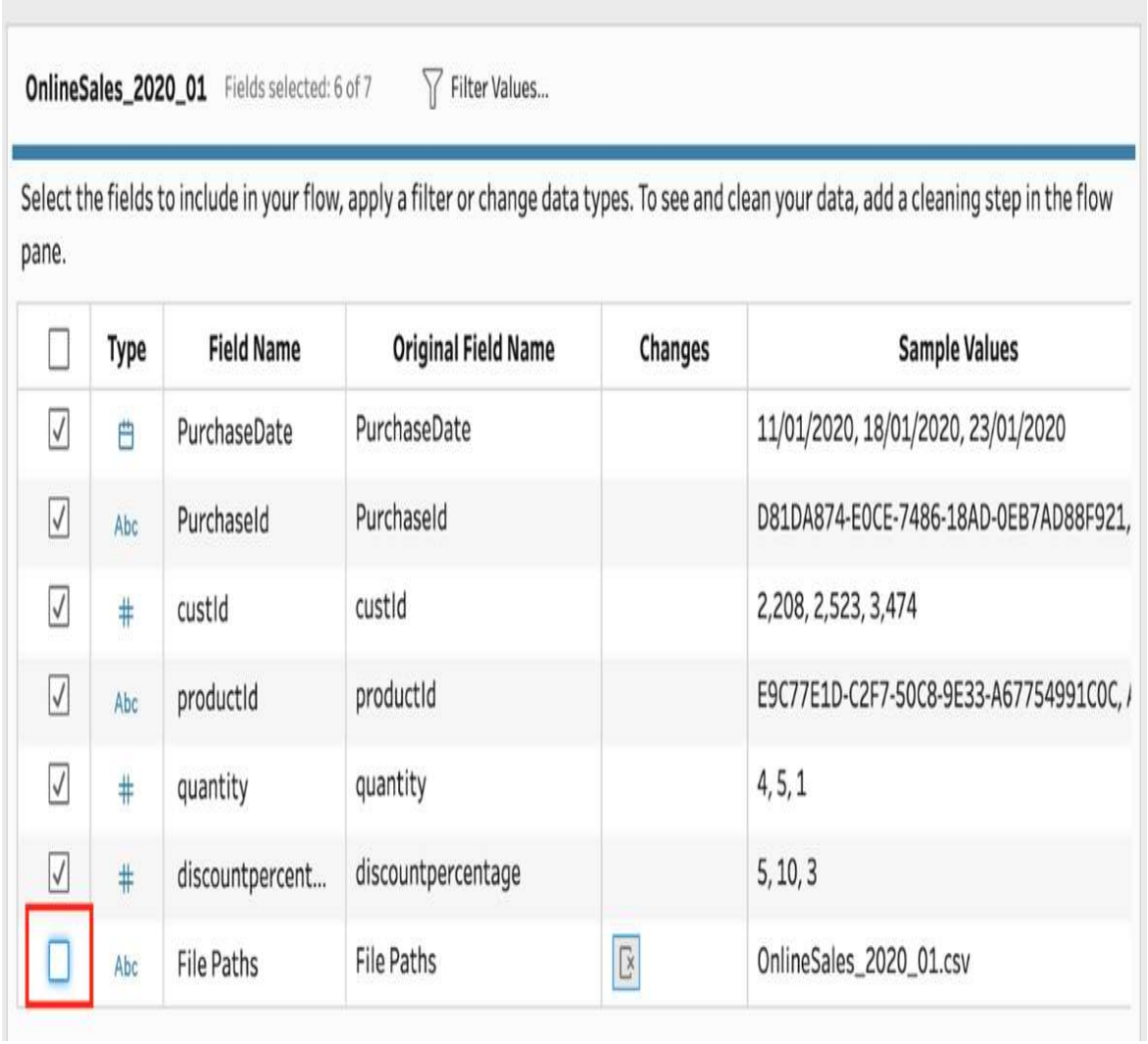

Figure 9.2 – Deselecting the File Paths field

3. Observe the field list and note how Tableau Prep has wrongly assigned a numeric data type to the **custId** field. This field represents the customer ID and although it consists of numbers, it will not be used as such in any calculation. Correct the data type by clicking the data type icon # and select **String** instead:

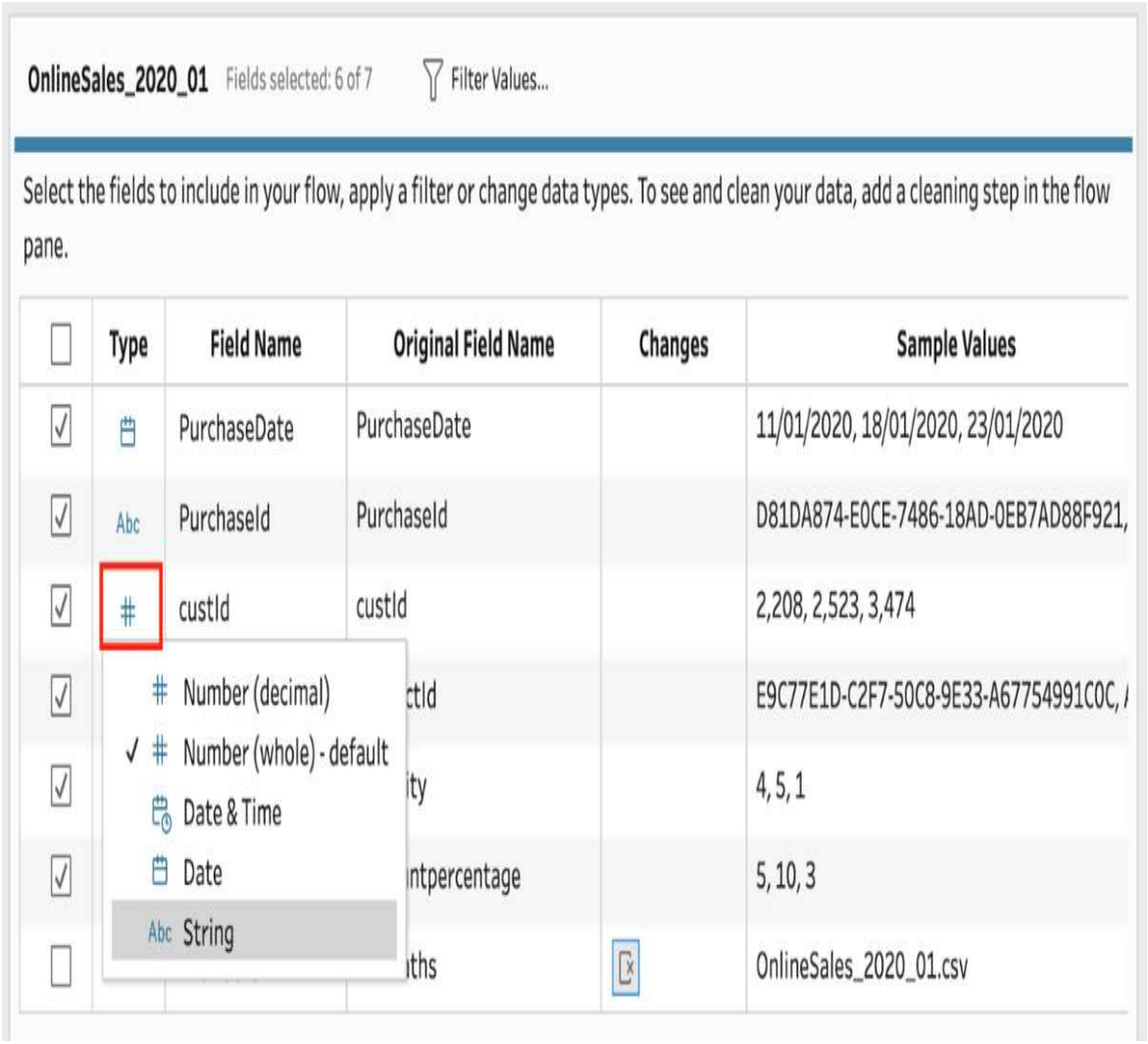

Figure 9.3 – Setting the custId type to String

4. Next, let's add the sales data for our physical stores. Use the Connect to Excel functionality and select the **STORE_SALES_EXPORT.xlsx** file provided in the sample files folder. Unlike the online sales data we have worked with so far, this dataset contains data for the full 6 months, so we don't have to perform a union here.
5. With the new input selected, correct the data type for the **TransactionID** and **CustomerID** fields by changing the type to **String**. This is the same solution we applied in *Step 3*, and something that occurs frequently in real-world scenarios when your data contains numeric IDs.
6. Before we continue, let's name the steps in our flow. As we'll build out a relatively large flow, naming your steps is useful for ensuring that your flow remains easy to understand. Rename the **OnlineSales_2020_01** input step by double-clicking its name and changing the name to **Online Sales**. Then, rename the second dataset to **In-Store Sales**:

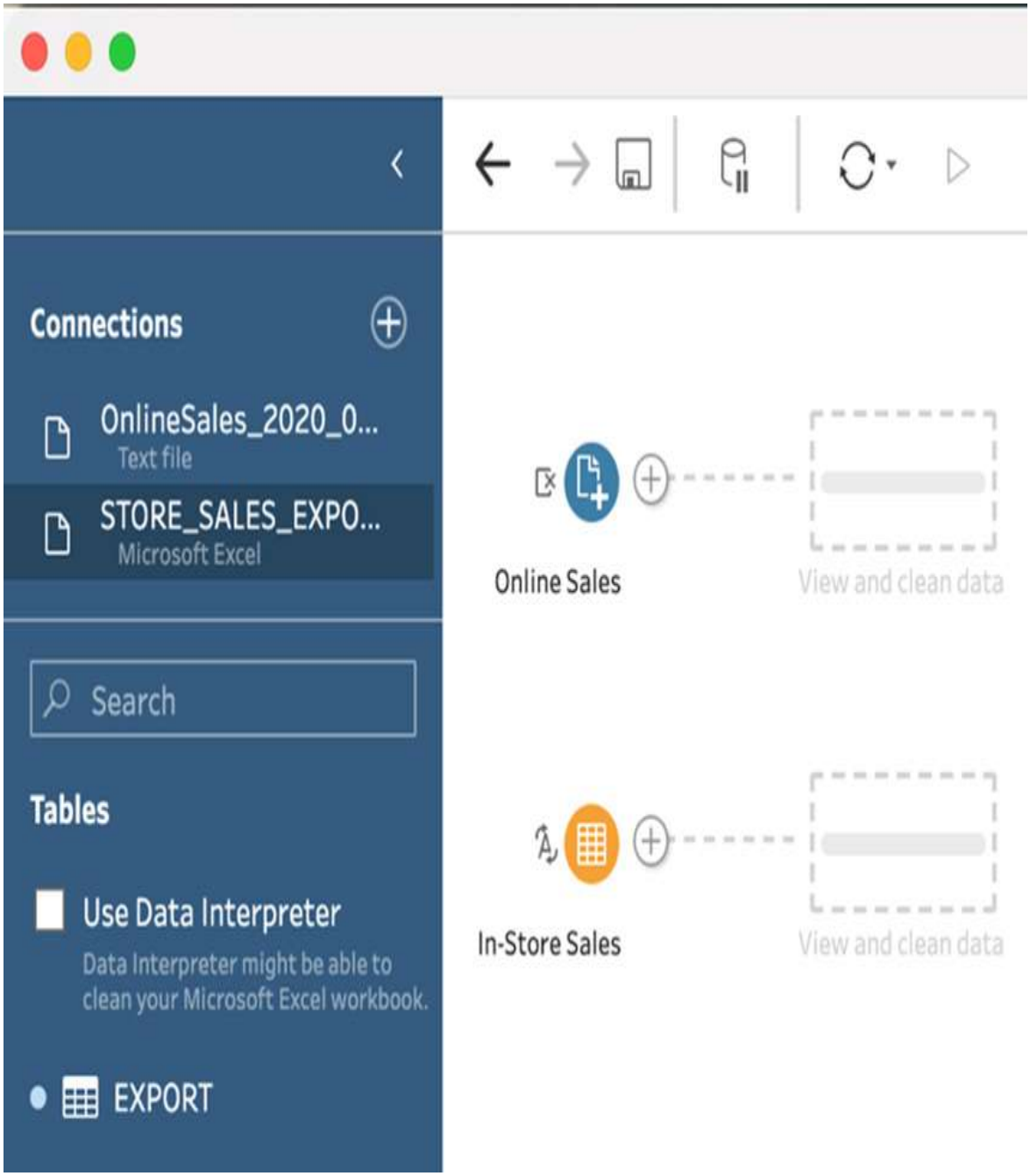

Figure 9.4 – Double-clicking the inputs to rename them

7. Click the + icon besides the **In-Store Sales** input and then select **Clean Step**. Observe the **TransactionDate** field values, as highlighted in the following screenshot. Each value here seems to be a number and not a date. This is because the input data has been formatted as a **UNIX TIMESTAMP**. This type of data issue is not uncommon, and we need to create a simple calculated field to convert this value to a date, as Tableau Prep cannot automatically convert this source field to a date:

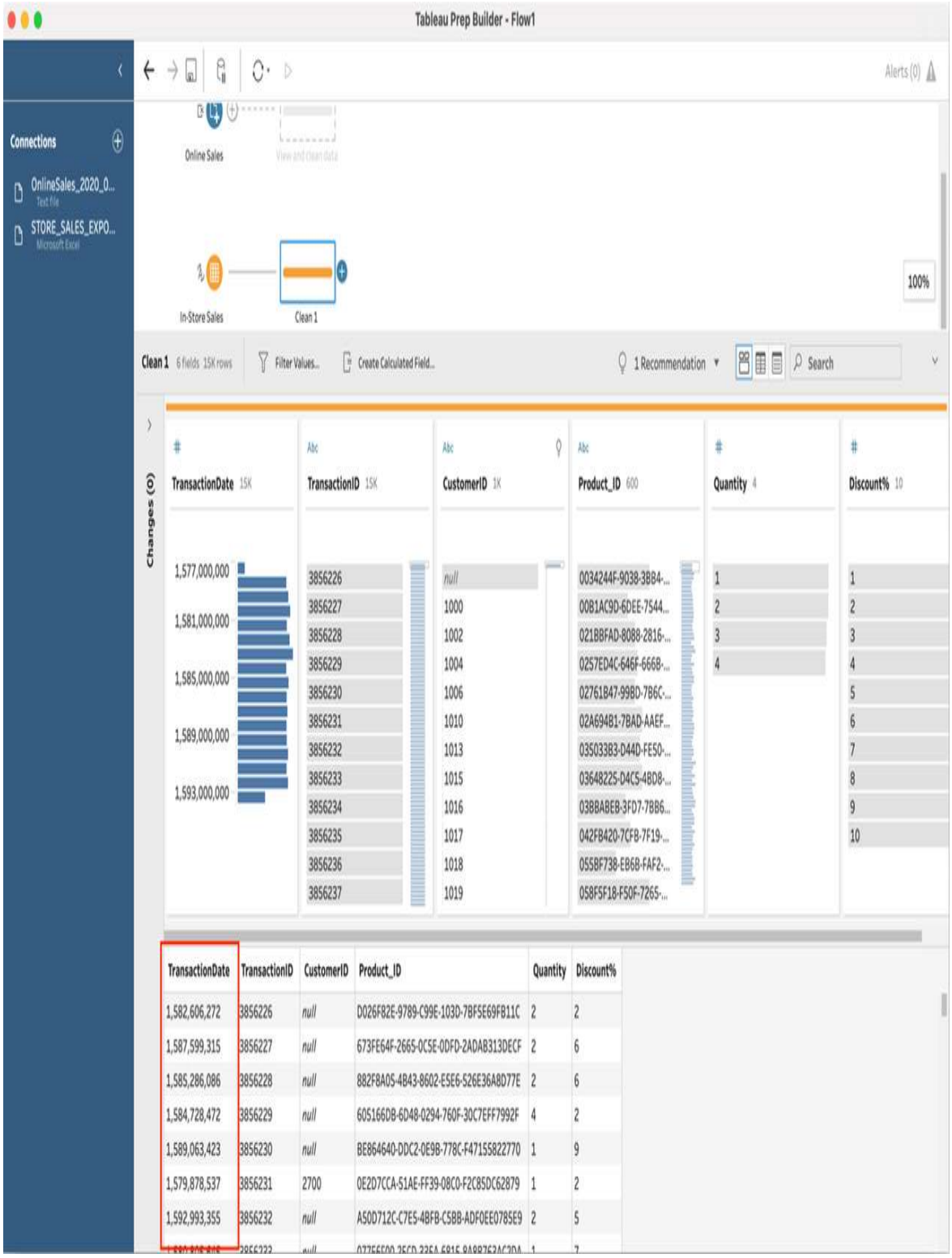

Figure 9.5 – This dataset contains a UNIX TIMESTAMP field

8. With **Clean Step** still selected, click on **Create Calculated Field...**. Name the new field **Purchase Date** and set the expression to **DATEADD('second',[TransactionDate],#1970-01-01#)**, which is the expression

to convert a Unix timestamp to a regular datetime format. Click **Save** when done to apply your new calculation:

Figure 9.6 – Calculating the date value

Observe the outcome and ensure that the format is indeed date and time, as shown in the following screenshot:

Purchase Date	TransactionDate
25/02/2020, 04:51:12	1,582,606,272
22/04/2020, 23:48:35	1,587,599,315
27/03/2020, 05:14:46	1,585,286,086
20/03/2020, 18:21:12	1,584,728,472
09/05/2020, 22:30:23	1,589,063,423
24/01/2020, 15:08:57	1,579,878,537
24/06/2020, 10:09:15	1,592,993,355

Figure 9.7 – The result of converting the Unix timestamp

9. We won't need the specific time for the purchase date, so let's change the data type from **Date & Time** to **Date** by clicking the data type icon in the field list and then selecting **Date**, as shown in the following screenshot:

Figure 9.8 – Using the icon dropdown to change the data type to Date

10. We also no longer require the original **TransactionDate** field. To remove this field using the clean step, click the context menu next to the field name and then select **Remove**, as shown in the following screenshot:

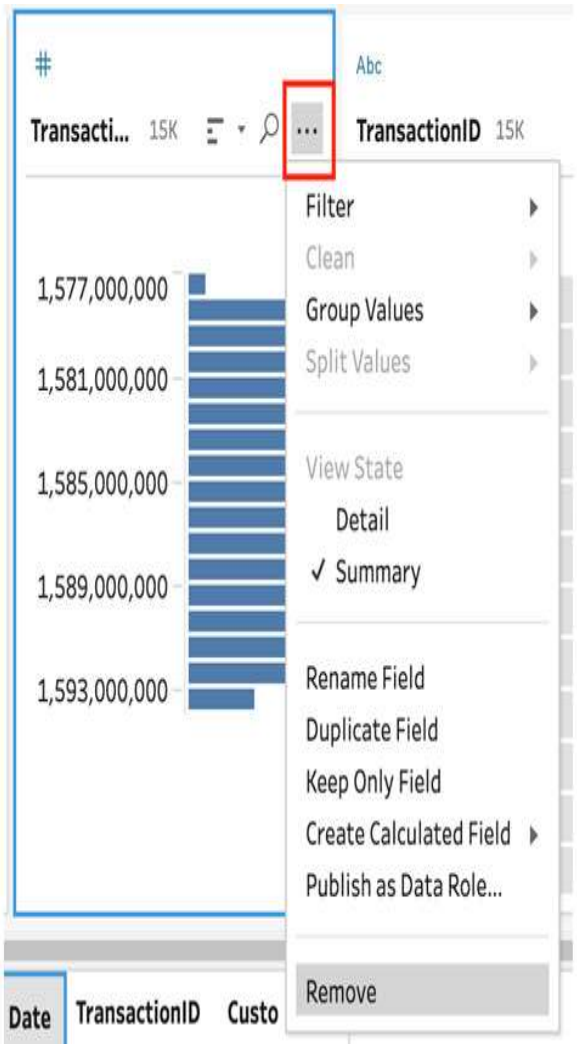

Figure 9.9 – Removing the TransactionDate field using the clean step

11. Next, we're going to combine the online sales data with our in-store sales data. To do this, we need to perform a union. Drag and hover **Clean Step** on top of the **Online Sales** input. Then, from the options that appear, hover over **Union** and release, as shown in the following screenshot:

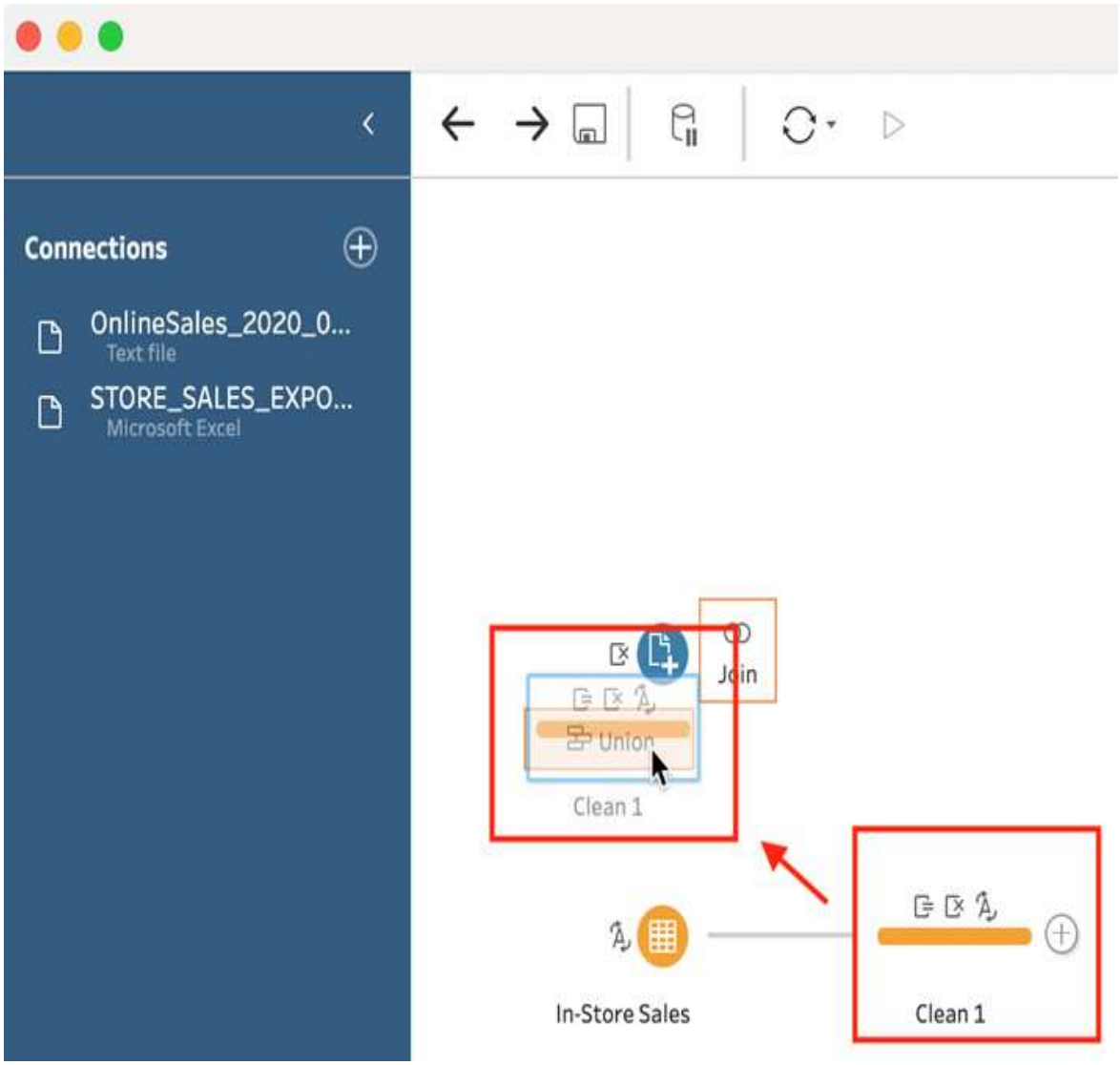

Figure 9.10 – Dragging the clean step on top of the online sales step

This will automatically create a **Union** step and your screen should look like the following screenshot:

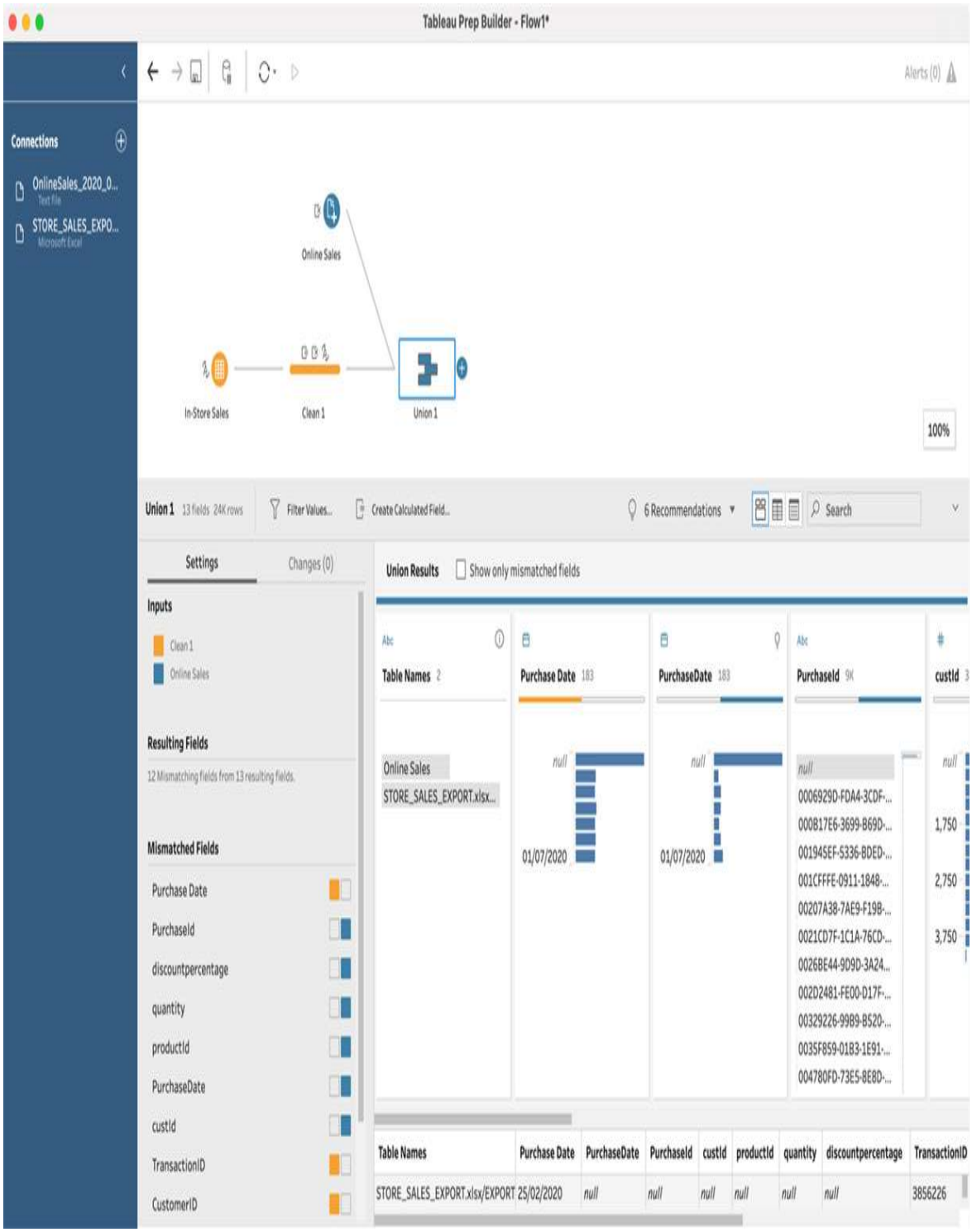

Figure 9.11 – A Union step has been added as a result of the drag and drop action

12. In the bottom left of the window, we can see that there are quite a few **Mismatched Fields** options. This is to be expected when you combine data from different sources, as we have just done. Fortunately, both our sources include fields with a similar meaning and they just have different field names, which prevents Tableau Prep from automatically aligning them. To resolve this, click the field pairs that

represent the same information (hold the *Command* or *CTRL* key to select the second field), and then right-click and select **Merge Fields**, as shown in the following screenshot for the **Purchase Date** and **PurchaseDate** fields. Note that the newly merged field will take the name of the field you right-clicked:

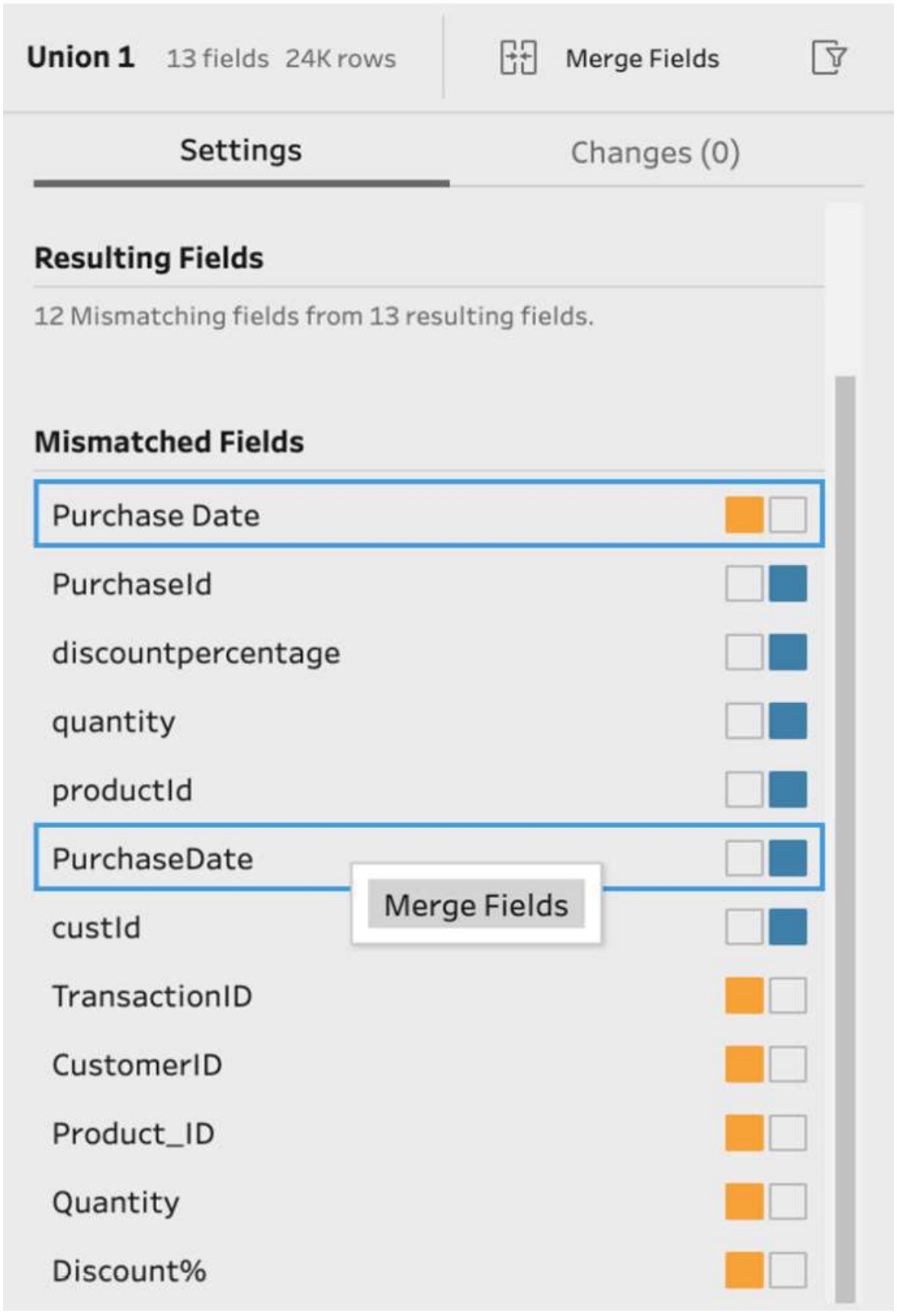
Union 1 13 fields 24K rows
Merge Fields
Settings
Changes (0)
Resulting Fields
12 Mismatching fields from 13 resulting fields.
Mismatched Fields
Purchase Date
PurchaseId
discountpercentage
quantity
productId
PurchaseDate
Merge Fields
custId
TransactionID
CustomerID
Product_ID
Quantity
Discount%

Figure 9.12 – Right-clicking and select Merge Fields to merge the selected fields

Perform this MERGE FIELDS action for the field pairs listed here:

1. **Purchase Date** and **PurchaseDate**
2. **PurchaseId** and **TransactionID**
3. **discountpercentage** and **Discount%**
4. **quantity** and **Quantity**
5. **productId** and **Product_ID**
6. **custId** and **CustomerID**

When you've completed all the merges, your **Settings** tab should look like the following screenshot:

Figure 9.13 – The Resulting Fields section is empty when all mismatches have been merged

13. With the **Union** step still selected, notice that a new field has appeared in the **Union Results** field list, named **Table Names**. This field indicates where each row originated, that is, from our online sales dataset or the in-store dataset. This field may come in handy for downstream analysis, so let's rename the value **STORE_SALES_EXPORT.xlsx/EXPORT** to **In-Store Sales** and the field name itself to **Sales Type**, as shown in the following screenshot:

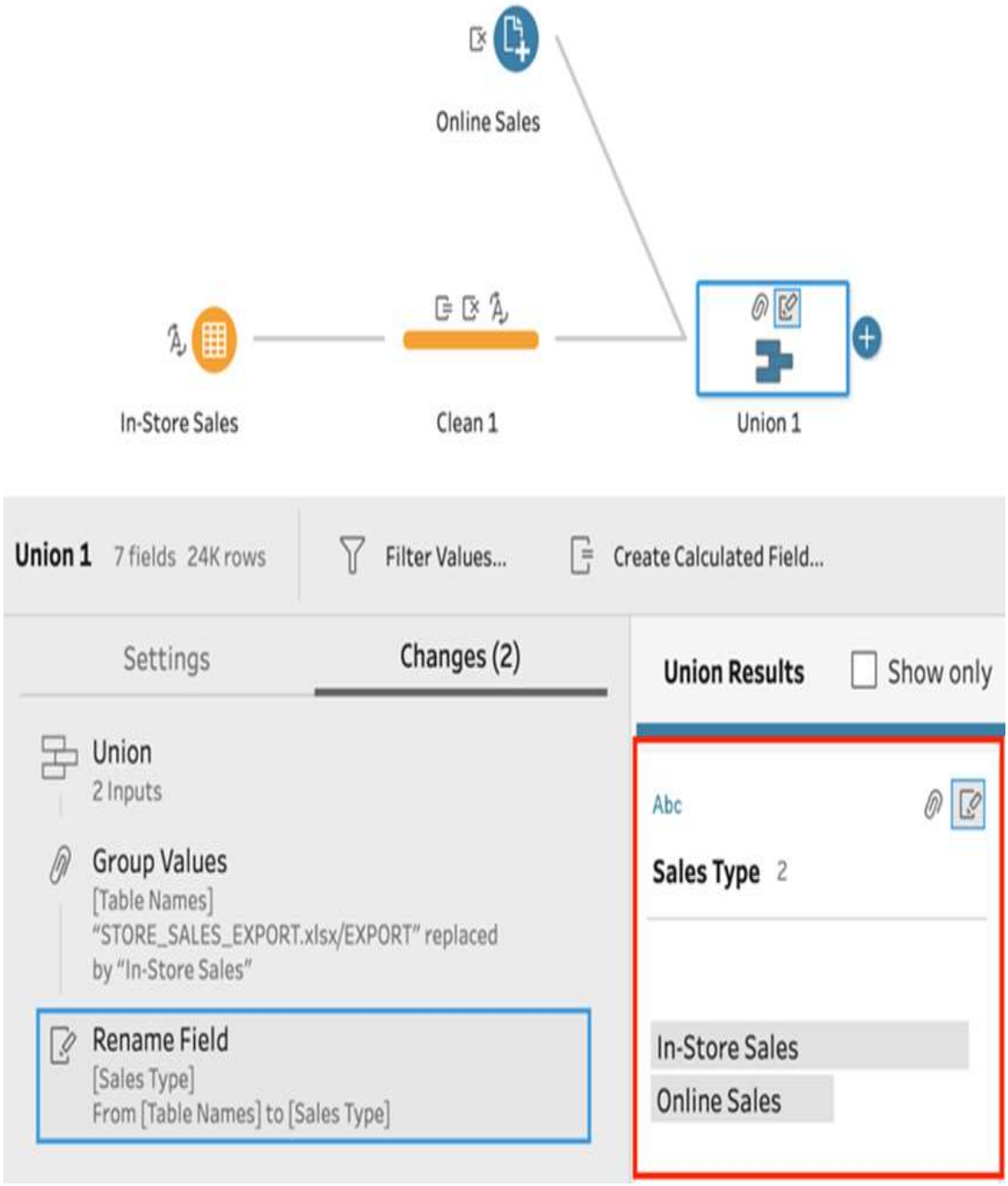

Figure 9.14 – Renaming the value and field name for the automatically added Table Names field

14. Next, create another data connection, this time to the **Products.csv** file, provided in the sample files with this lesson.
15. This **Products.csv** file we just added contains descriptive product information. For example, instead of using a product ID such as *1931E212-FF85-3A36-620A-8C56D1C6B605*, we can get a name such as *Modern Utility Laptop Messenger Bag*. To add this information to our existing dataset as additional columns, we

need to perform a join. To do this, drag the input on top of the **Union** step. When the **Union** and **Join** options appear, drop the input on top of the **Join** option to instantly add a join step.

16. Configure the join by specifying a common field between the two datasets, in this case, **productId** and **ID**, as shown in the following screenshot. The default join type, inner, can be left as-is:

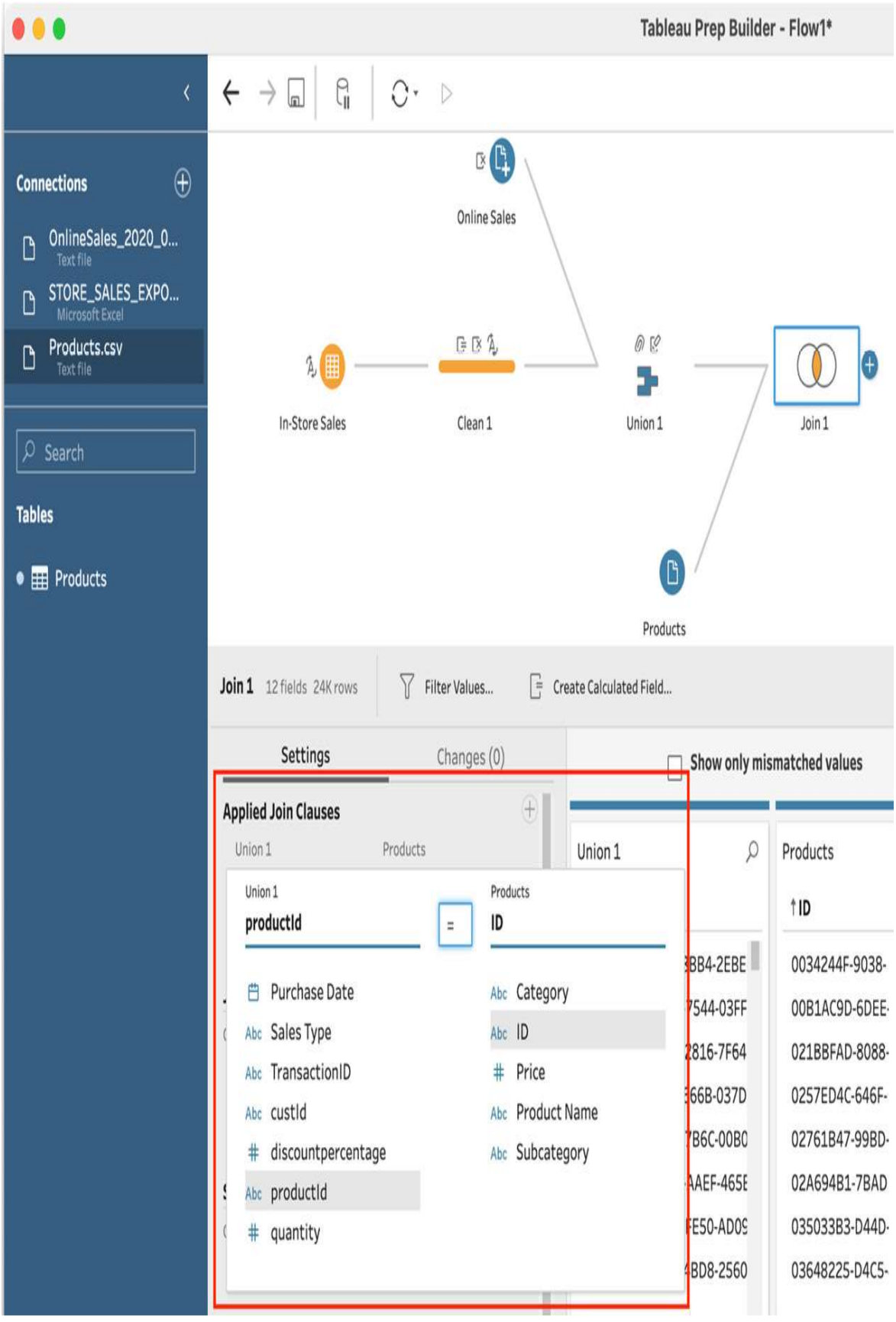
Tableau Prep Builder - Flow1*
Connections
OnlineSales_2020_0...
Text file
STORE_SALES_EXPO...
Microsoft Excel
Products.csv
Text file
Search
Tables
Products
Online Sales
In-Store Sales
Clean 1
Union 1
Join 1
Products
Join 1 12 fields 24K rows
Filter Values...
Create Calculated Field...
Settings
Changes (0)
Show only mismatched values
Applied Join Clauses
Union 1
Products
Union 1
productId
=
Products
ID
Purchase Date
Sales Type
TransactionID
custId
discountpercentage
productId
quantity
Category
ID
Price
Product Name
Subcategory
Union 1
Products
ID
0034244F-9038-
00B1AC9D-6DEE-
021BBFAD-8088-
0257ED4C-646F-
02761B47-99BD-
02A694B1-7BAD
035033B3-D44D-
03648225-D4C5-

Figure 9.15 – Configuring the join to join on the productId and ID fields

17. As is typical with a join, we now have a redundant field for product ID. Remove the **ID** field from the field list by selecting **Remove** from the field context menu. This way, we only have the **productId** field as the identifier:

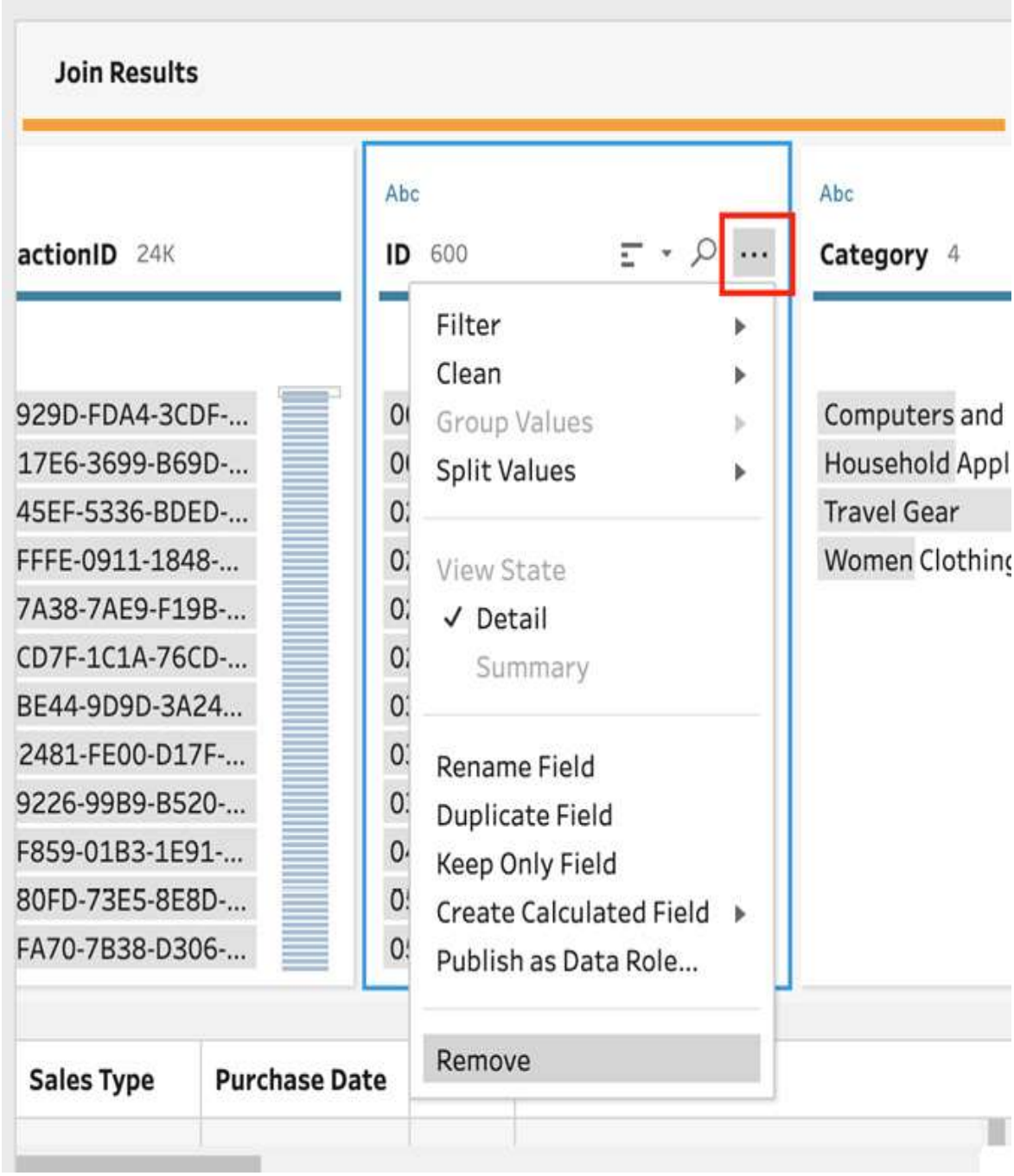

Figure 9.16 – Removing the field ID from Join Results

18. Add another data source, this time a Tableau extract named **ShippingData.hyper**. This data is provided by our shipping department and contains shipping information for sales completed online. Rename the step **Shipping**.

19. Add a clean step to the **Shipping** input and observe the field named **ID**. The shipping ID here is made up of two identifiers; first, the shipping department's ID, followed by an underscore symbol and then the purchase ID. We need to split this field so that these values are stored separately. To do this, select **Custom Split...** from the context menu for the **ID** field:

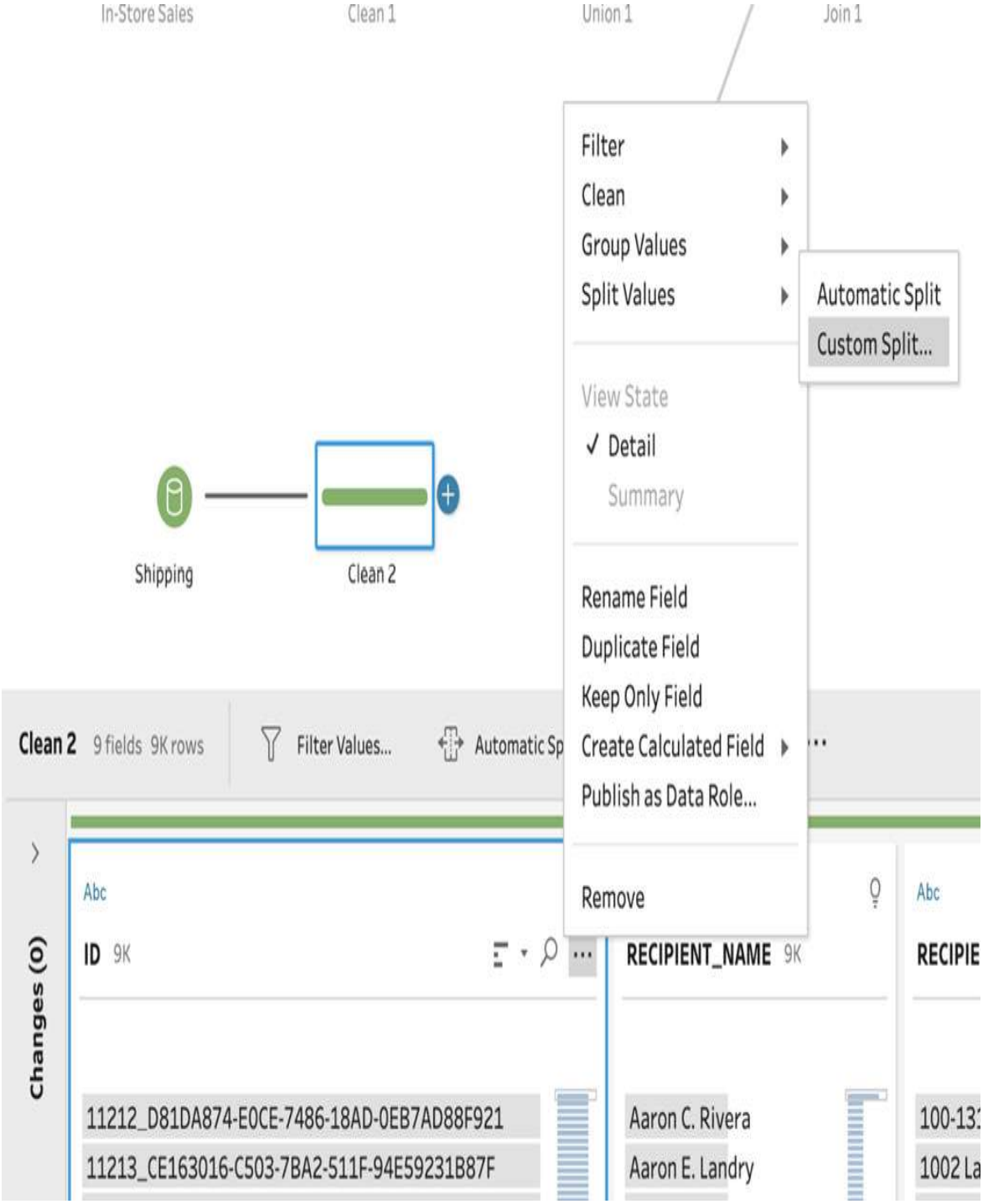

Figure 9.17 – Selecting Custom Split for the ID field

1. Configure the split to use the underscore (_) symbol as a separator and split the first **2** fields, as shown in the following screenshot:

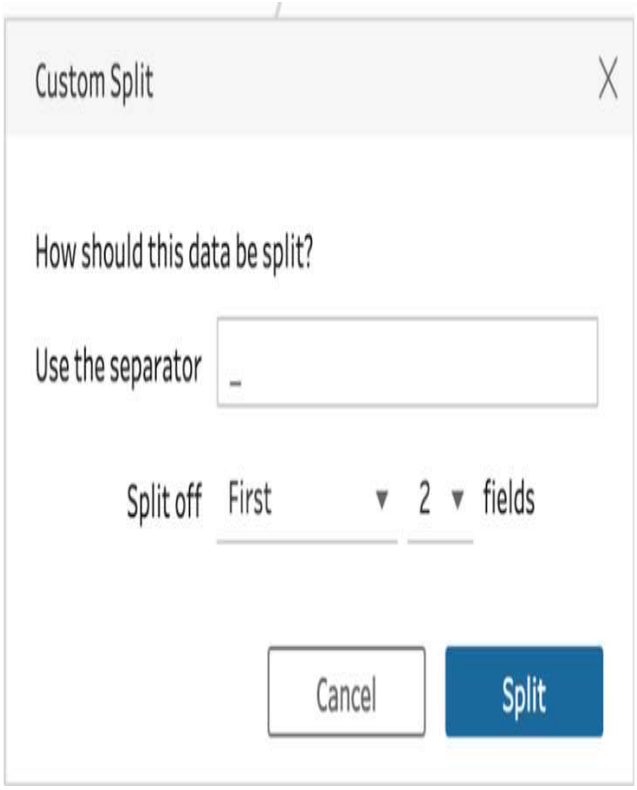

Figure 9.18 – Configuring Custom Split with an underscore and the first 2 fields

- When you're ready, click **Split**. This will then split the ID field into two new fields, named **ID - Split 1** and **ID - Split 2**:

20. Rename the **ID - Split 1** field to **Shipping ID** and the **ID - Split 2** field to **Purchase ID**.
21. We will no longer need the original **ID** field, so use the context menu to remove it from the dataset.
22. Drop the **Shipping** input on top of the existing join in order to create another **Join** step. Configure the join clause to use the **TransactionID** and **Purchase ID** fields to perform the join. Because only sales are shipped, the shipping data does not contain information for store sales. As such, we need to set this join to a left join type. A left join will result in including all data from the left dataset, which is our main flow, and any matching data from the right dataset, which is our shipping data. Set **Join Type** to **left** by selecting the left circle in the Venn diagram. Your flow and join settings should look like those in the following screenshot:

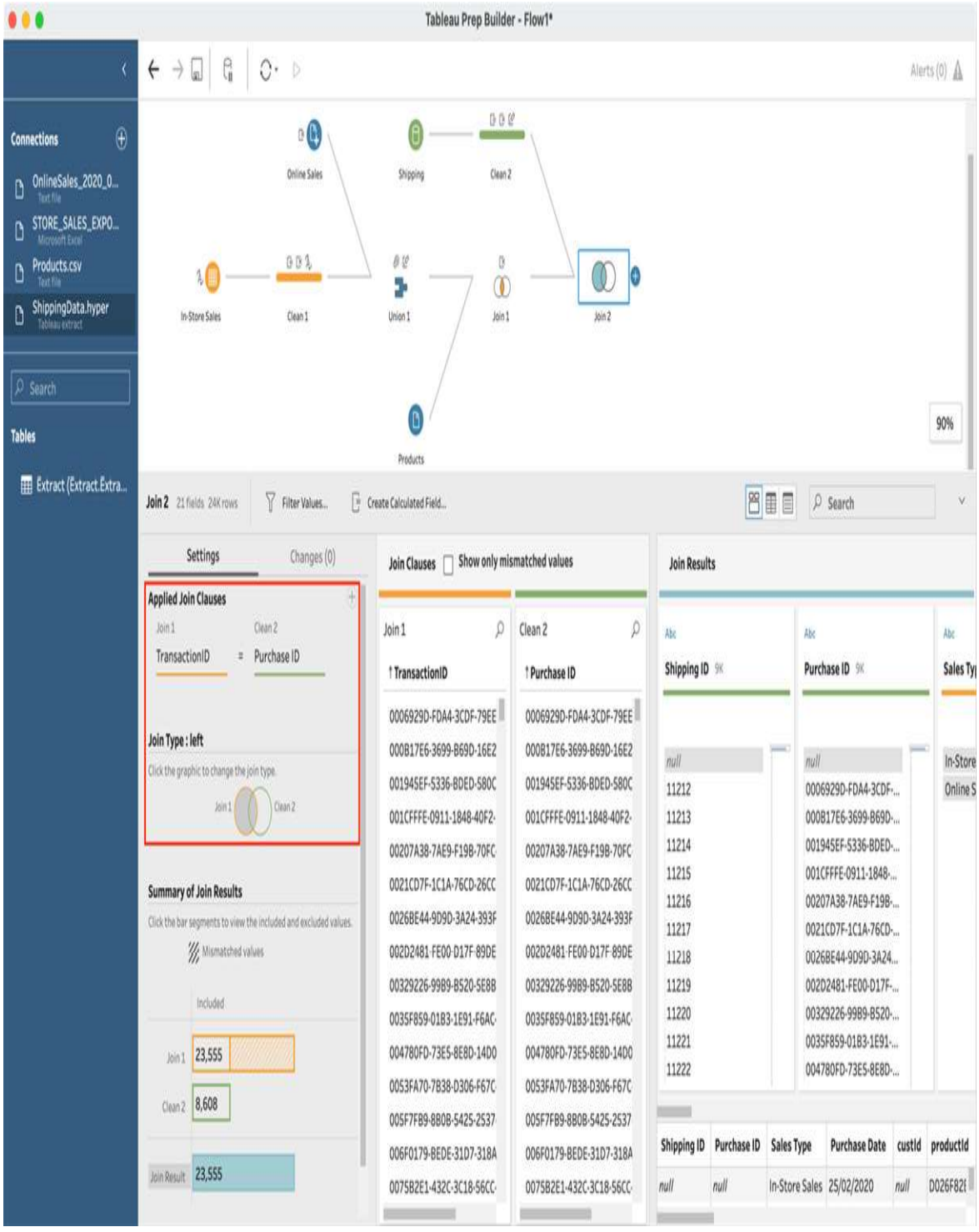

Figure 9.19 – Configuring the join clause and join type

23. Delete the now redundant **Purchase ID** field. We still have the **TransactionID** field to identify a given row of data.
24. Add your fifth data connection to this flow. This time, select the **CustomerList.csv** text file. This input contains information about our customers, such as their full name. Rename the input step to **Customers**.

25. The **Customers** data includes an **ID** field, which has been incorrectly set to a numeric format by Tableau Prep. Click the data type icon for the **ID** field and change the type to **String**.
26. Join the **Customers** data to the existing flow by dropping it on the **Join 2** step. Configure the join clause to join on the **custId** and **id** fields. Because in-store checkouts do not always involve a customer loyalty card, the customer ID is not always known. Given the missing customer IDs, set the join type to **left** using the Venn diagram so that all rows are included from our main flow, including those for which we do not have customer details.
27. Delete the redundant customer ID field, named **id**, which originated from the **Customers** data.
28. Add our final data connection, the text file named **returns_h1_2020.csv**, and rename the step to **Returns**.
29. Correct the data type for the **return_id** field by setting it to **String**.
30. Rename the **status** field to **Return Status** so that we don't mix it up later with the existing status fields from the **Shipping** and **Customer** data.
31. Join the **Returns** step with the main flow by dropping it on top of **Join 3** to create a new join. Configure the join clause to use the **TransactionID** and **purchase_id** fields. Once more, use the Venn diagram to set the join type to **left**. Not all customers are returns, so we want to return all transactions and any matched rows from the **Returns** dataset.
32. Remove the redundant **purchase_id** field from the dataset.
33. Click the + icon on the last join and add a **Clean** step. Using the **Clean** step, rename the fields as follows:
    1. **custId** to **Customer ID**
    2. **productId** to **Product ID**
    3. **quantity** to **Quantity**
    4. **discountpercentage** to **Discount %**
    5. **TransactionID** to **Transaction ID**
    6. **RECIPIENT_NAME** to **Recipient Name**
    7. **RECIPIENT_STREET** to **Recipient Street**
    8. **RECIPIENT_CITY** to **Recipient City**
    9. **RECIPIENT_POSTAL** to **Recipient Postal**
    10. **RECIPIENT_REGION** to **Recipient Region**
    11. **SHIPMODE** to **Shipping Mode**
    12. **TRACEID** to **Shipping Courier Tracking ID**
    13. **STATUS** to **Shipping Status**
    14. **name** to **Customer Name**
    15. **surname** to **Customer Surname**
    16. **status** to **Customer Membership Status**
    17. **return_id** to **Return ID**
34. As a final step, we need to add an output step to our flow. Click the + icon on the **Clean** step and select **Output**. Configure the output to write to a location of your choosing and set the filename to **2020-H1 Sales Data.csv** and the **Output** type to **CSV**. Your final flow should look like the following screenshot:

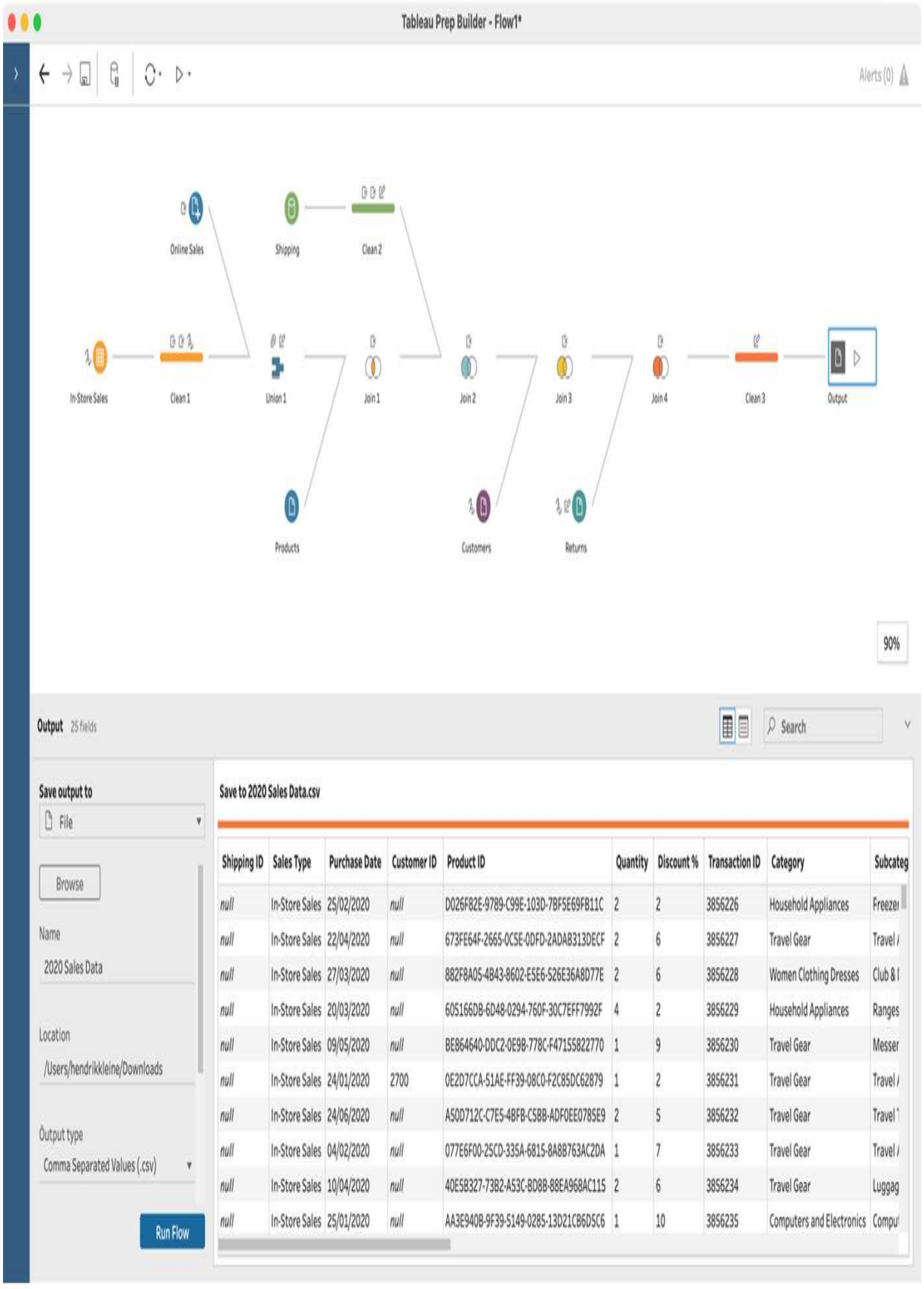
Tableau Prep Builder - Flow1*
Alerts (0)
Online Sales
Shipping
Clean 2
In-Store Sales
Clean 1
Union 1
Join 1
Join 2
Join 3
Join 4
Clean 3
Output
Products
Customers
Returns
90%
Output 25 fields
Search
Save output to
File
Browse
Name
2020 Sales Data
Location
/Users/hendrikkleine/Downloads
Output type
Comma Separated Values (.csv)
Run Flow
Save to 2020 Sales Data.csv
Shipping ID
Sales Type
Purchase Date
Customer ID
Product ID
Quantity
Discount %
Transaction ID
Category
Subcateg
null In-Store Sales 25/02/2020 null D026F82E-9789-C99E-103D-7BF5E69FB11C 2 2 3856226 Household Appliances Freezer
null In-Store Sales 22/04/2020 null 673FE64F-2665-0C5E-0DFD-2ADAB313DECF 2 6 3856227 Travel Gear Travel
null In-Store Sales 27/03/2020 null 882F8A05-4B43-8602-E5E6-526E36A8D77E 2 6 3856228 Women Clothing Dresses Club & I
null In-Store Sales 20/03/2020 null 605166DB-6D48-0294-760F-30C7EFF7992F 4 2 3856229 Household Appliances Ranges
null In-Store Sales 09/05/2020 null BE864640-DDC2-0E9B-778C-F47155822770 1 9 3856230 Travel Gear Messer
null In-Store Sales 24/01/2020 2700 0E2D7CCA-51AE-FF39-08C0-F2C85DC62879 1 2 3856231 Travel Gear Travel
null In-Store Sales 24/06/2020 null A50D712C-C7E5-4BFB-C5BB-ADF0EE0785E9 2 5 3856232 Travel Gear Travel
null In-Store Sales 04/02/2020 null 077E6F00-25CD-335A-6815-8A8B763AC2DA 1 7 3856233 Travel Gear Travel
null In-Store Sales 10/04/2020 null 40E5B327-73B2-A53C-BD8B-88EA968AC115 2 6 3856234 Travel Gear Luggag
null In-Store Sales 25/01/2020 null AA3E940B-9F39-5149-0285-13D21CB6D5C6 1 10 3856235 Computers and Electronics Compu