talend | Connecting the Data-Driven Enterprise

Participant Guide

Talend Studio Introduction

Version 6.5

Talend Inc.
800 Bridge Parkway, Suite 200
Redwood City, CA 94065
United States
+1 (650) 539 3200

Welcome to Talend Training

Congratulations on choosing a Talend training course.

Working through the course

You will develop your skills by working through use cases and practice exercises using live software. Completing the exercises is critical to learning!

If you are following a self-paced, on-demand training (ODT) module, and you need an answer to proceed with a particular exercise, use the help suggestions on your image desktop. If you can't access your image, contact customercare@talend.com.

Exploring

You will be working in actual Talend software, not a simulation. We hope you have fun and get lots of practice using the software! However, if you work on tasks beyond the scope of the training, you could run out of time with the environment, or you could mess up data or Jobs needed for subsequent exercises. We suggest finishing the course first, and if you have remaining time, explore as you wish. Keep in mind that our technical support team can't assist with your exploring beyond the course materials.

For more information

Talend product documentation (help.talend.com)

Talend Community (community.talend.com)

Sharing

Intentionally blank

CONTENTS

Intentionally blank

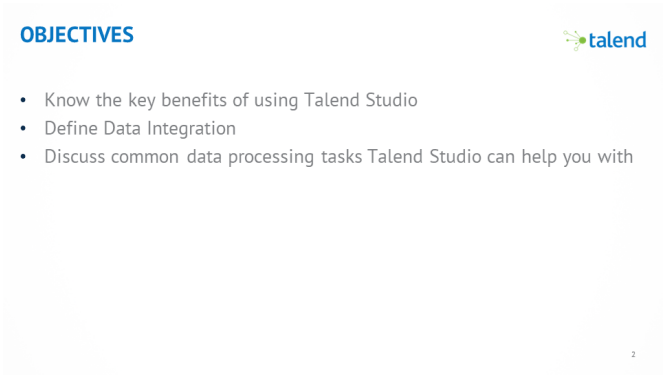
LESSON 1

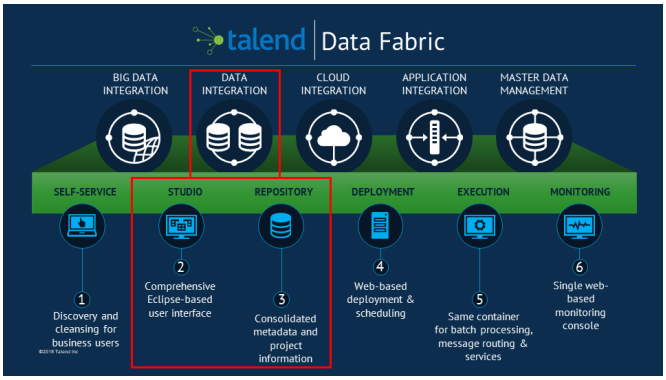
Introduction to Talend Studio in Context

In this lesson you have an overview of Talend Studio with basic data integration examples.

This chapter discusses:

Concepts





Talend Data Fabric uses the same Eclipse environment for several applications: big data integration, data integration, application integration (ESB), mdm and cloud integration, and data preparation.

This course covers the basics of Talend Studio, introducing the most commonly used components with data integration examples.

Talend Studio provides rich capabilities:

Design Faster: easy data integration with drag-and-drop user interface, no scripting.

Collaborate better: team support for developers, reliable continuous delivery of new functionalities.

Cleanse earlier: profiling and data matching studio to cleanse your data more accurately.

Manage more: centralize and manage deployment through a web-based administration console.

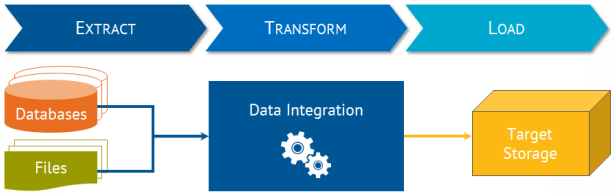
Scale easier: scale up with a high availability environment and advanced clustering.

KEY BENEFITS OF TALEND STUDIO

- Connectivity to source and target data systems
- Scalability and performance
- Transformation flexibility
- Data cleansing components
- Logging and exception handling
- Easy integration with web services
- Hundreds of components available

WHAT IS DATA INTEGRATION?

- Merge data coming from different systems
- Implementation is simplified by using an **ETL** tool



Data integration primarily supports the analytical processing of large, distributed data sets by aligning and combining different data sources through an ETL (Extract Transform Load) tool:

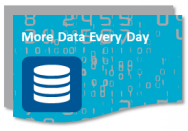
Extract: get the data from source systems as efficiently as possible.

Transform: perform calculations/ transformations on data. At this step, cleansing and standardization are also required most of the time.

Load: load the data to the target storage.

WHY DATA INTEGRATION IN TALEND STUDIO?

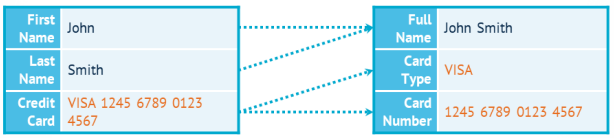
- Different formats
- Different database structures and models
- Different standards
- Different systems
- Removes mistakes and corrects data
- Adjusts data from multiple sources to be used together
- Structures and aggregates data to be used by BI tools



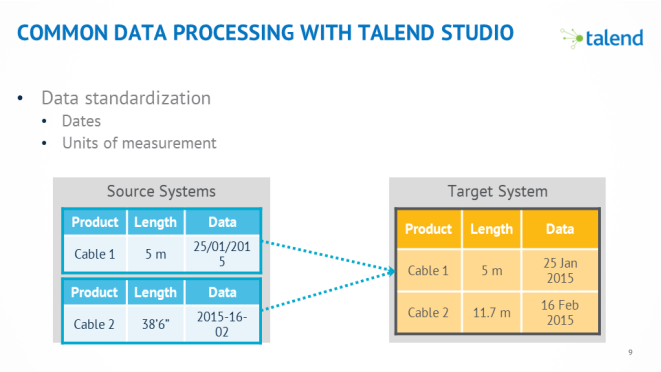
With Talend Studio you can get the most out of your data by managing and integrating a wide range of data formats, data structures, standards and systems. You can enhance and add value to your data by removing mistakes, correcting, organizing and aggregating data so that it can be processed by BI tools and many more.

COMMON DATA PROCESSING WITH TALEND STUDIO

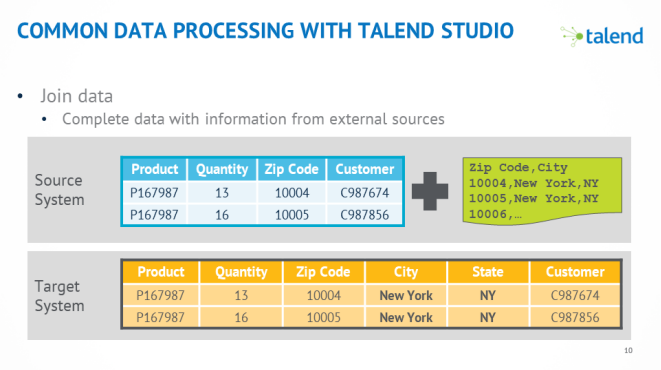
- Splitting or merging fields
  - Hundreds of components to assist you



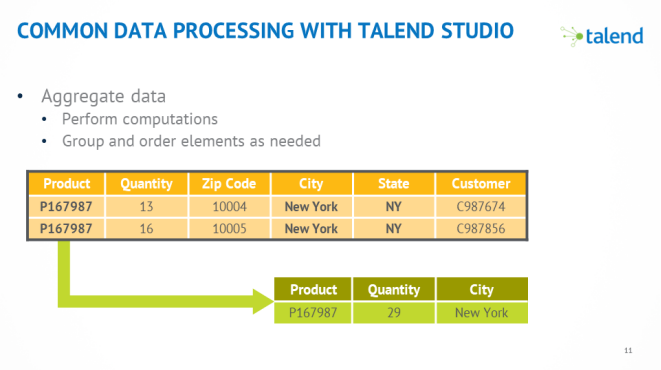
Splitting or merging fields is easy using predefined functions and mapping facilities.



When merging data on cable sales registered by the company in different locations, you want to use the same standards for data representation and measurement units.



Join data from heterogeneous sources and load it to a target system.



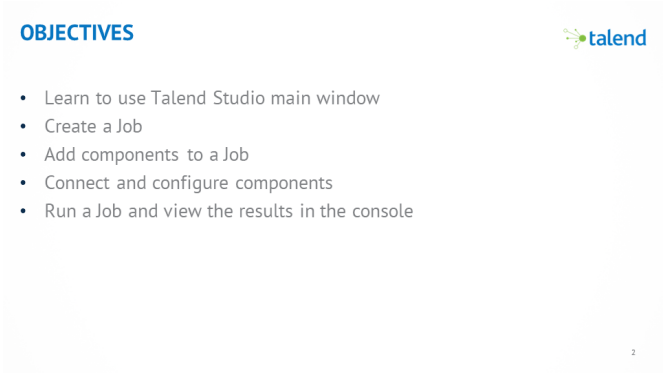
Aggregate data to perform computations and statistics.

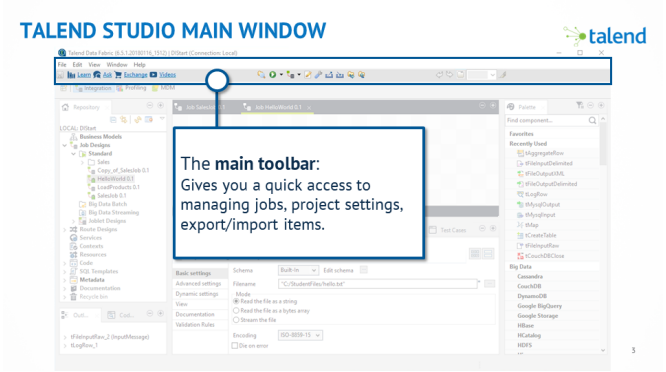
LESSON 2

Getting Started with Talend Studio

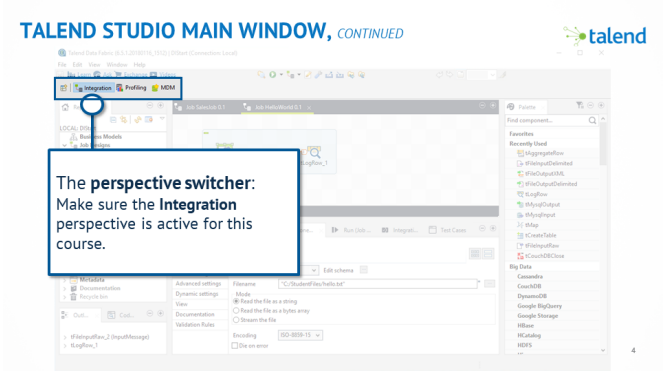
This chapter discusses:

Concepts

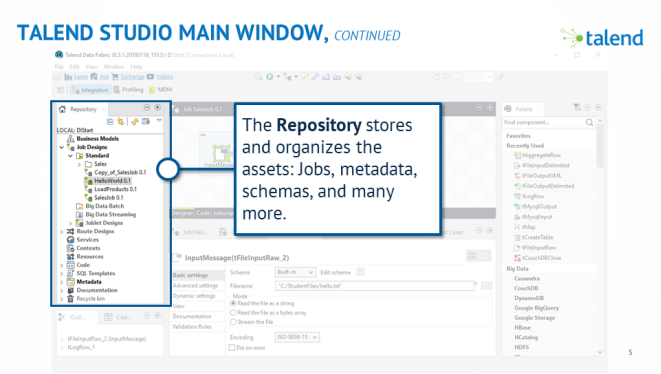


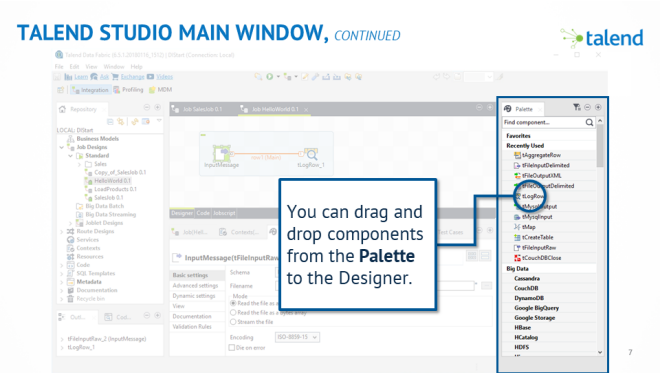


The **main toolbar** allows you to quickly find, run, or create Jobs, as well as importing or exporting items.

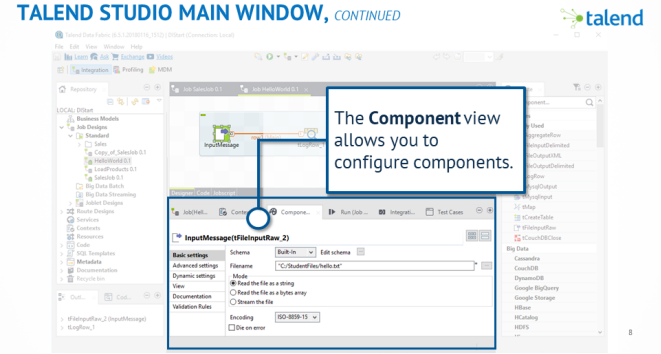


A **Perspective** defines the initial set and layout of views in the Talend Studio workbench. The selected perspective is always the active one. There are several perspectives in your training environment because the Talend Data Fabric can be used to create various types of projects, including data integration, Big Data, application integration, master data management, and data profiling. In this course, you use only the Integration perspective.

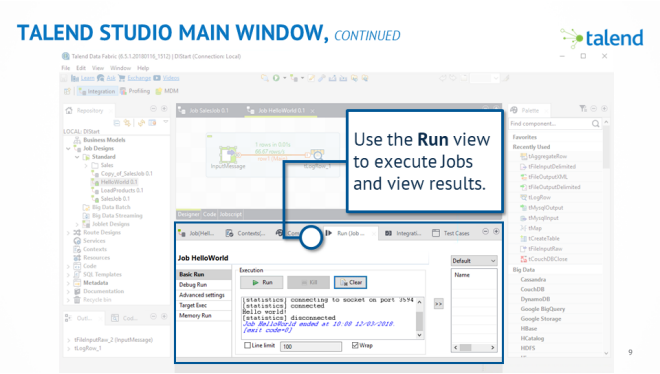




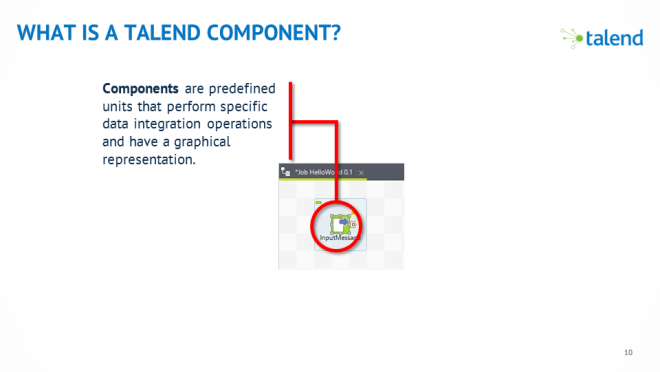
The **Palette** contains the technical components you use to design Jobs, grouped by families. Drag and drop components to the Designer to use them in your Job.



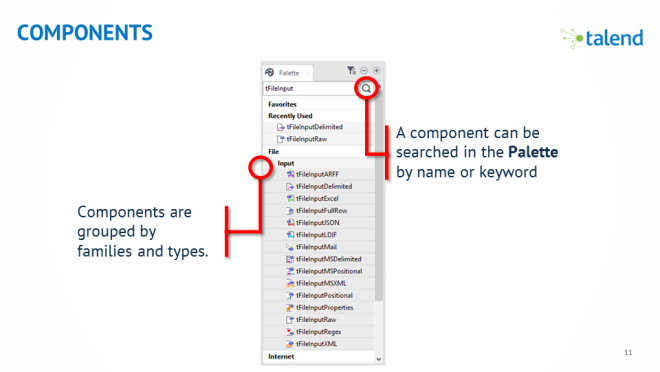
The **Component** view allows you to configure the properties of the component you selected in the Designer.



The **Run** view allows you to execute a Job and explore the results in the console.



Components are predefined units that perform specific data integration operations. They are implemented as Java code and have a graphical representation to allow easy configuration and simplified Job design.



Components are grouped by families (for example File, Databases) and types (for example Input, Process). Component main types are:

Input – Reads input from a source and returns output to the data flow

Process – Transforms data in the data flow

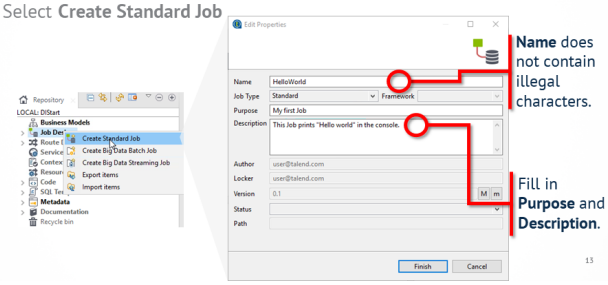
Output – Receives data from the data flow and outputs to a target

Virtual – Component constructed of two or more other components

Components are connected with **rows** (links that carry data) or **triggers** (links that transfer control).

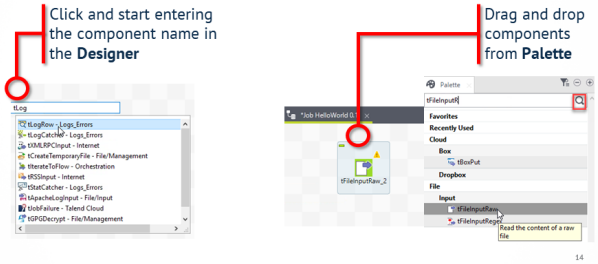
CREATE A JOB

- In the **Repository**, right-click **Job Designs**
- Select **Create Standard Job**



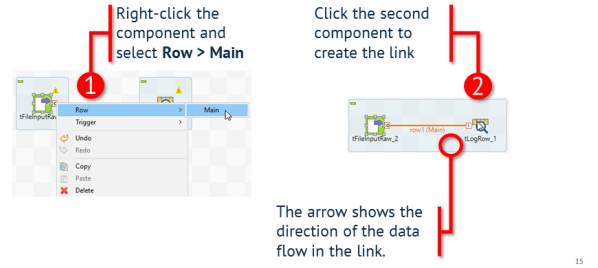
It is highly advised to fill in the **Purpose** and **Description** fields of a Job, for development and maintenance purpose.

ADD COMPONENTS TO A JOB

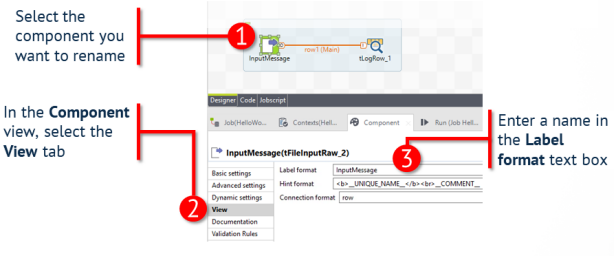


There are two methods to add components to your Job: directly from inside the Designer and using the Palette.

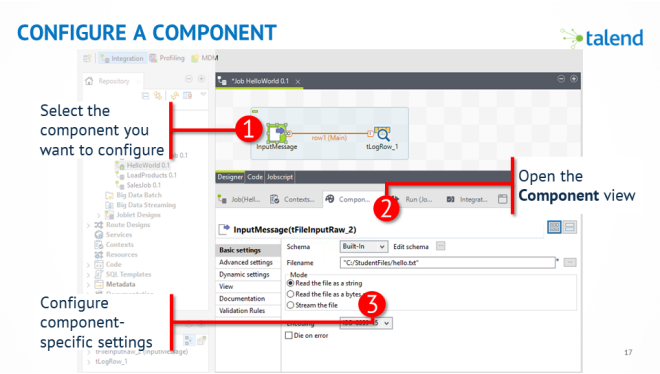
CONNECT COMPONENTS



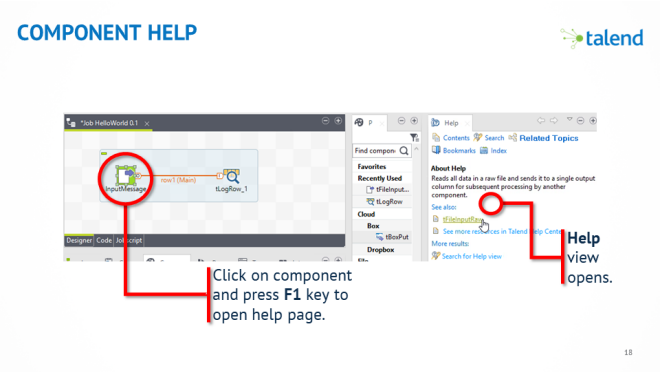
NAME A COMPONENT

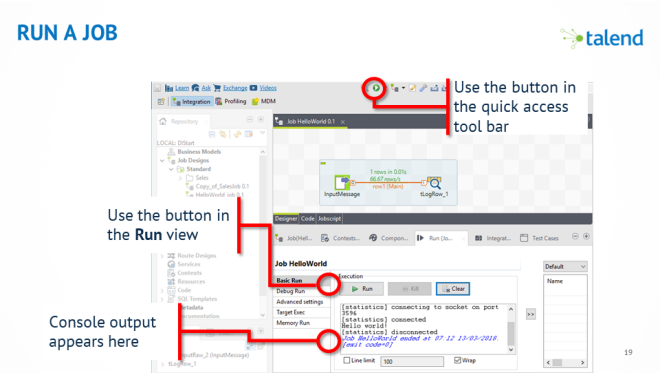


The **Label format** specifies the name of the component while the **Hint format** specifies the pop-up message displayed on mouse-over.

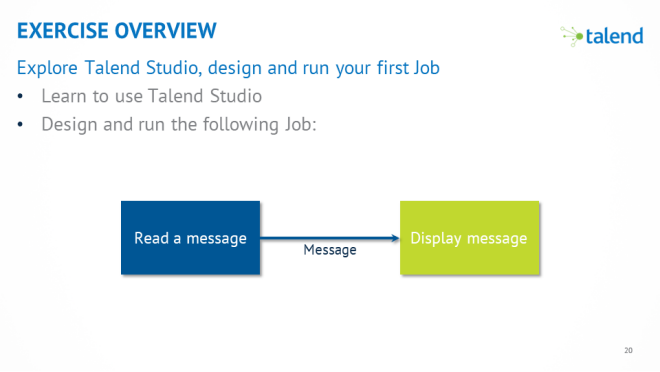


A double-click on a component will also open the **Component** view.





To execute a Job you can either use the Run icon in the Run view, or the one in the main toolbar. When you have several Job tabs open in the Designer, make sure to select the correct Job before using the Run icon.



LESSON SUMMARY

Learn to use Talend Studio main window

- Components, Job, The Designer, the Palette, the Repository etc.

Design a Job

- Create a Job
- Add components to a Job
- Connect and configure components
- Run a Job and visualize results in the console

21

Overview

Use Case

In this course, you are assigned a **Talend Data Fabric** training environment. The purpose of this lesson is to get you started with using Talend Studio. You create an empty project containing one simple job that displays *"Hello world!"* in the console.

Objectives

After completing this lesson, you will be able to:

- Create a project
- Create a Job
- Add components to a Job
- Link and configure components
- Run a Job and visualize the results in the console

Next step

The next step is to create a project.

Creating a Project

Task outline

Before developing projects in Talend Studio, you need to start Talend Studio and create a project.

This exercise takes about 5 mins to complete.

Creating a project

1. START TALEND STUDIO

   To start Talend Studio, double-click the Talend Studio icon on your desktop.

   The **Talend Data Fabric** window opens allowing you to open an existing project or create a new one.

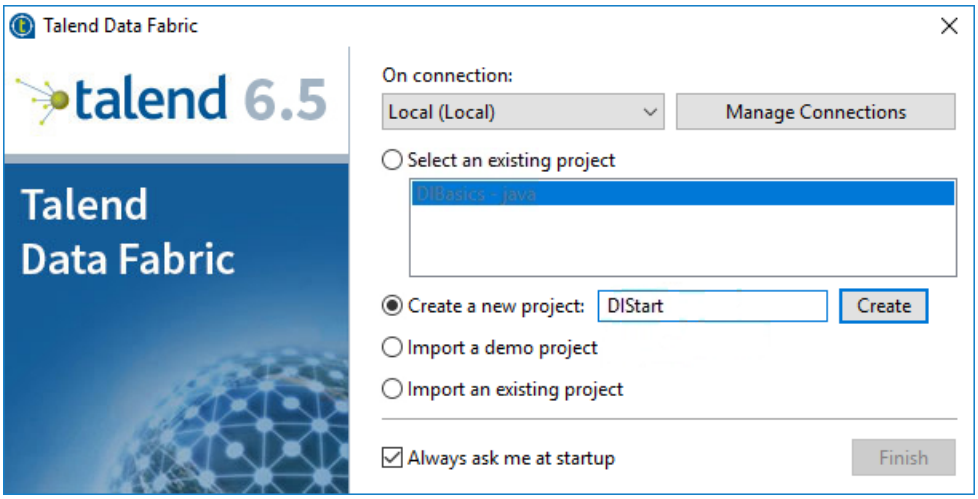
2. CREATE AN EMPTY PROJECT

   To create an empty project:

   a. For **On connection**, select **Local (Local)**, which means that once created, the project is stored locally, on your machine

   b. Select the radio button **Create a new project** and name it *DIStart*. Click **Create** and wait until the project appears in the list of existing projects. The project you created is now selected by default.

   c. Click the **Finish** button.

   NOTE:

   - If you are prompted to connect to the TalendForge community, select **Skip this Step**.
   - If the **Welcome** page appears, click **Start now!**

3. SWITCH TO THE INTEGRATION PERSPECTIVE

   The Talend Data Fabric Studio allows you to create several types of projects: data integration, big data, application integration, master data management, and data profiling. Depending on the project type, you can use one or more perspectives to work with.

In this course you work on data integration use cases, therefore you need to select the **Integration** perspective, if it is not already selected. The selected Perspective is always the active one.

Next step

You have learned how to start Talend Studio and create a new project. You are ready to create your first Job.

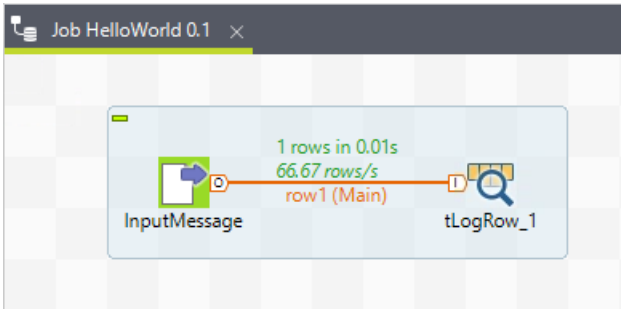
Creating Your First Job

Task outline

In this section, you create a Job that reads the string "Hello world!" from a file and sends it to the console.

This exercise takes about 15 mins to complete.

At the end, the Job looks like this:



Creating a Job

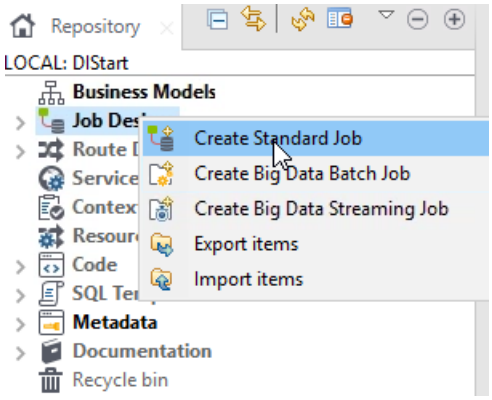
Jobs are created and stored in the **Repository**, in **Job Designs**.

1. CREATE A STANDARD JOB

   To create a Job:

   a. Right-click **Repository > Job Designs**.

   b. Select **Create Standard Job**. The **New Job** dialog box opens.



2. DEFINE THE JOB PROPERTIES

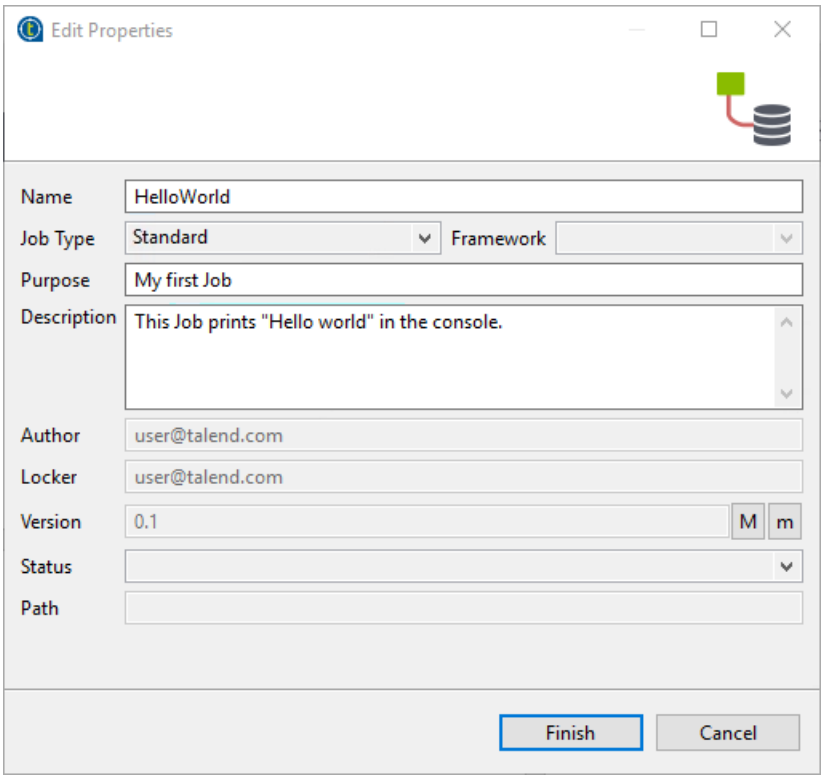
   Define the Job properties in the **New Job** dialog box:

   a. In the **Name** text box, enter *HelloWorld*.

   b. Fill-in the **Purpose** and **Description** text boxes.

   c. Leave the default values for the other parameters.

   d. Click **Finish**.

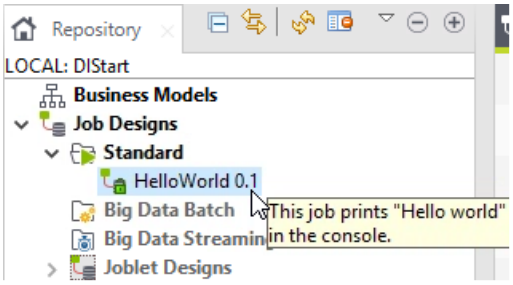


WARNING:
Avoid using common illegal filename characters. For example, *HelloWorld* is a valid name, but *Hello World* is incorrect due to the space between the words.

3. FURTHER EXPLANATIONS

   Notice the changes in the **Designer**, **Palette**, and rest of the views.

   New Jobs are automatically assigned version **0.1**, therefore, the Job you created is named "HelloWorld 0.1". A tab with the same name is open in the **Designer** and components are available in the **Palette**. You are ready to add components to your Job.



Adding components to your Job

The Job needs two components: a file containing the *Hello world!* string and a console to send the output to. To do this, you can use the **tFileInputRaw** and **tLogRow** components.

tFileInputRaw reads all data in a raw file and sends it to a single output column for subsequent processing by another component.

tLogRow displays data or results in the **Run** console to monitor data processed.

In the following, you explore two ways to add components to your Job: using the **Palette** and directly from inside the **Designer**.

1. ADD THE tFileInputRaw COMPONENT USING THE PALETTE

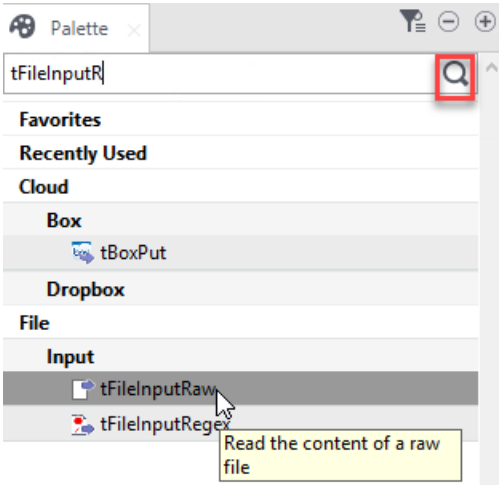
   To add a component using the **Palette**:

   a. In the Palette, look for the **Search** box.

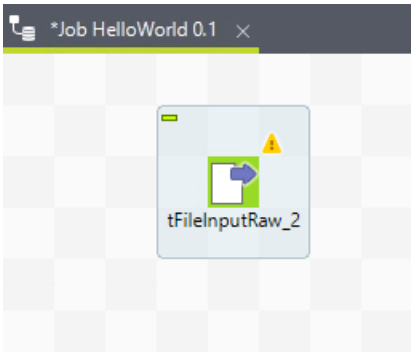
   b. Start entering *tFileInputRaw* in the **Search** box. To list components matching your research, press the **Enter** key (or click the **Search** button). Notice that a short description is displayed next to each component name when you hover the mouse over it.

   c. Select **tFileInputRaw** on the list

   d. Drag the component and place it in the **Designer**.



When you are finished, the Job looks like this:



TIP:
To follow Talend Studio design best practices, place components in the **Designer** from the upper left to the lower right corner, as the data flow generally proceeds from left to right and from top to bottom. Therefore, place your input components on the upper left part of the **Designer**.

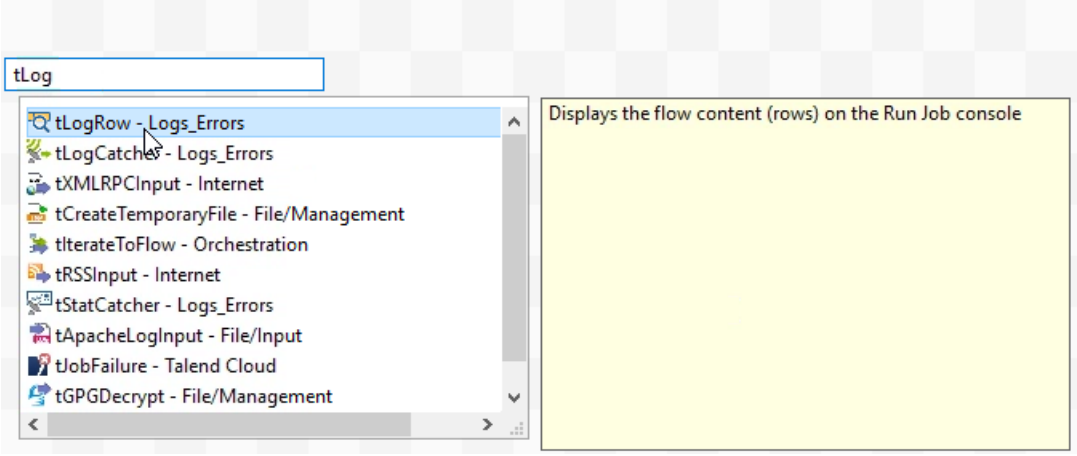
2. ADD THE tLogRow COMPONENT DIRECTLY FROM INSIDE THE DESIGNER

   To add a component directly from inside the **Designer**:

   a. Click anywhere in the **Designer**.

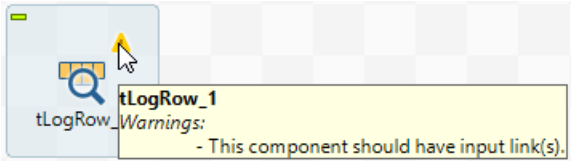
   b. Start entering *tLogRow*. As you type, components matching your search are listed. Notice that a short description is displayed next to each component name when you select it with the mouse.

   c. Double-click **tLogRow** on the list. The component appears in the workspace.



3. INSPECT THE WARNINGS

   Notice that the components are both displayed with a small warning icon. If you hover the mouse over the icon, a warning message says that the component should have inputs/outputs linked.



TIP:
Any time you need help on a component, just click on it and press the **F1** key.

Connecting components

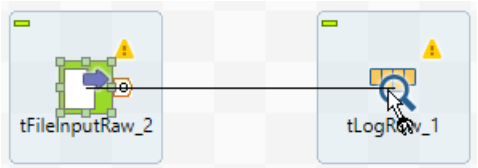
1. CONNECT THE COMPONENTS

   You can connect components with **rows**. These are links that allow data to flow between components.

   To connect tFileInputRaw and tLogRow components:

   a. Right-click in the middle of the input component (**tFileInputRaw**). The component is highlighted in a frame.

   b. Select **Row > Main**. A connection link is visible between the input component and mouse pointer. You can now link the output component.



   c. Click in the middle of the output component (**tLogRow**) to create the link. The arrows on the link show the direction of the flow.

NOTE:
Notice that once you connect the components, they are part of a same subJob which is represented by a blue rectangle.

Configuring components

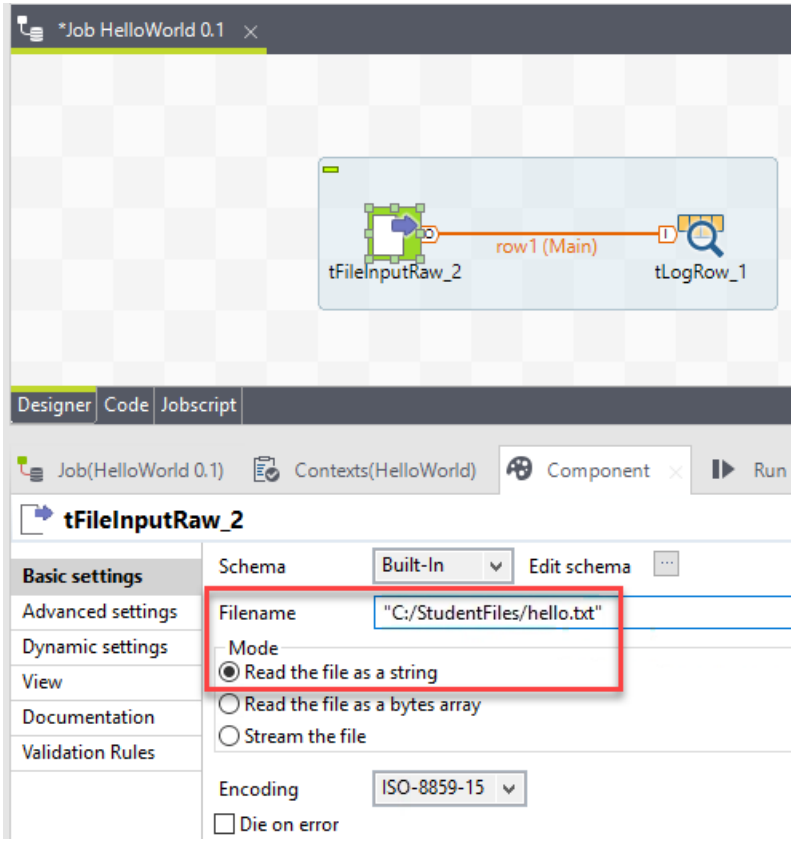
Remember that the **tFileInputRaw** component must read the "*Hello world!*" from a file, so you must first create a text file containing this string.

1. CREATE A TEXT FILE
    a. Open Notepad++ (on the Windows taskbar, click the **Notepad++** icon, or go to **Windows Start Menu > Notepad++**).
    b. Create a new file (**File > New**).
    c. Enter *Hello world!* .
    d. Go to **File > Save as**. Name the file *hello.txt* and save it under C:/StudentFiles/
    e. Close Notepad++.
2. DEFINE COMPONENT BASIC SETTINGS

    You can configure components using the **Component** view in the Designer. Depending on the component you use, the **Component** view displays different properties.

    To configure a component, follow these steps:

    a. Double-click in the middle of the **tFileInputRaw** component. The **Component** view opens.
    b. For **Filename**, click the **[...]** to select the file path *C:/StudentFiles/hello.txt*. Make sure the path you provide is the actual location where you saved your file.
    c. Select **Read the file as a string**

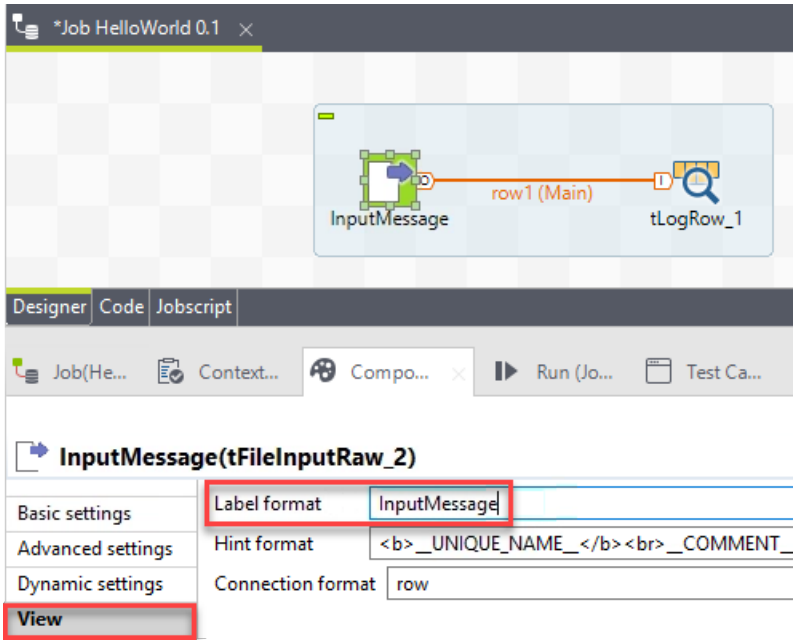


3. NAME THE COMPONENTS

   To name a component, in the **Component** view:

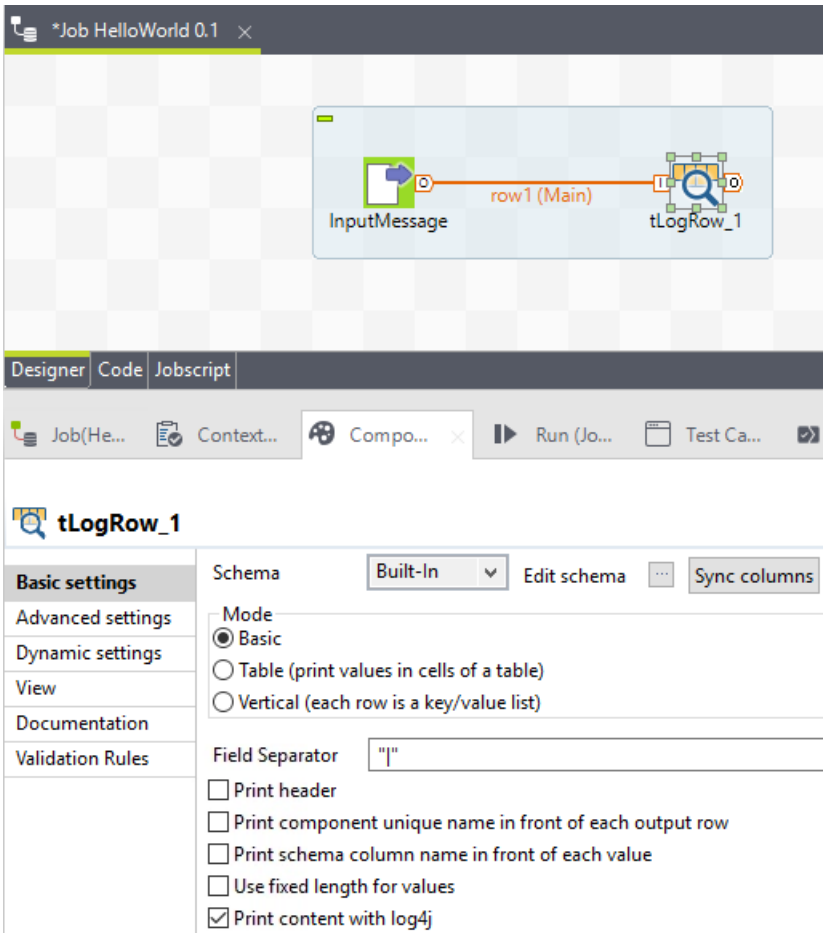
   a. Click the **View** tab.

   b. Replace the text in the **Label format** text box with *InputMessage*. This updates the component name in the **Designer**.



4. VERIFY THE tLogRow PROPERTIES

   To check the settings for **tLogRow_1**, double-click the component and look at the settings in the **Component** view:



5. SAVE THE JOB

   Notice that on the **Designer** tab, there is an asterisk (*) in front of your Job name (as you can see in the previous figure), which means the Job has unsaved changes. To save the Job, on the main toolbar, to the left, click the **Save** button ( ).

   TIP:
   You can also press **Ctrl + S** to save the Job.

Next step

You have created your first Job and now you are ready to run it.

Running a Job

Task outline

In this exercise, you learn how to run the Job you created and visualize the results in the console.

This exercise takes about 5 mins to complete.

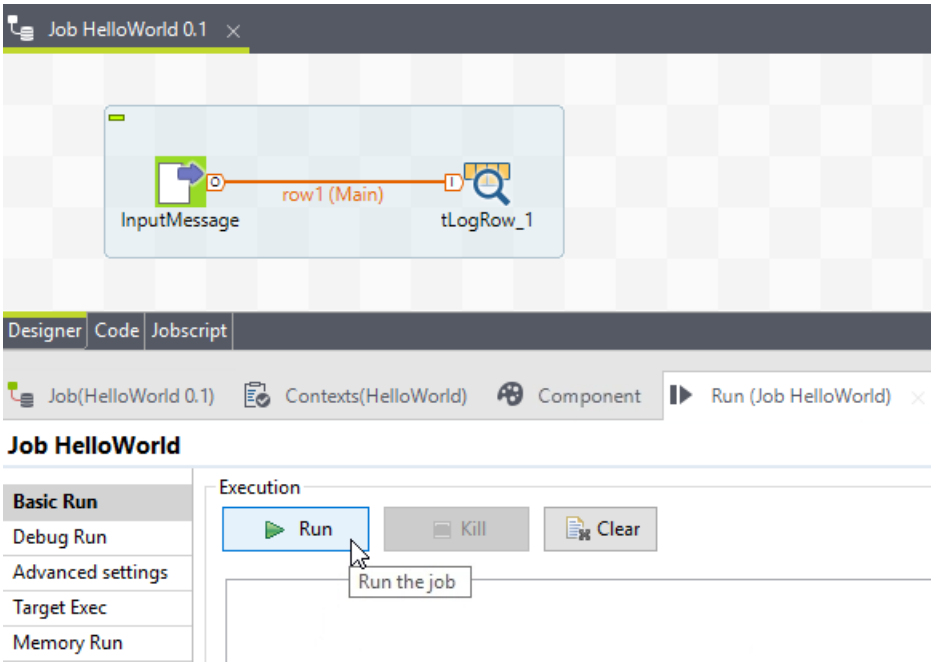
Running a Job

1. OPEN THE RUN VIEW

   To run the Job:

   a. In the **Designer**, open the **Run** view.

   b. Click **Run**.



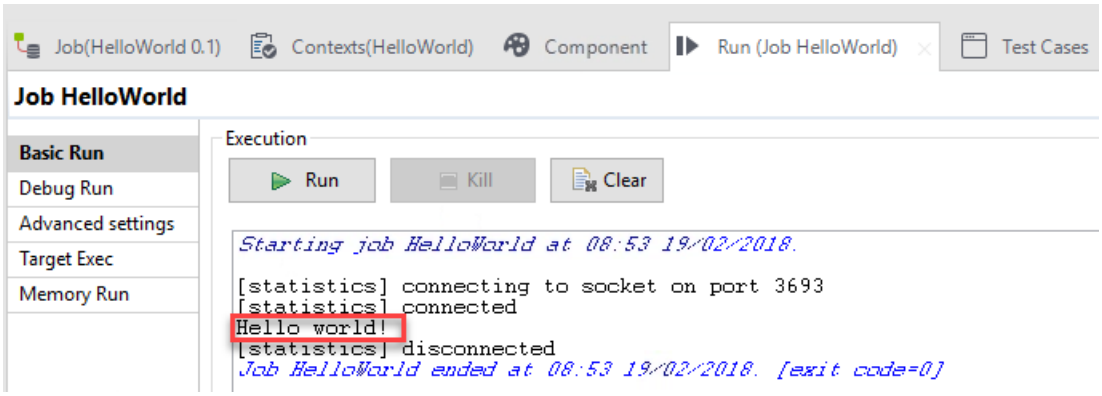
TIP:

You can also use the **Run** icon ( ) on the main toolbar to run the Job. When you have several Job tabs open in the **Designer**, make sure to select the correct Job before using the **Run** icon.

2. REVIEW THE JOB OUTPUT

   Talend Studio first saves and then builds the Job. This may take a few moments. Then the Job is executed and the "Hello world!" message is displayed in the console in the **Run** view. The Job ends with an exit code = 0 which means that the Job

executed correctly.



Congratulations! You have built and run your first Talend Studio Job.

Next step

You have almost finished this section. Time for a quick review.

Review

In this lesson, you learned the basics required to create projects using Talend Studio.

You created your first Job, learned how to execute it and visualize the results in the console. You experienced different methods of adding and linking components, as well as how to configure them. You used the **Palette** to search for components, and you opened the component documentation help page.

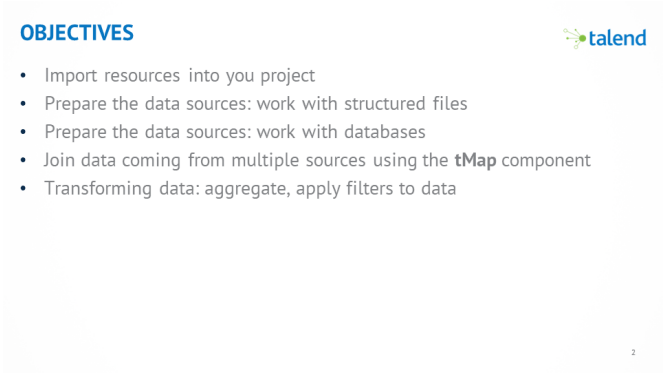
You will have the opportunity to use other components and create more complex Jobs in the following lesson.

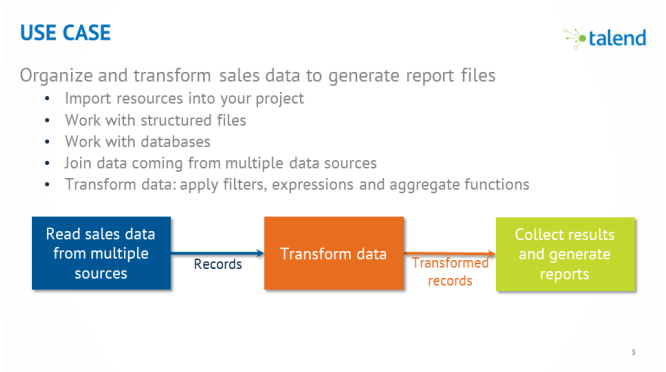
LESSON 3

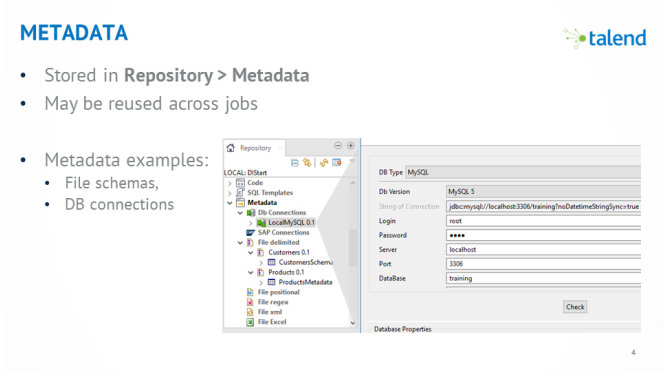
Using Talend Studio to Design a Data Integration Project

This chapter discusses:

Concepts





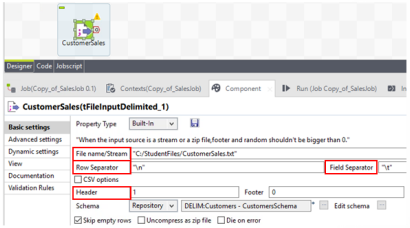


Metadata is stored in the **Repository > Metadata** folder and may be reused across Jobs. Indeed, some resources are good candidates for reuse, especially when they do not change from one project to the next, such as database connection settings or metadata (for example, a file schema that holds a file structure).

WORK WITH STRUCTURED FILES

Reading from a structured file

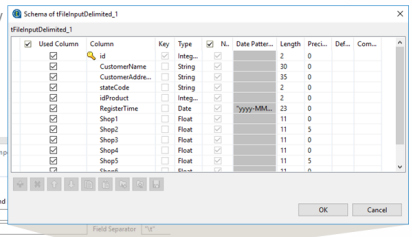
- Use **tFileInputDelimited**
- **File name** to read from
- File structure:
  - **field/row separator** and
  - **number of header lines**



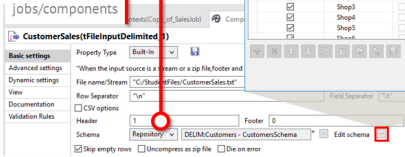
5

WORK WITH STRUCTURED FILES

- The **schema** defines how data is interpreted



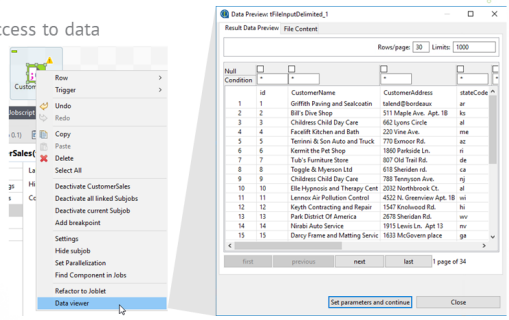
Repository schema may be reused across jobs/components



6

DATA VIEWER

Quick access to data

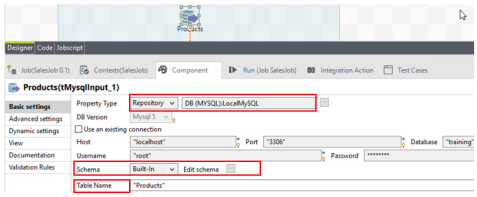


7

WORK WITH DATABASES

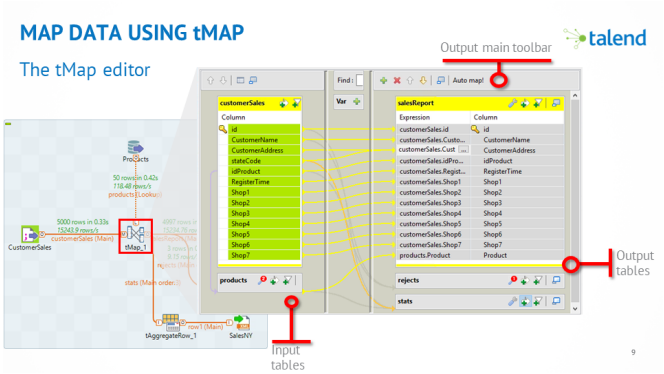
Read data from a database

- Use **tMySqlInput**
  - Connect to the database using the Db connection metadata
  - Edit schema and table name

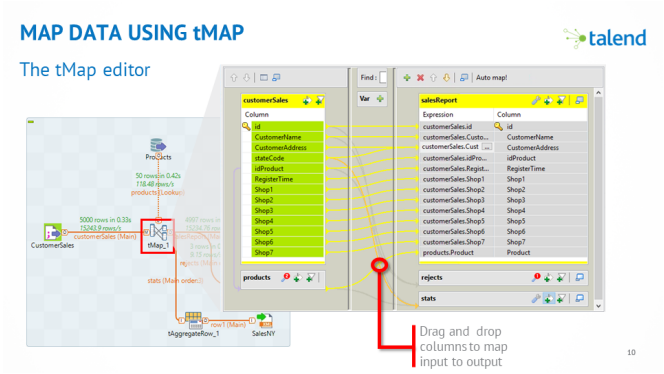


8

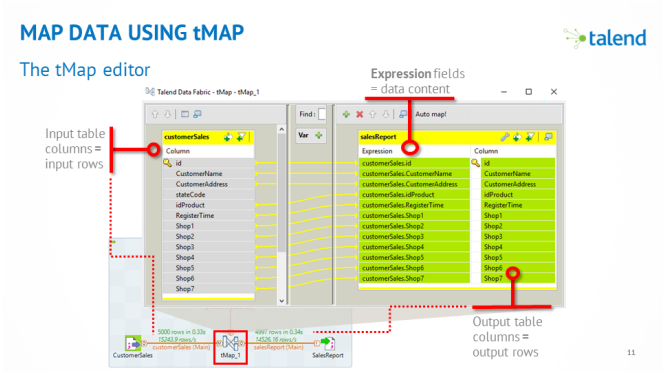
To read data from a database, you use a t**MysqlInput** component. You need to configure this component by specifying the database connection properties and database schema.



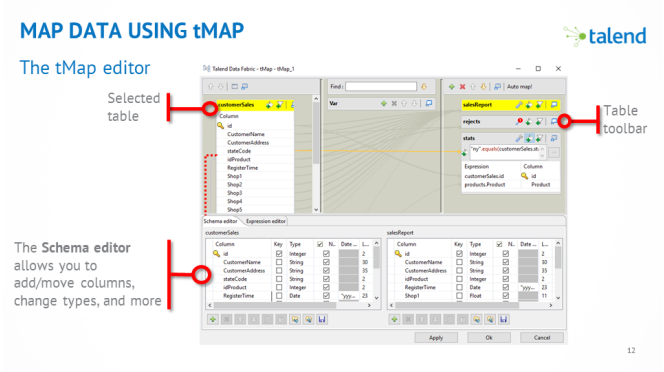
The **tMap** component transforms and routes data from single or multiple sources to single or multiple destinations. Input and output are interpreted as tables.



The tMap editor allows you to map data easily from input to output by drag and drop. The arrows indicate the mapping of columns.



The columns of the input table are an exact match of the rows entering the tMap component. In the output table, the **Expression** fields hold the matched data content, and the **Column** contains the data labels. Output rows are an exact match of the output table columns.



JOIN DATA SOURCES USING TMAP

talend

- Use **tMap** to join and transform multiple inputs to multiple outputs
  - **tMap** is often part of a solution to various data challenges

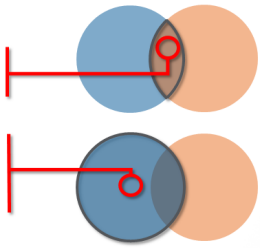
13

JOIN DATA SOURCES USING TMAP

talend

Join Types

- A **join** combines rows from two or more tables
  - An **inner join** returns rows that only appear in both tables
  - A **left outer join** (or **left join**) returns all records from the left table with NULL column values if no match is found in the right table

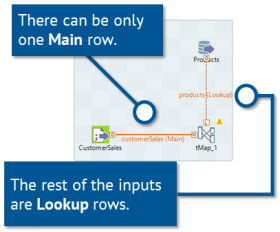


14

JOIN DATA USING TMAP

talend

- **tMap** takes multiple data sources

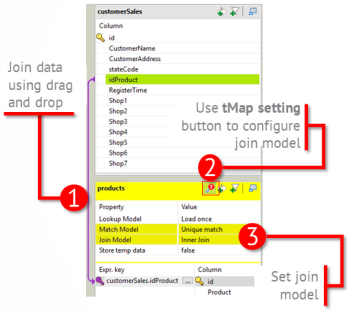


1. Define multiple data sources
2. Configure join
3. Configure a *rejects* table

15

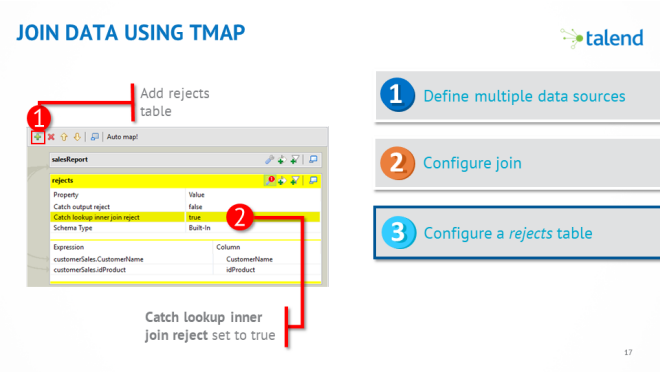
JOIN DATA USING TMAP

talend

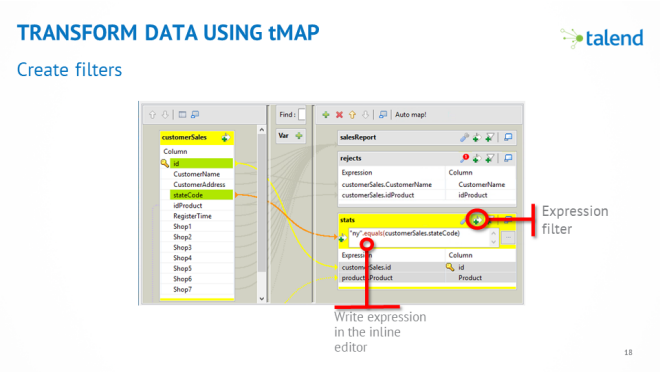


1. Define multiple data sources
2. Configure join
3. Configure a *rejects* table

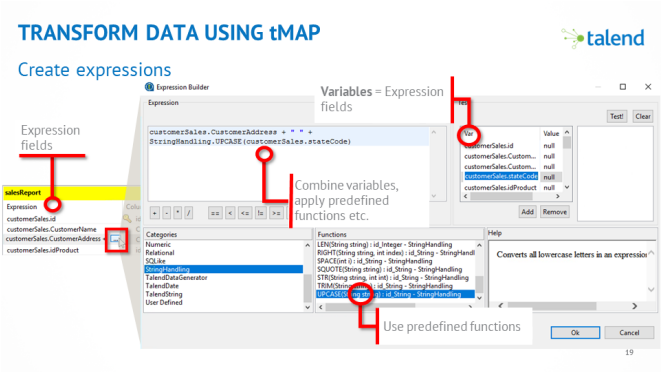
16



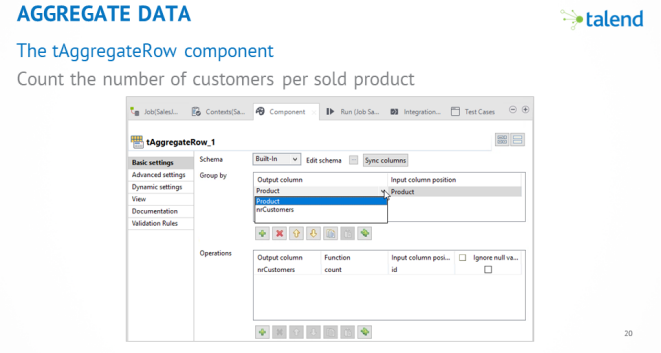
Configure a *rejects* table to catch the columns that did not match the inner join condition.



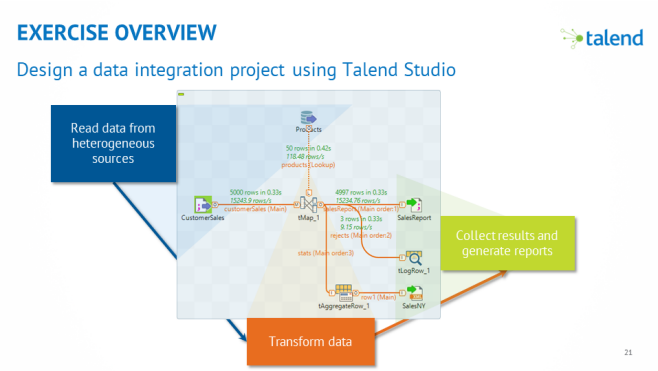
Use the inline editor to create simple expressions.



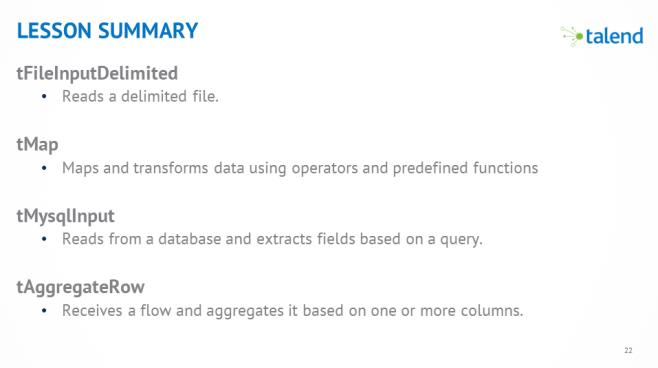
To easily create complex expressions, you use the **Expression builder**, which provides editing tools and a library of functions grouped by categories. In the editor, the **Variables** represent your Expression fields. Copy/paste variables to use them in expressions.

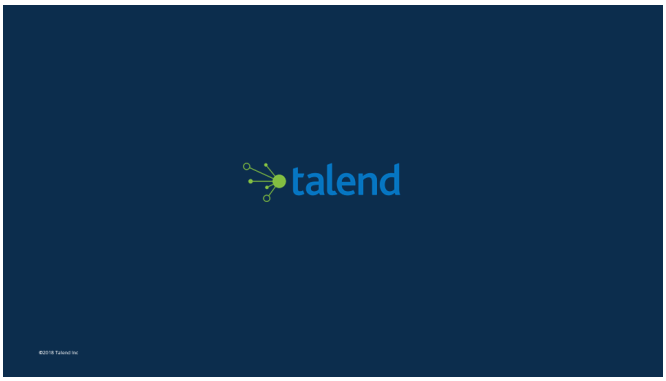


The **tAggregateRow** component receives a flow and aggregates it based on one or more columns. You can use it to create GROUP BY statements. In this example, you group data by product name and count the number of customer sales per product. You apply transformations on columns defined by **Input column position** and send the output to the columns defined by **Output column**.



In this exercise, you learn how to design a data integration project using Talend Studio. Your role is to analyze sales data from a CRM system and output some reports that will help you prepare the next marketing campaign. You create a Job that joins data from the customer sales file and the products database and processes the results to extract statistics for the sales reports.





Overview

Use case

In this lesson, you learn how to design a data integration project using Talend Studio. You follow the classical phases of a data integration chain: extract data from heterogeneous sources, transform it and load the results in the target storage.

For this exercise, your role is to analyze sales data from a CRM system and output some reports that will help you prepare the next marketing campaign. You have been asked to focus on a specific list of products targeted by the marketing policy in your company.

The data you have at hand is:

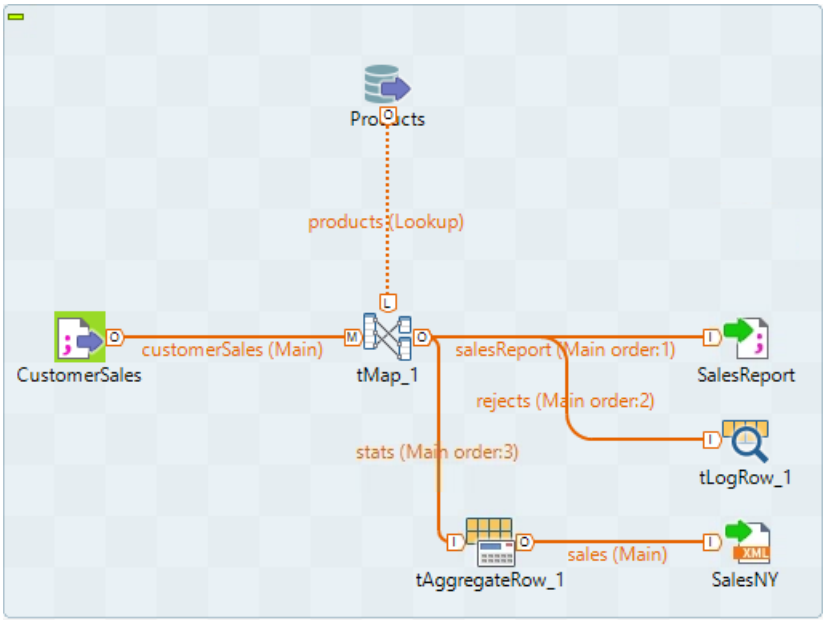
- A list of targeted products that you can read from an existing Products database
- Sales data from a CRM system stored in the CustomerSales.txt file

You must create a Job that joins data from the customer sales file and the products database and processes the results to extract statistics for the sales reports.

As a result, you want to:

- Generate a global sales report
- Generate a report containing statistics on the number of sold products for the city of New York
- Display in the console products that were bought by customers but not targeted by the current marketing campaign

When you finish, the Job looks like this:



Objectives

After completing this lesson, you will be able to:

- Import resources into your project
- Work with structured files
- Work with databases
- Join data coming from multiple data sources
- Process data flows

Next step

The next step is to prepare the data sources.

Importing External Resources into Your Project

Task outline

In this section, you learn how to import external resources into your project. Indeed, some resources are good candidates for reuse, especially when they do not change from one project to the next, such as database connection settings or metadata (for example, a file schema that holds a file structure).

For your convenience, all the metadata you need in this lesson has already been created. However, you need to import it into your project.

This exercise takes about 5 mins to complete.

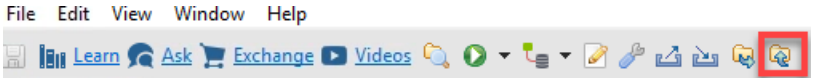
Importing items into your project

Before you start this exercise, copy the content of the **C:/IntroTalendStudio** folder to **C:/StudentFiles** on your VM. The folder contains the files CustomerSales.txt, LoadProducts.zip and refProducts.csv. You are now ready to import items into your project.

The items to import are:

- The **LocalMySQL 0.1** database connection settings for the connection to the local MySQL database instance
- The **LoadProducts 0.1** Job, which allows you to create the Products table
- The **Customers 0.1** and **Products 0.1** file schemas, which hold the structure of the Customers.csv file and schema of the Products database

To import resources from a different Studio project, you can use the Import items button on the main toolbar.



The metadata to import is located at: C:\StudentFiles\LoadProducts.zip.

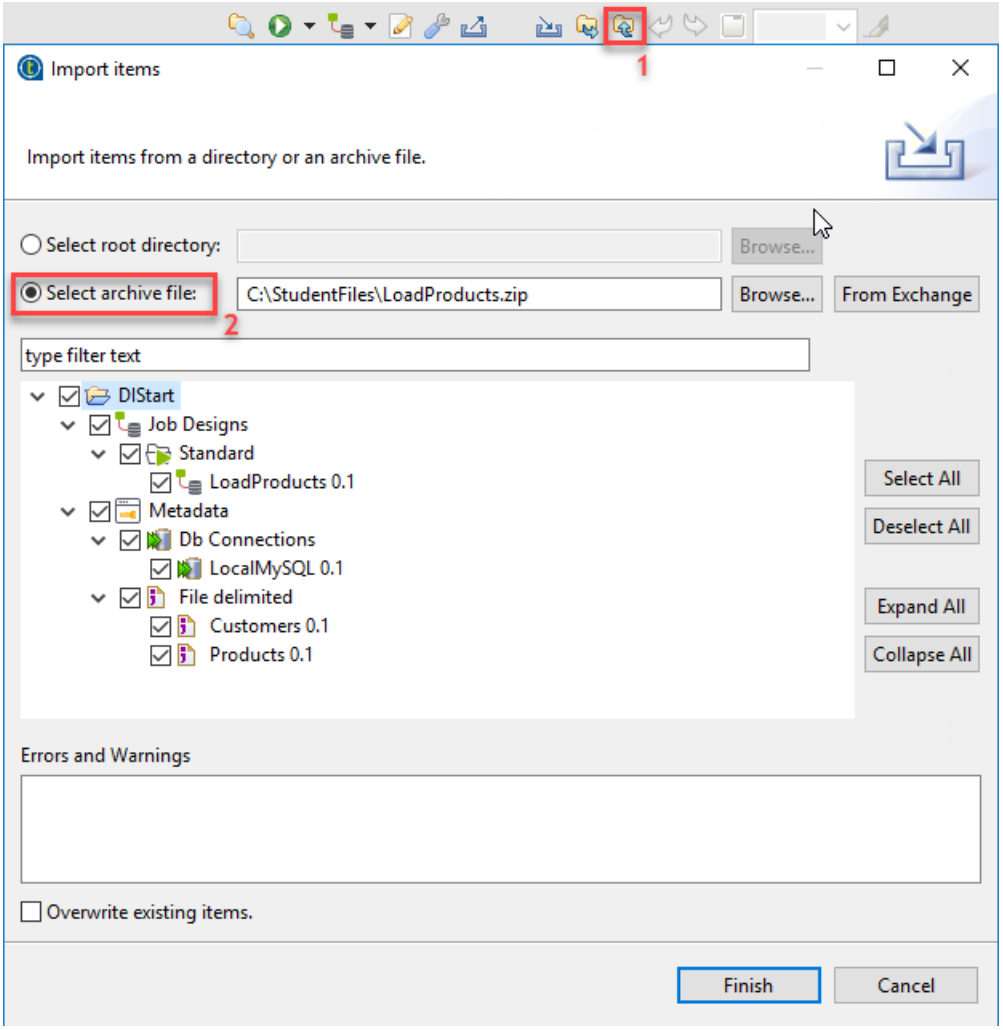
1. IMPORT THE METADATA

   To import the metadata:

   a. Click the **Import items** icon n the main Studio toolbar.

   b. In the dialog box that opens, select the check box **Select archive file**

c. Click **Browse** and select the zip file located at *C:\StudentFiles\LoadProducts.zip*.

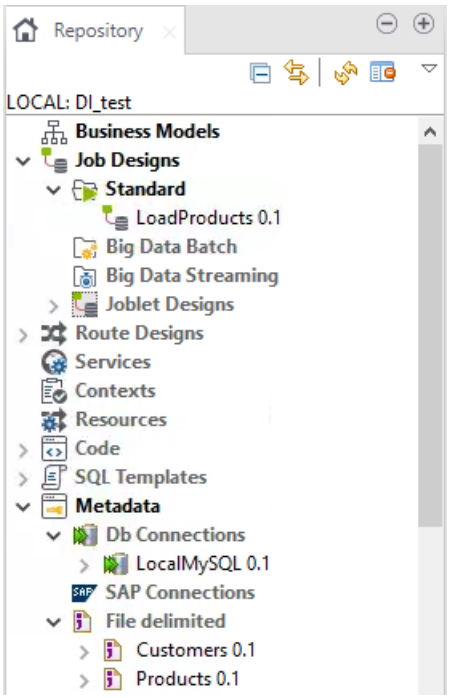


d. Select the elements to import: **LoadProducts 0.1**, **LocalMySQL 0.1**, **Customers 0.1** and **Products 0.1**

e. Click **Finish**.

NOTE: When you import the metadata into your project, it is stored in the **Repository > Metadata** folder. Jobs are stored in the **Job Designs > Standard** folder.

When you finish this task, the **Repository** for your project looks like this:



Next step

You have imported the metadata you need and are ready to prepare the data sources.

Preparing Data Sources: Working with Structured Files

Task outline

In this lesson, you learn how to read and interpret data coming from a structured file (in this case, the CustomerSales.txt file from the CRM system).

This exercise takes about 15 minutes to complete.

Reading data from a structured file

To read data from a structured file (for example, CustomerSales.txt) you use a **tFileInputDelimited** component. You configure this component by specifying where the file is located and how the data inside is organized.

The **tFileInputDelimited** reads a delimited file row by row, splits them into fields, and sends the fields, as defined in the schema, to the next component.

Once the data is read from the file, you must give a meaning to it so that you can process it further. You can do this by assigning a schema to your component.

A schema holds information about the way data is interpreted, and is therefore considered metadata. A schema defines the columns/rows to be processed and passed on to the next component.

For your convenience, all the schemas you need for this lesson have been already created. Before continuing, make sure you have imported all necessary metadata into your project, as suggested in Import "Existing Resources into Your Project".

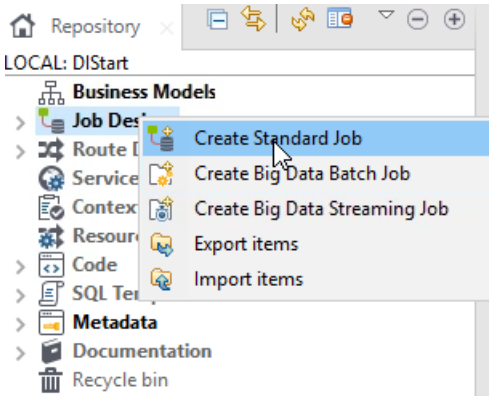
To get started, you must create a Job.

1. CREATE A JOB

   To create a Job:

   a. Right-click **Repository > Job Designs**.

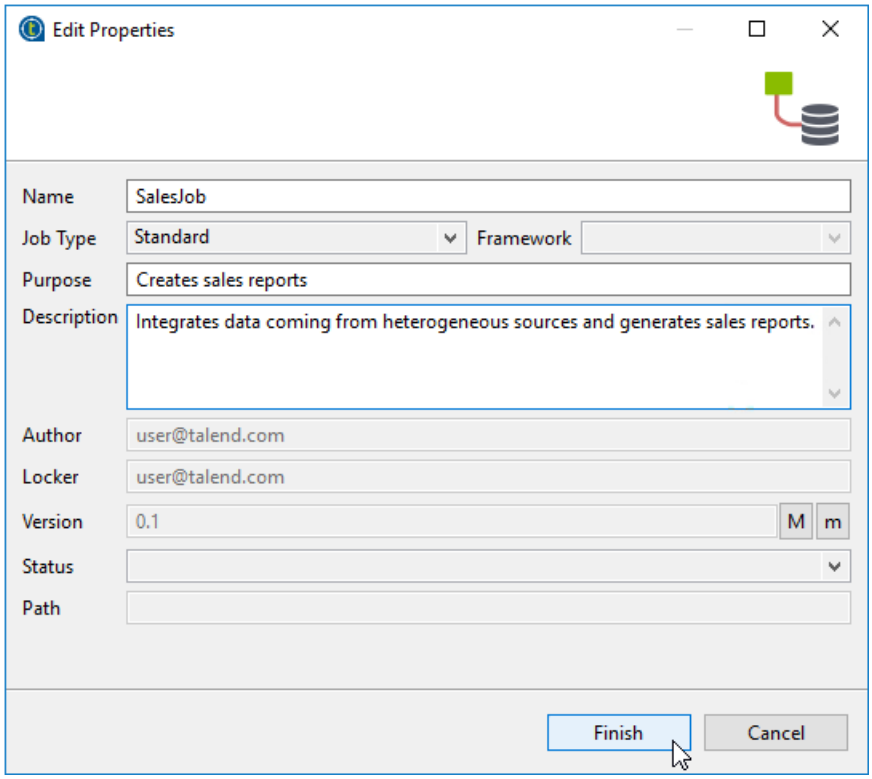
   b. Select **Create Standard Job**. The **New Job** dialog box opens.



   c. In the **Name** text box, enter *SalesJob*.

   d. Fill-in the **Purpose** and **Description** text boxes.

   e. Click **Finish**.



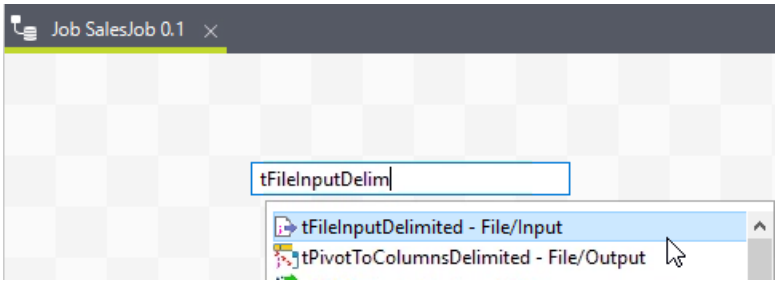
The Job opens in the Designer.

TIP:
When creating a Job, only the name is mandatory, but it is a good practice to complete the remaining fields to help document the Job.

2. ADD A tFileInputDelimited COMPONENT TO READ A STRUCTURED FILE

   To add a **tFileInputDelimited** component:

   a. Click anywhere in the **Designer**
   b. Start entering *tFileInputDelimited*. As you type, components matching your search term appear.



   c. Double-click **tFileInputDelimited** on the list. The component appears in the Designer.

3. SET BASIC FILE CONFIGURATION

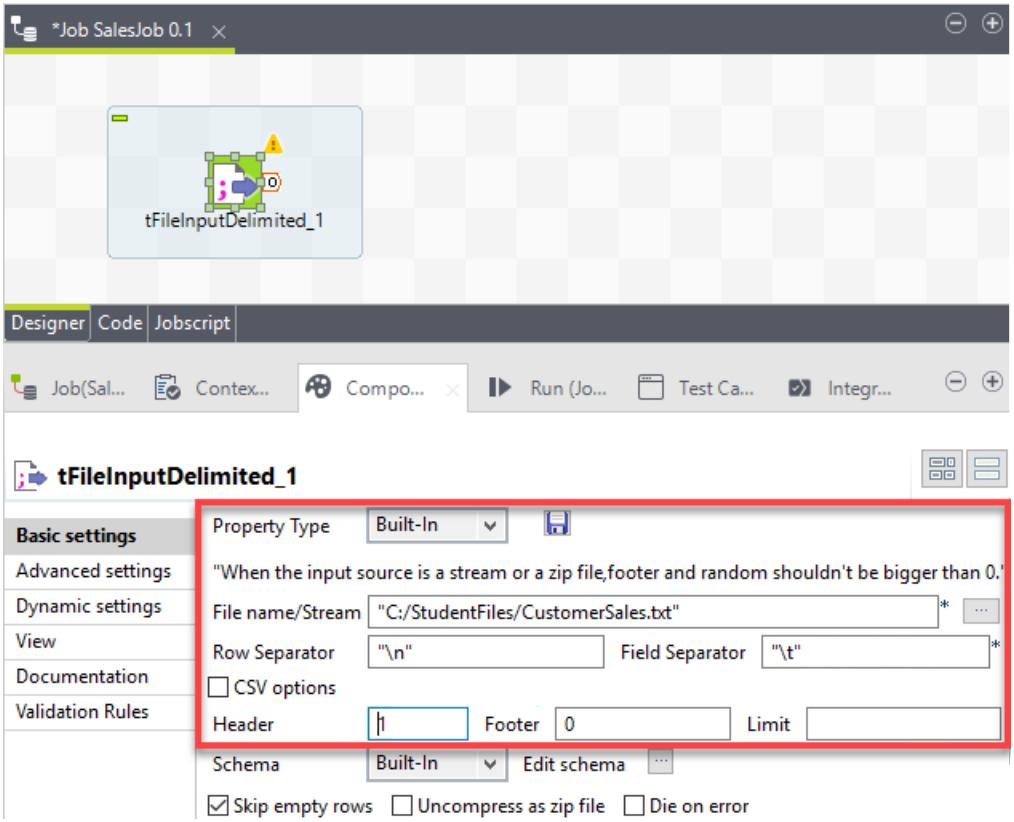
   To configure a **tFileInputDelimited** component, you must specify the name of the file to read from, and its structure.

   a. Double-click the **tFileInputDelimited_1** component to open the **Component** view.
   b. To set the **File name**, click **[...]** and select the *C:/StudentFiles/CustomerSales.txt* file path.

c. Leave the default **Row Separator** as *"\n"*.

d. Set the tab character as **Field Separator**: *"\t"*, to specify that the file contains tab-separated values.

e. In the **Header** text box, enter *1*. This means the first line in the file is ignored, which is what you want because this line does not contain actual data.

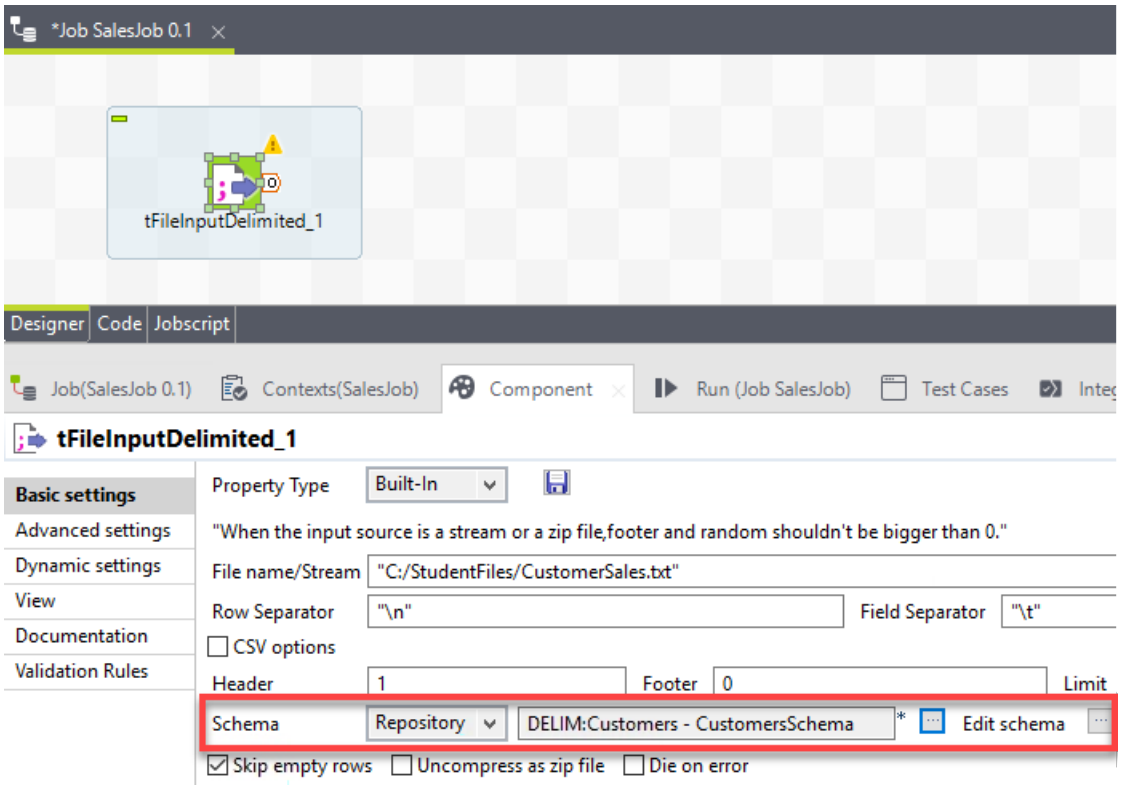


You are almost finished. You just need to configure the schema, which will let you process data further along.

4. CONFIGURE THE SCHEMA

a. For **Schema**, select *Repository*.

b. To its right, click **[...]** and select **Metadata > File delimited > Customers 0.1 > CustomersSchema**.



TIP:

In which case to use **Built-In** and **Repository**:

Built-In: Information is stored locally in the Job. Use Built-in for information that you need once or very rarely.

Repository: Information is stored in the repository. Use Repository for information that you need repeatedly in multiple components or Jobs.

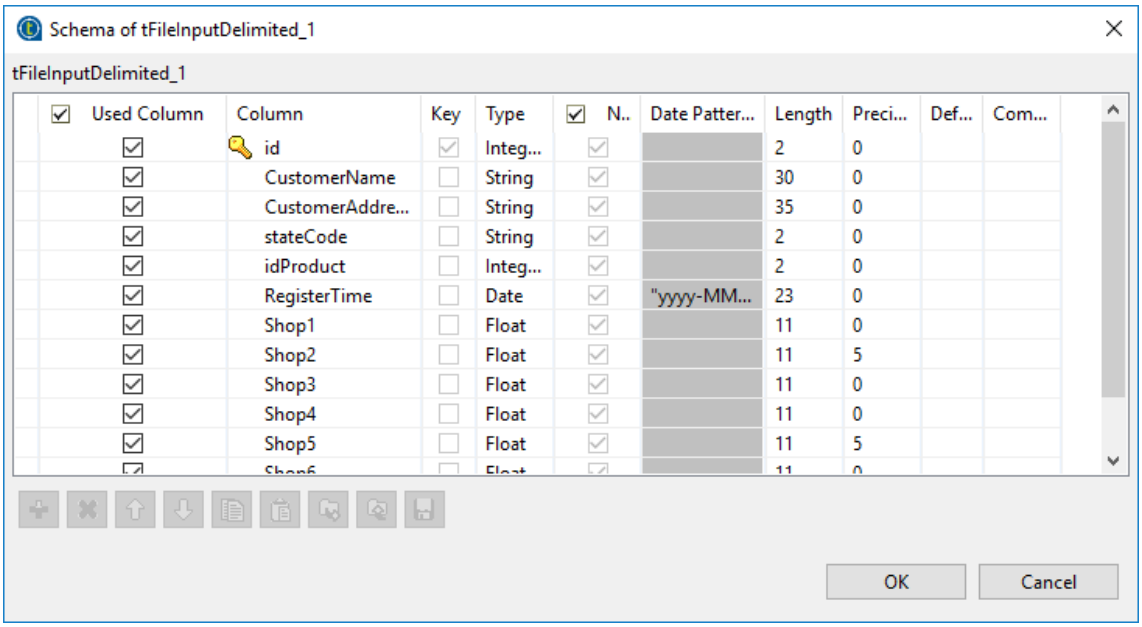
5. VIEW THE SCHEMA

To visualize the schema:

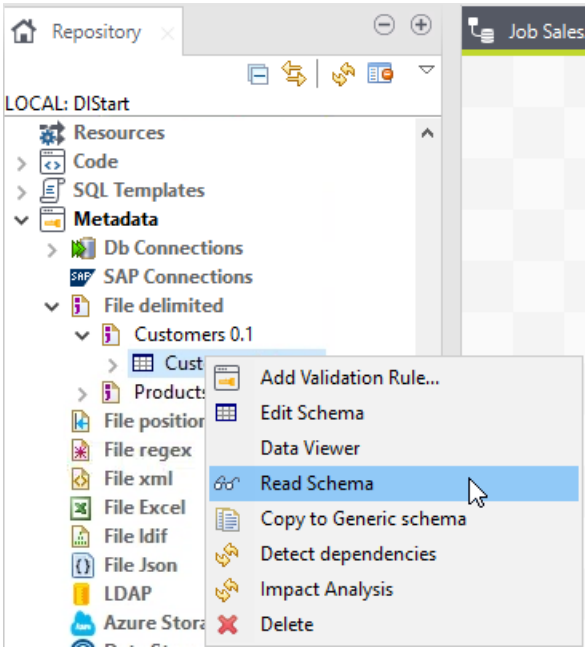
a. To the right of **Edit schema**, click **[...]**. A dialog box opens.

b. Select **View schema** and click **OK**.

A dialog box showing the schema opens. Take a few moments to look at the schema. You can see that it is composed of columns with a given type and other related properties. Close the window.



You can also view the schema right in the **Repository > Metadata**. Locate the schema you are interested in, right-click, and select **Read Schema**.



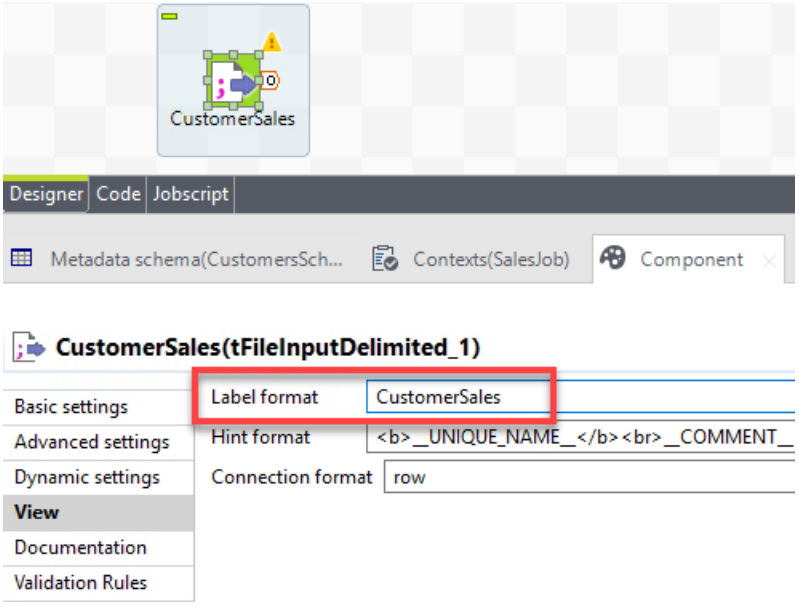
6. NAME THE COMPONENT

   To name the component:

   a. Click the **View** tab in the **Component** view.

   b. Replace the text in the **Label format** text box with *CustomerSales*. This automatically updates the component

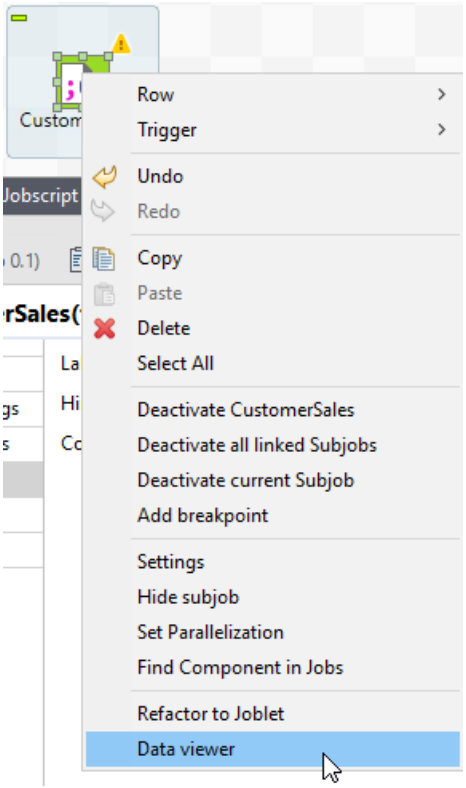
name in the design workspace.



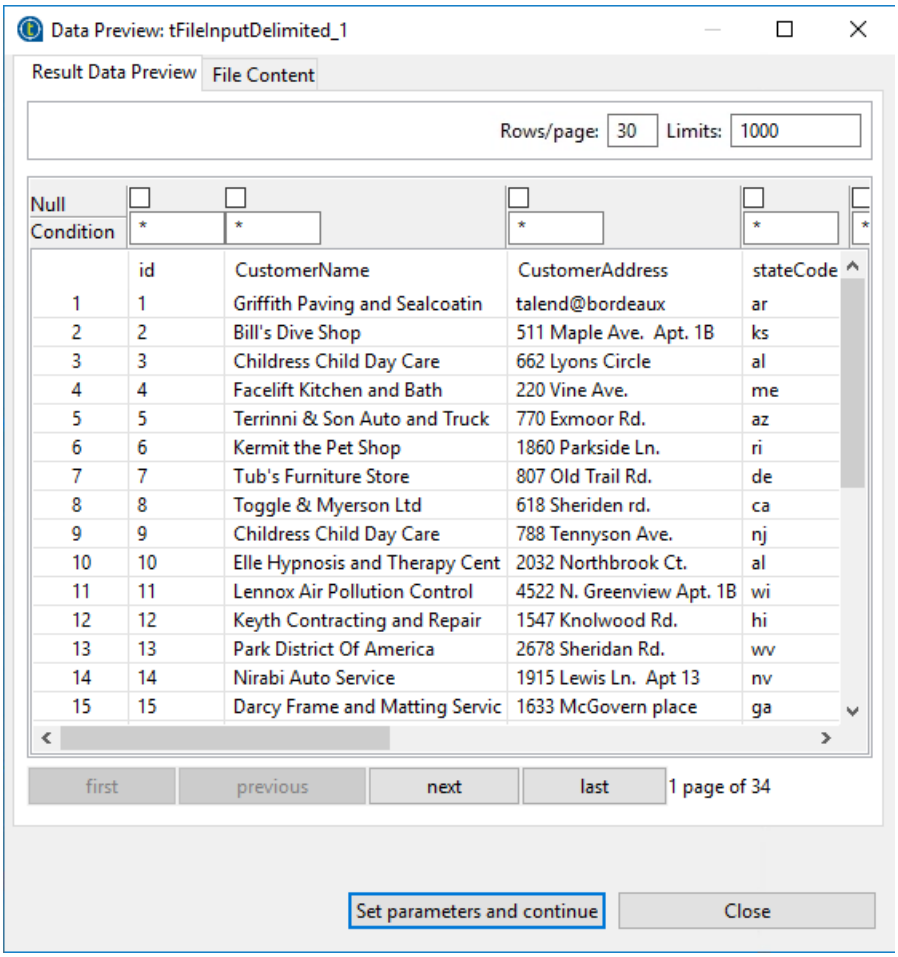
7. VISUALIZE DATA USING THE DATA VIEWER

   To visualize data using the Data viewer:

   a. Right-click the **CustomerSales** component
   b. Select **Data viewer**.



A window displays a preview of the data retrieved from the CustomerSales.txt file. The **File Content** tab displays the raw content of the file. Click **Close**.



Next step

You have set a **tFileInputDelimited** component to read data from a structured file, and now you are ready to read data from a database.

Preparing Data Sources: Reading Data from Databases

Task outline

In this lesson you learn how to read and interpret data from the Products table. You run the Job you imported to create the Products table and data, and then you can start using it.

This section takes about 15 minutes to complete.

Reading data from a database

First you must create the **Products** table. The LoadProducts 0.1 Job you imported allows you to create the table and add some data to it. After you run this Job, the table is created inside the existing "training" database.

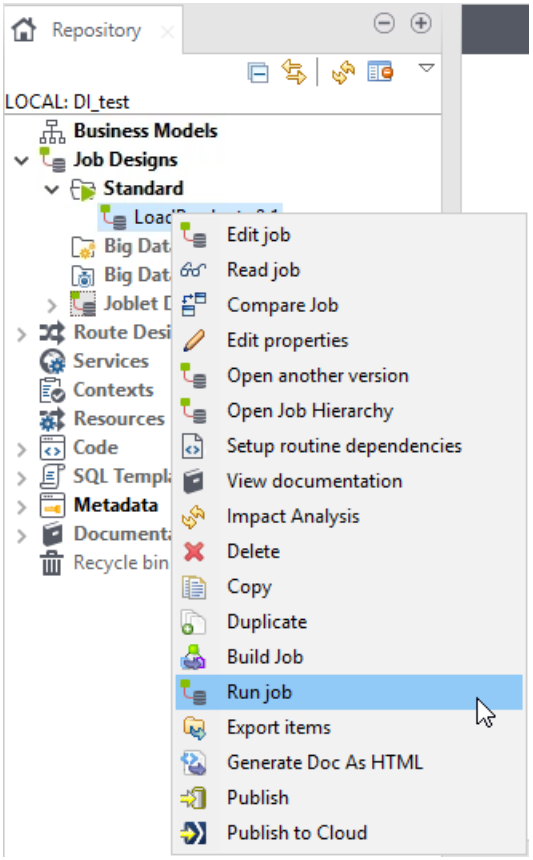
To read data from a database, you use a **tMysqlInput** component. You must configure this component by specifying the database connection properties and database schema. The **tMysqlInput** component executes a DB query with a strictly defined order which must correspond to the schema definition. Remember that you already imported this metadata into your project; there is no need to re-create it. You only need to assign it to the tMysqlInput component.

1. CREATE THE PRODUCTS TABLE AND DATA

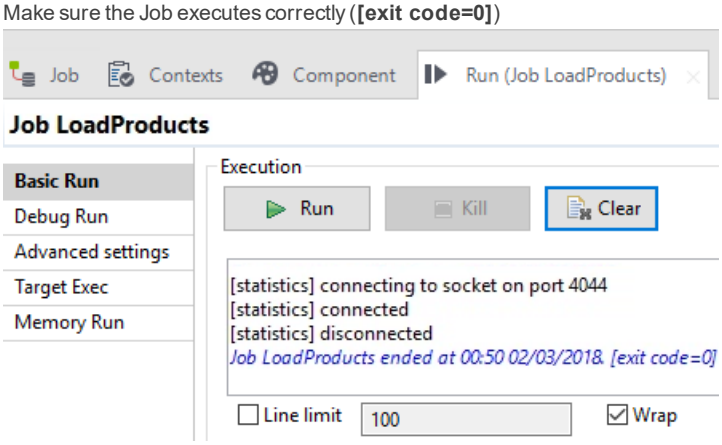
   To create the Products table, run the LoadProducts 0.1 Job:

   a. In **Repository > Job Designs > Standard**, right-click **LoadProducts 0.1**

   b. Select **Run Job**.



c. Make sure the Job executes correctly (**[exit code=0]**)



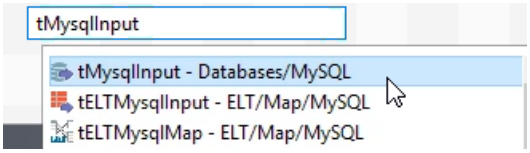
The Products table has been created.

2. ADD A tMysqlInput COMPONENT TO READ DATA FROM A DATABASE

   To add a **tMysqlInput** component:

   a. In the **Designer**, click in the space to the right of the **CustomerSales** component.

   b. Start entering *tMysqlInput* to list matching components.



   c. On the list, double-click **tMysqlInput**. The component appears in the Designer.

3. CONFIGURE THE tMysqlInput COMPONENT

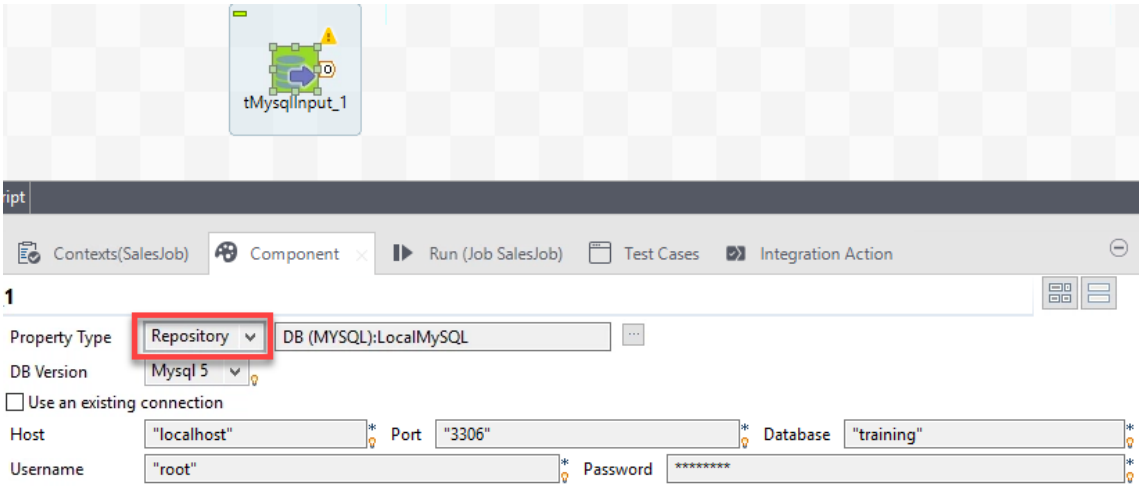
   Configure the tMysqlInput component using the LocalMySQL database connection:

   a. Double-click in the middle of **tMysqlInput_1** to open the **Component** view.

   b. For **Property Type**, select *Repository*. This allows you to reuse existing metadata.

   c. Click **[...]** and select **Metadata > Db Connections > LocalMySQL 0.1**. Click **OK**. Notice that the database connection properties (host, database, username) are automatically filled in.

   d. For **DB Version**, select *Mysql 5*.



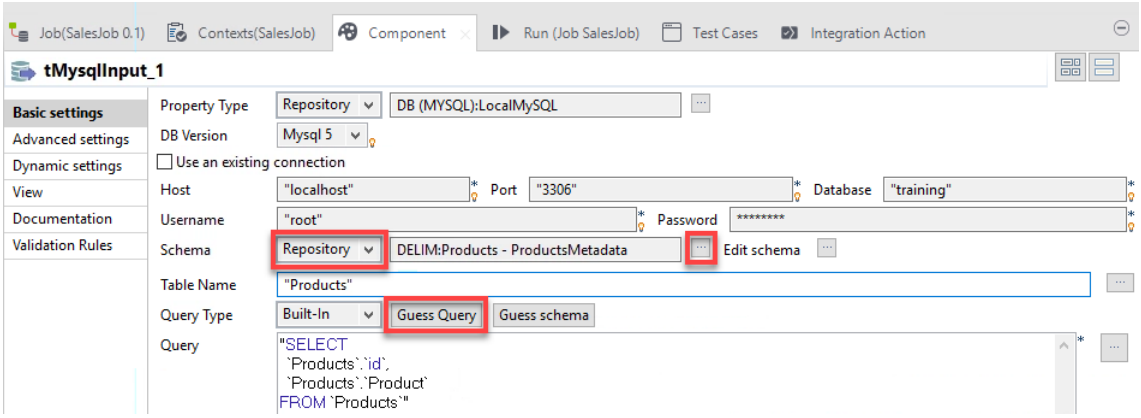
4. SET THE COMPONENT SCHEMA

   a. For **Schema**, select *Repository*.

   b. To its right, click **[...]** and select **Metadata > File delimited > Products 0.1 > ProductsMetadata**

   c. In the **Table Name** text box, enter *Products*

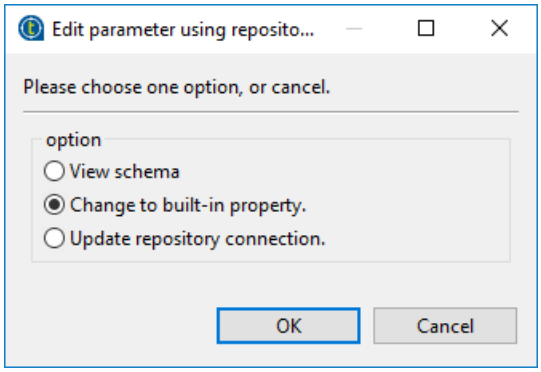
   d. Click the **Guess Query** button. This generates a default query that retrieves all columns from the table, which is exactly what you want.



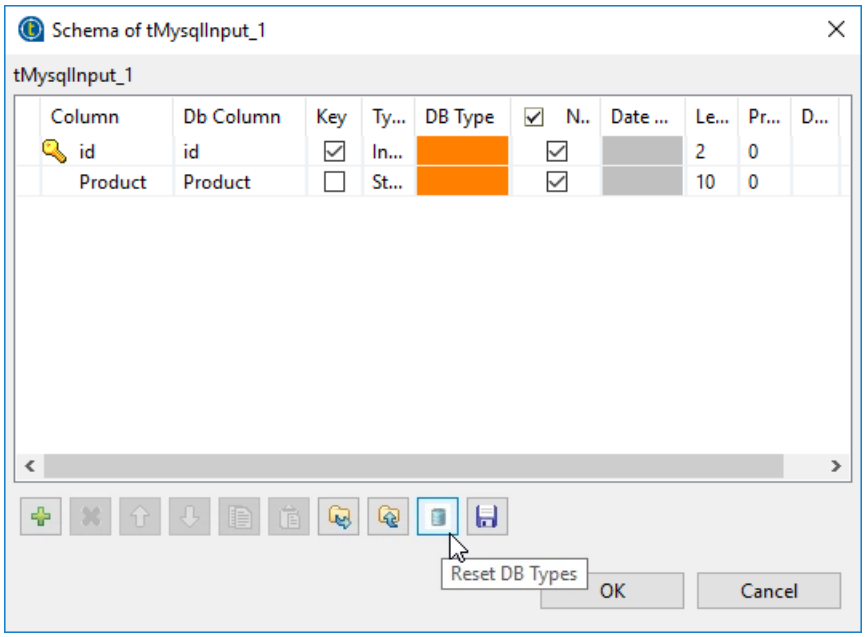
5. EDIT THE SCHEMA

Note that the **tMysqlInput_1** component has a warning that the schema database type (column **DB Type**) is not correct. This column in the schema defines the type of rows this component sends to the component connected to it. Therefore, you must specify a type for this column.

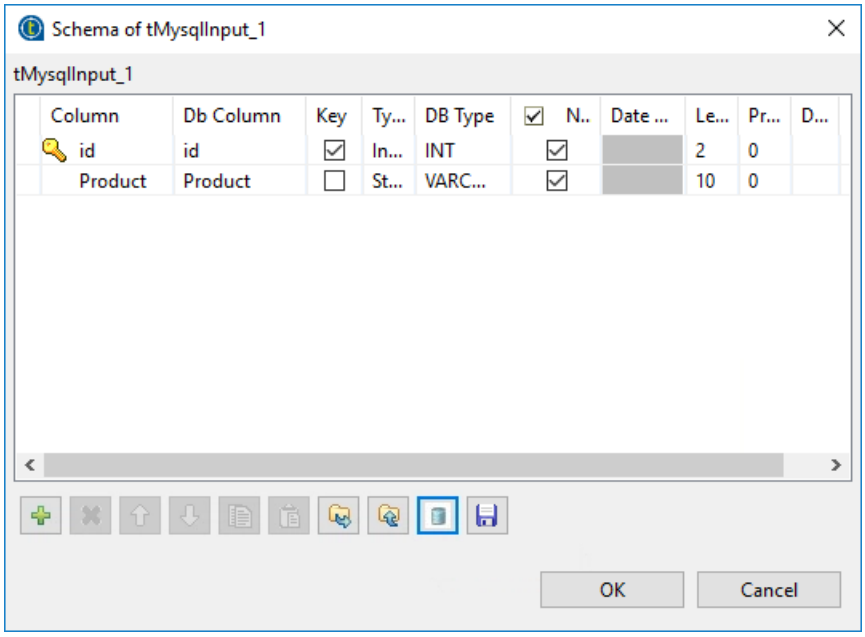
   a. To the right of **Edit schema**, click **[...]**. A dialog box opens with schema editing options.



   b. Select the radio button for **Change to built-in property**. This allows you to work on a local copy of the schema rather than the shared version stored in the repository. Click **OK**. A dialog box opens displaying the component schema. Notice that the **DB Type** column is empty and highlighted in orange.



c. Click the **Reset DB Types** button as shown in the screenshot. The column types are automatically filled in. Click **OK**.

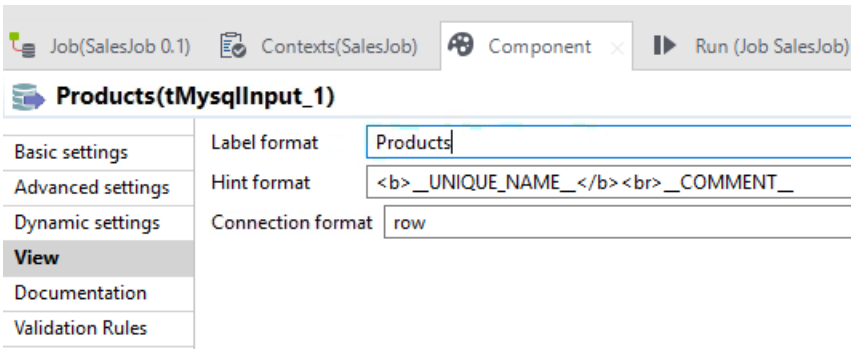


6. NAME THE COMPONENT

   To name the component:

   a. In the **Component** view, select the **View** tab.

   b. Replace the text in the **Label format** text box with *Products*. This automatically updates the component name in the Designer.

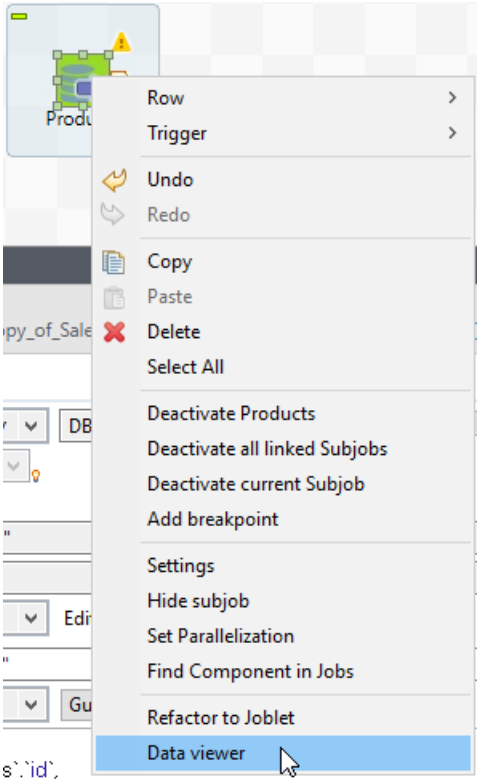


1. VISUALIZE DATA USING THE DATA VIEWER

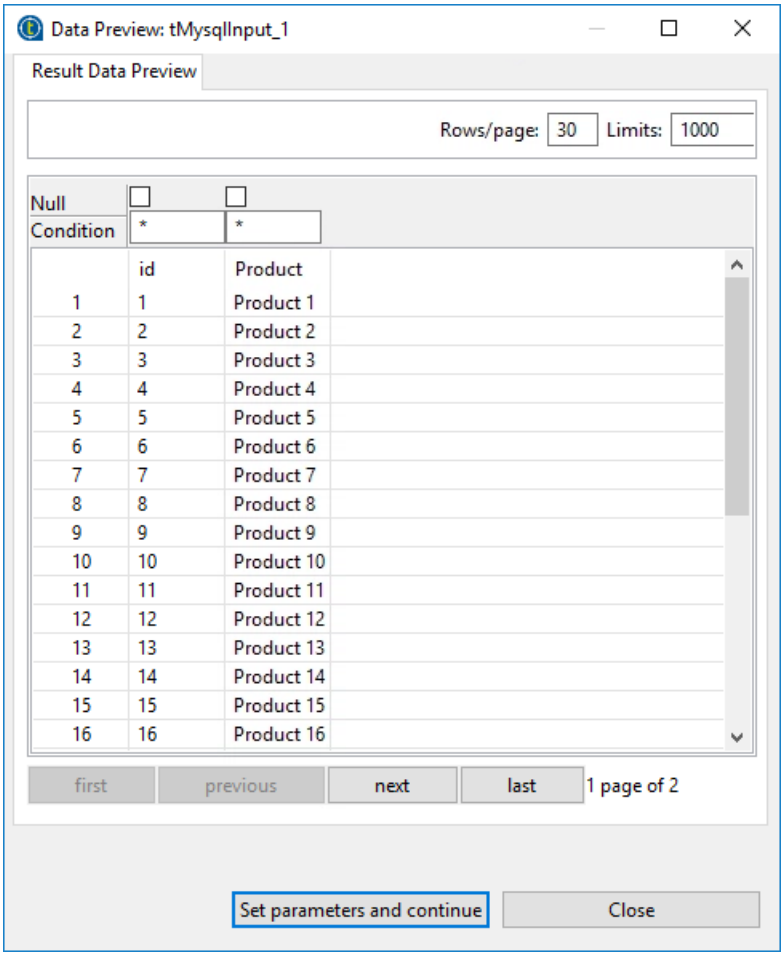
   To visualize data using the **Data viewer**:

   a. Right-click in the middle of the **Products** component.

   b. Select **Data viewer**.



A window opens displaying a preview of the data retrieved from the Products table. Click **Close** when you are finished.



Next step

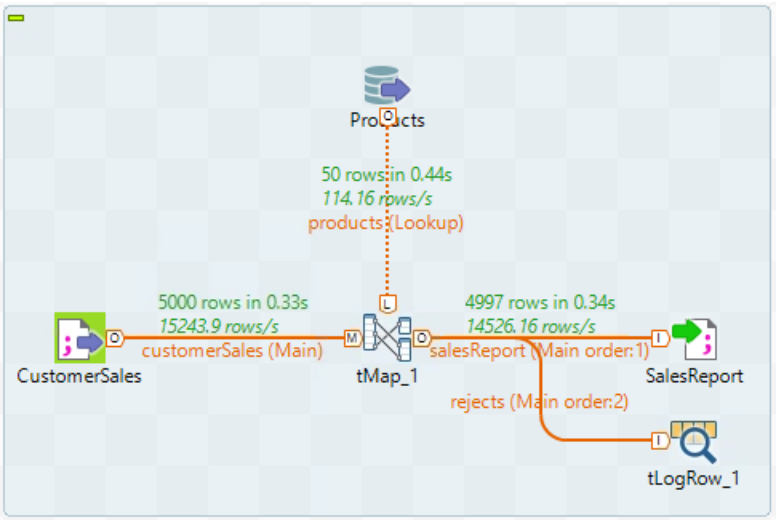
Earlier, you configured a component to read data from a structured file, and in this lesson, you learned to read data from a database using the **tMysqlInput** component. You are now ready to join data coming from multiple sources.

Joining Data From Multiple Data Sources Using tMap

Task outline

In this section, you learn to join data coming from multiple sources. Most enterprises have data in multiple locations and need to combine it, either in order to store it in a unified format or to process it consistently.

This exercise takes about 45 minutes to complete. At the end, the Job looks like this:



Mapping data using a tMap component

The **tMap** component maps data from input to output, with the capability to perform a variety of transformations on the data. It transforms and routes data from single or multiple sources to single or multiple destinations. You will find yourself using it as the centerpiece of almost all your Talend Data Integration Jobs.

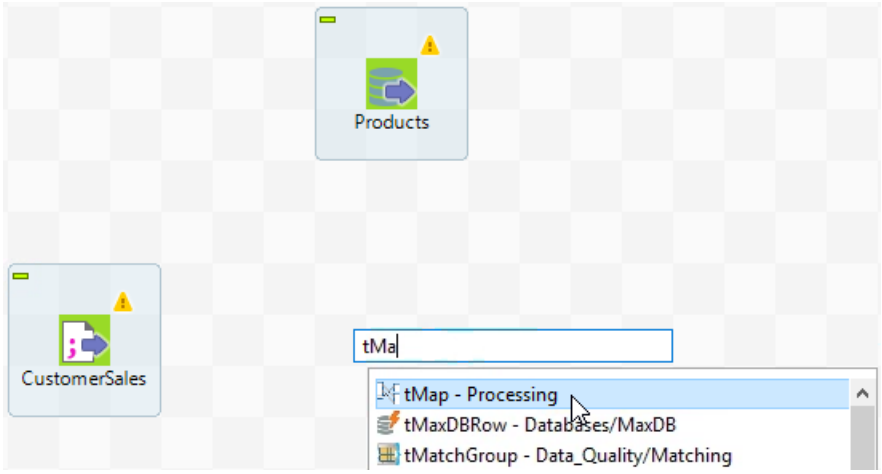
In this exercise, you join data using **tMap** and output results to generate reports. You can output data to several structured file formats, for example **tFileOutputDelimited** and **tFileOutputXML**.

1. ADD A tMap COMPONENT

   To add a tMap component:

a. Inside the **Designer**, click to the right of the **CustomerSales** component and below the **Products** component.

b. Start entering *tMap* to bring up components matching your search term.



c. On the list, double-click **tMap**. The component appears in the workspace.

2. CONNECT THE INPUT TO THE tMap COMPONENT

To connect the input to the **tMap** component:

a. Right-click in the middle of **CustomerSales** and select **Row > Main**. A connection link is visible between the input component and mouse pointer. You can now link to the output component.

b. Click the output component, **tMap**. The two components are connected.

c. Right-click the **Products** component and select **Row > Main**.

d. Click the output component (**tMap**) to link the components.

WARNING:
Notice that the second row you added automatically becomes a **Lookup** row. You can provide multiple input sources for a tMap component, but only one can be the **Main** row; the rest are **Lookup** rows.

Notice the warning displayed for the tMap component. The tMap component transforms data, so it needs an output destination.

3. NAME THE ROWS

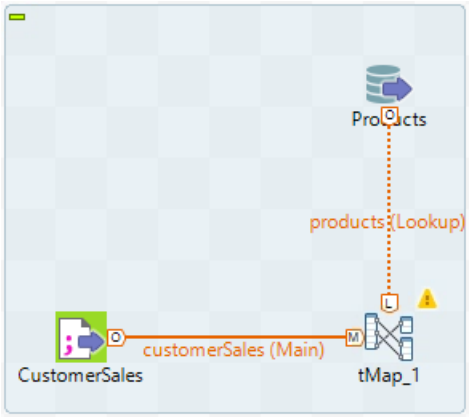
To name the rows between inputs and the **tMap** component:

a. Click **row1** (the link between **CustomerSales** and **tMap_1**) to select the row. When selected, the row name is highlighted in a frame.

b. Click once in the highlighted frame to activate editing and name the row *customerSales*.

c. Repeat the same process for the **row2** (the link between **Products** and **tMap_1**) and name it *products*

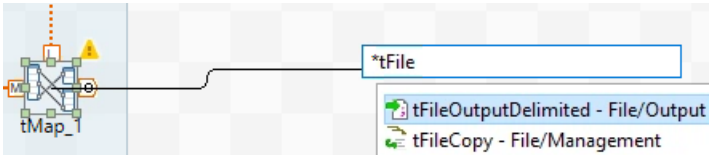
Your Job looks like this:



4. CREATE AN OUTPUT FILE

You want to create a file that is the sales report, containing the output you get from mapping the initial data. To write to a structured file, use a **tFileOutputDelimited** component. The **tFileOutputDelimited** component outputs the input data to a delimited file according to the defined schema.

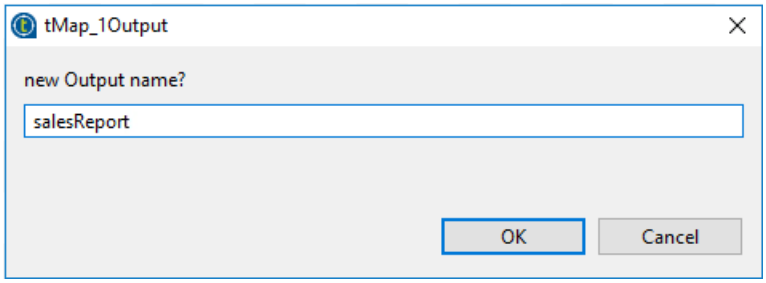
a. Right-click **tMap_1**. Holding the right-click, move the mouse pointer outside the component. When you release the mouse, a search box is displayed with a list of suggested components.



b. Start entering **tFileOutputDelimited** in the search box.

c. Double-click **tFileOutputDelimited** to add it to your Job. A dialog box opens, asking you to enter a name for the link between tMap and the output file

d. Name the link *salesReport* and click **OK**.

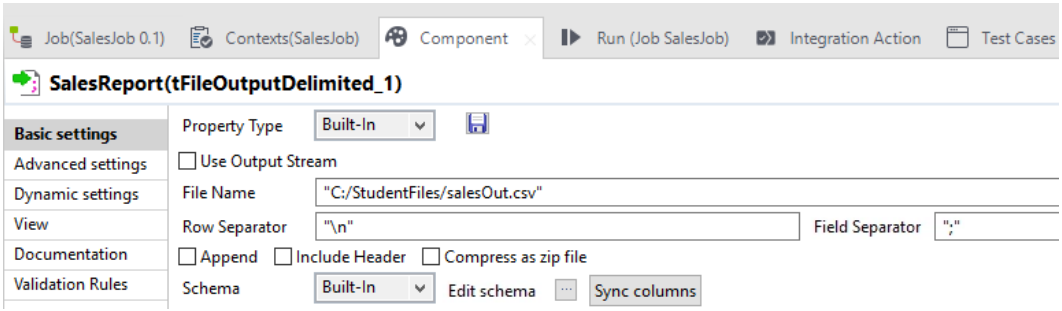


5. CONFIGURE THE OUTPUT FILE

To configure the output file:

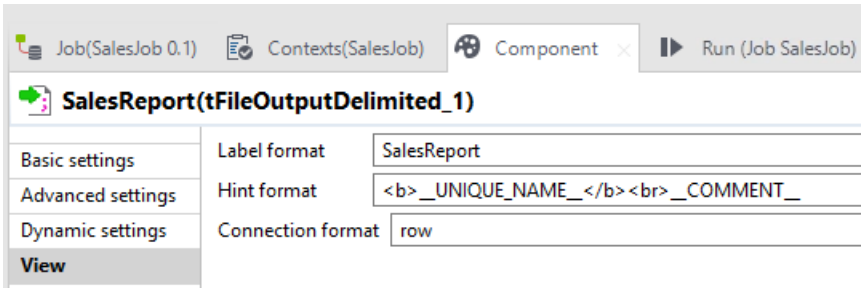
a. Double-click the **tFileOutputDelimited_1** component to open the **Component** view.

b. In the **File name** text box, enter *"C:/StudentFiles/salesOut.csv"*

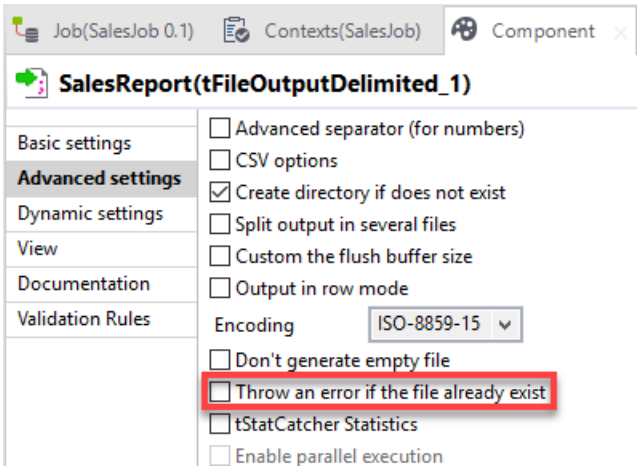


c. Leave the other settings as configured.

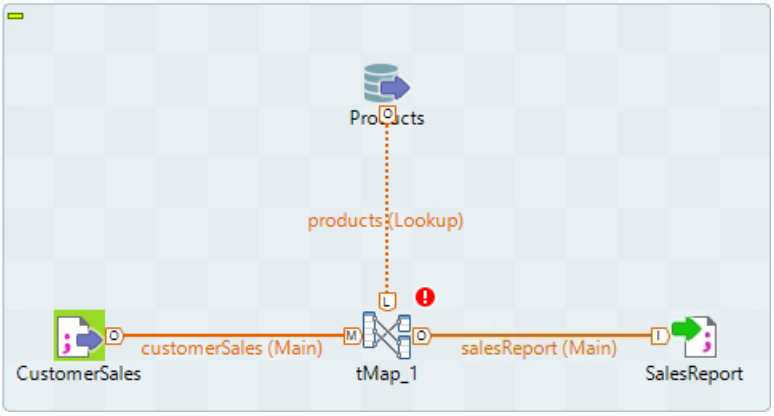
d. To name the component, on the **View** tab, replace the text in the **Label format** text box with *SalesReport*



e. By default, this type of component is configured to throw an error if the file already exists. Because you want to over-write the file each time you run the Job, on the **Advanced settings** tab, deselect the check box **Throw an error if the file already exist**.



When you are finished, the Job looks like this:



Notice that the upper right corner of **tMap_1** shows an error icon. This is due to the fact that you did not define the output you expect from the mapping. You do this by configuring the tMap component.

Configuring the tMap output table

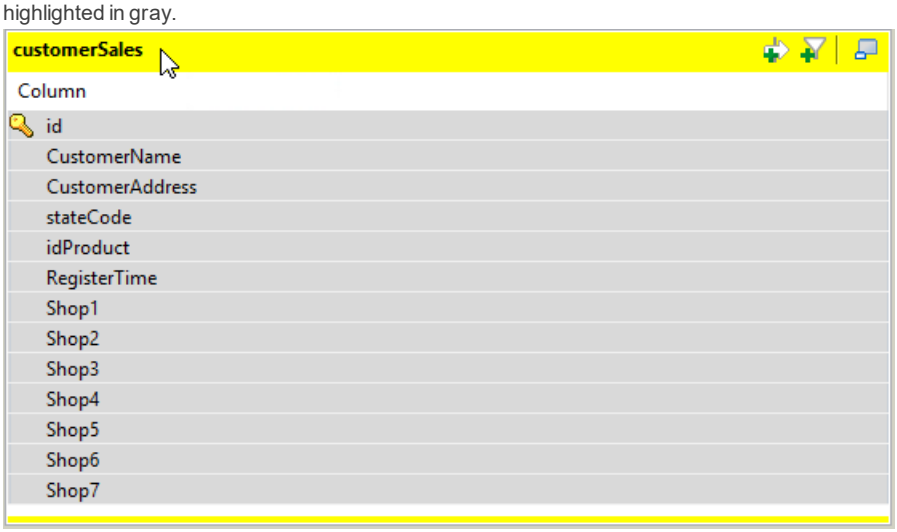
1. DEFINE THE OUTPUT TABLE

   Data from the CustomerSales file and Products table is transmitted through the links customerSales and products, as rows, to the **tMap** component. The goal of this step is to define which incoming data to pass to the output.

   Define the output you expect from the mapping:

   a. Double-click **tMap_1** to open the mapping editor. Notice the **customerSales** and **products** tables on the left (the input flows). You can recognize their schemas because they match the schemas of the input components. Notice on the right that a table has been created with the same name as the output link **salesReport**. You now need to define the schema for this output table.

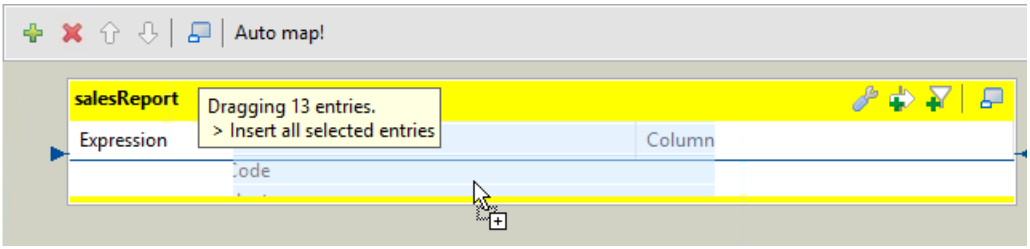
   b. Click on the **customerSales** toolbar at the top. The toolbar turns to yellow and the columns are all selected and highlighted in gray.



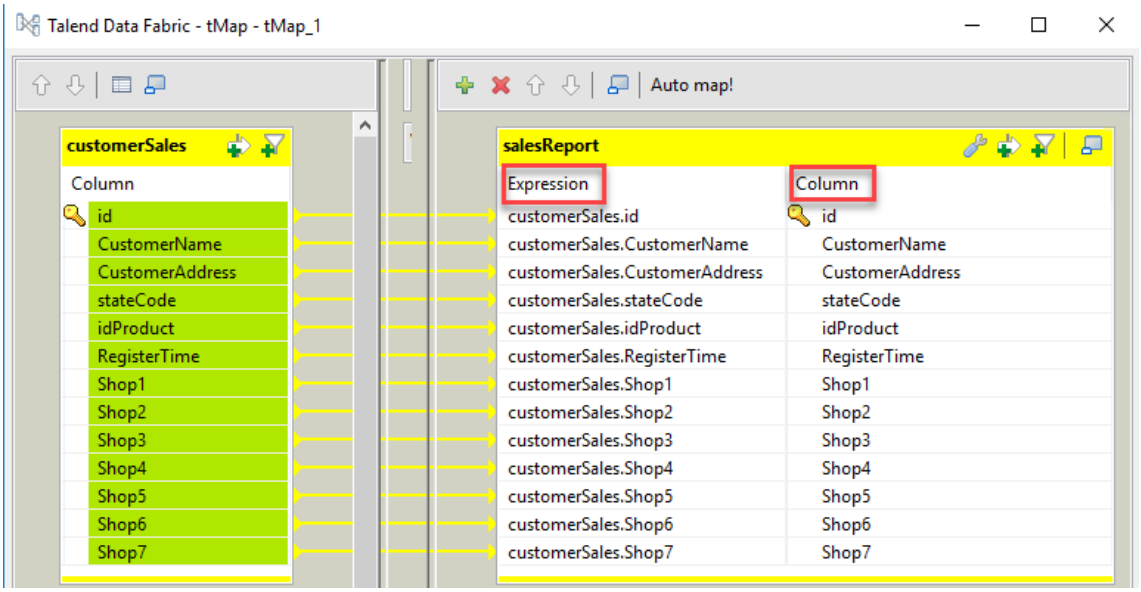
   You can drag and drop them to the output table but before doing so, carefully read the instructions in the next step.

   c. Click anywhere in the grayed area to grab the columns and drag them to the **salesReport** table. A blue line with a pointed arrow shows the exact position of the mouse pointer. When the arrow is placed in an appropriate pos-

ition,release the mouse.



When you are finished, the tMap looks like this:



Notice the arrows indicating the mapping of columns. In the **salesReport** table, notice the **Expression** that holds the actual data content, and the **Column** which contains the data labels. At this point, the data and column names are an exact match to those in the **customerSales** table.

2. CREATE EXPRESSIONS

You want to merge the address with the state code and create a single field that contains the full address of the customer. Because some of the state codes in the file are in lowercase, you want to set them all to uppercase. The expression you want to build is:

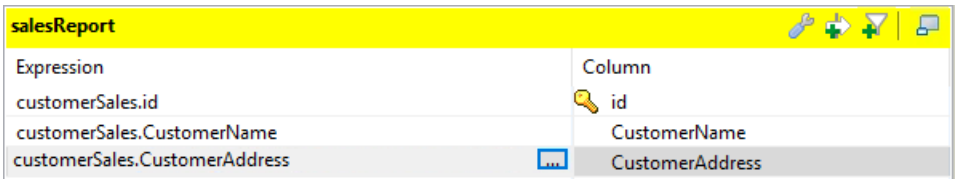
customerSales.CustomerAddress + " " + StringHandling.UPCASE("customerSales.stateCode")

Talend Studio allows you to create expressions directly in the expression fields, expression editor and Expression builder.

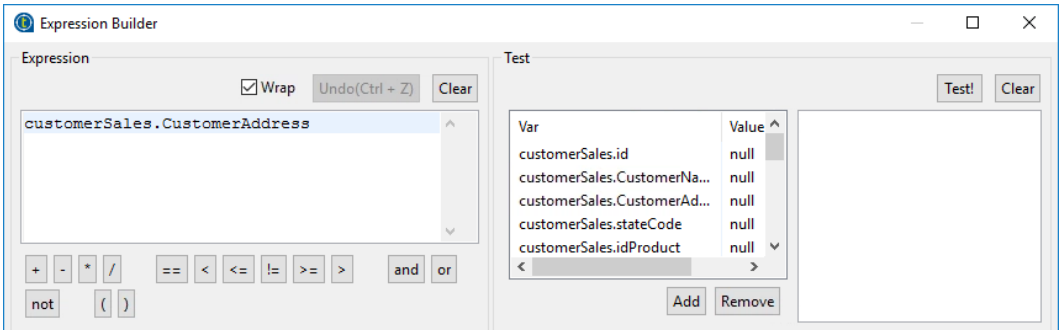
NOTE:
To easily create complex expressions, you use the **Expression builder**, which provides editing tools and a library of functions grouped by categories.

To create an expression on an output column:

a. In the salesReport table, click the output column **customerSales.CustomerAddress**. The **[...]** button appears on the right.



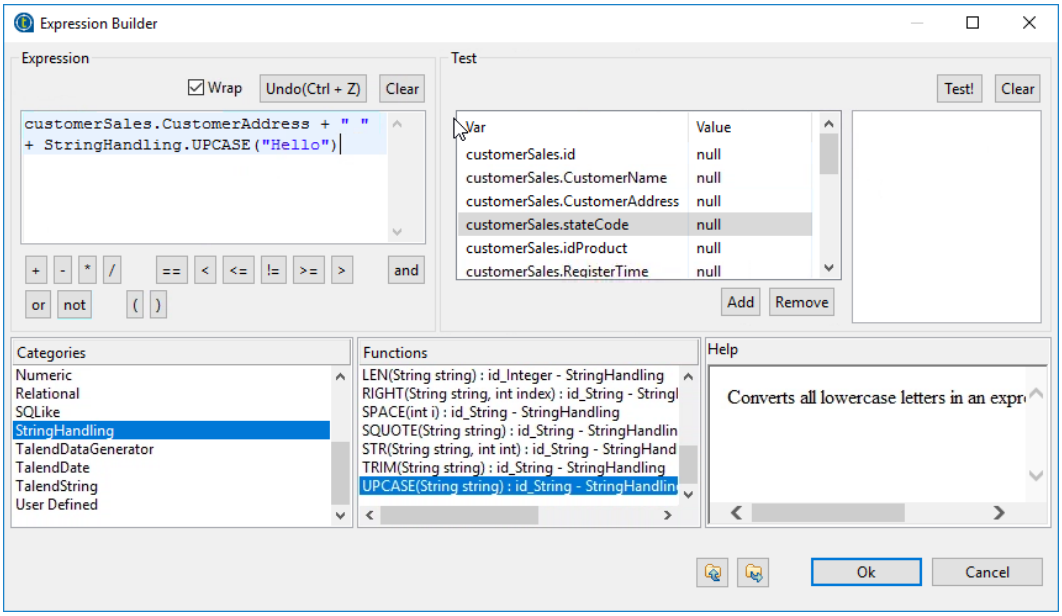
b. Click **[...]** to open the Expression builder. On the left, the expression field already contains the column you selected.



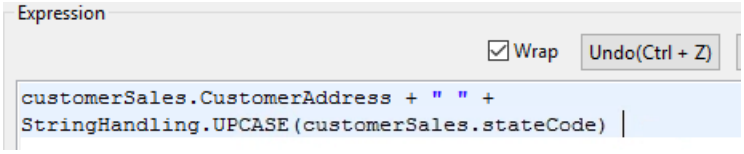
c. Edit the field to change the expression to *customerSales.Address* + " " +. You can now add a function that sets the state codes to uppercase.

d. In **Categories**, select **StringHandling**.

e. In **Functions**, scroll down and select **UPCASE**. Double-click **UPCASE** to add it to your expression. The expression looks like this:

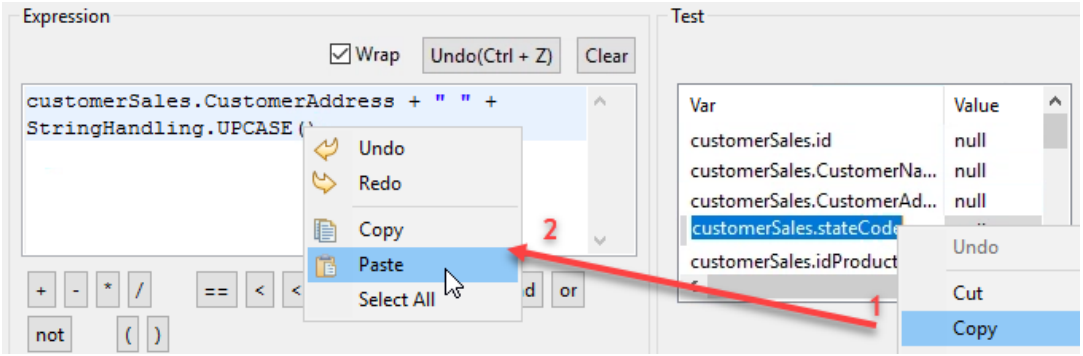


f. Delete the default "Hello" string and replace it with the variable *customerSales.stateCode*, which is of type String. When you are finished, the expression looks like this:



g. Click **OK** to close the Expression builder.

TIP:
To avoid typing mistakes while using variables in expressions, you can directly select a variable from the list, right-click to copy and then paste in the expression editing box in the right location.



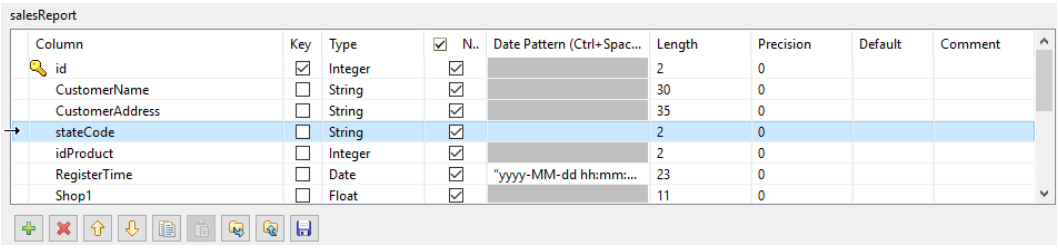
The **CustomerAddress** column in the **salesReport** table now contains both the address and the state code. The **stateCode** column is not useful anymore, you learn in the following how to delete it.

3. DELETE COLUMNS

To update the table schema (add, delete, move columns etc.), you can use the **Schema editor**, located at the bottom of the **tMap** editor. In the bottom right you find the **salesReport** schema.

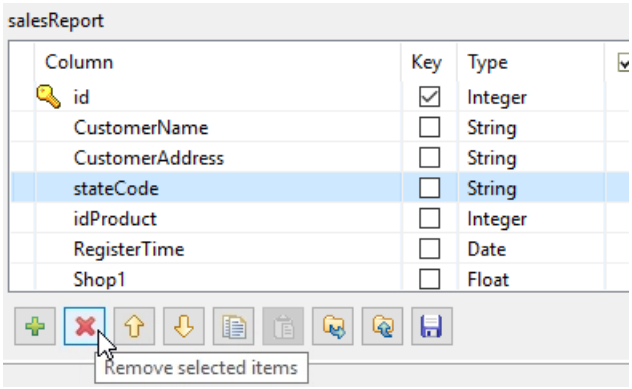
To delete the **stateCode** column from the **salesReport** schema:

a. Point the mouse to the left of the **stateCode** column. An arrow appears and the column is highlighted.



b. Click once to select the **stateCode** column. The column is highlighted in blue.

c. Click the **Remove selected items** button



WARNING: Make sure you click the **Remove selected items** icon in the schema editor of **salesReport**, not to be confused with the similar **Remove selected output table** icon in the upper right part of the map editor. **Remove selected output table** deletes the entire output table!

You have configured the output from the **customerSales** table. You now want to join data coming from the Products database and update the output you expect in salesReport accordingly.

Configuring a join model in tMap

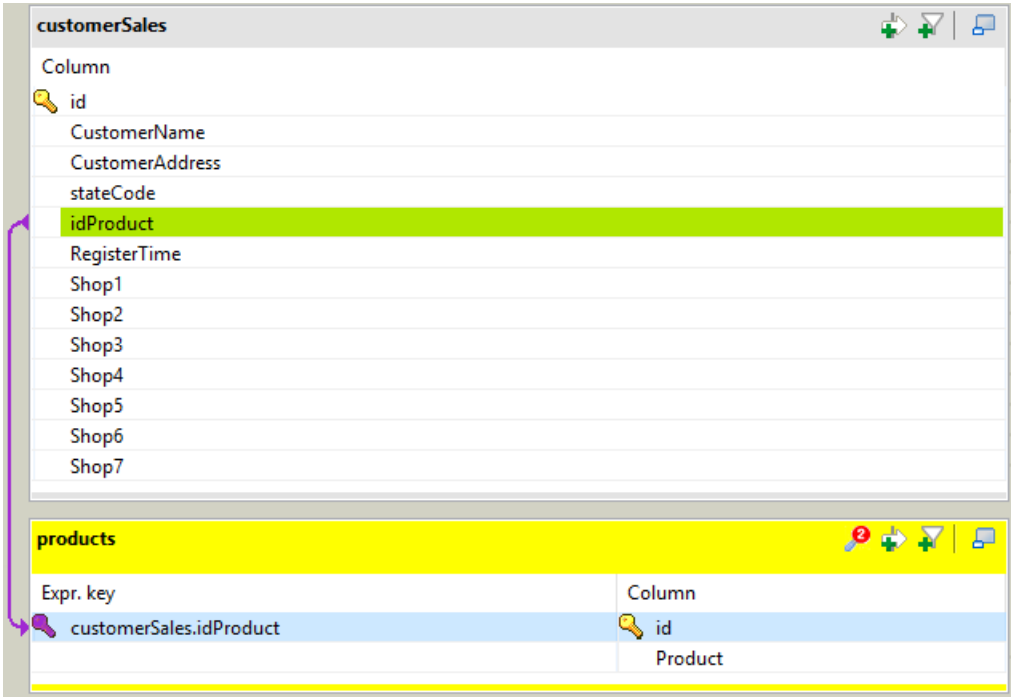
You can use **tMap** to create a join model that combines rows from the input tables, based on a related column between them. In this case, the related column is the product ID. As explained earlier, you are interested only in customers who bought products listed in the **Products** table, as they are the target of the marketing campaign. Therefore, you want to use an inner join that returns only records existing in both input tables.

1. CREATE A JOIN

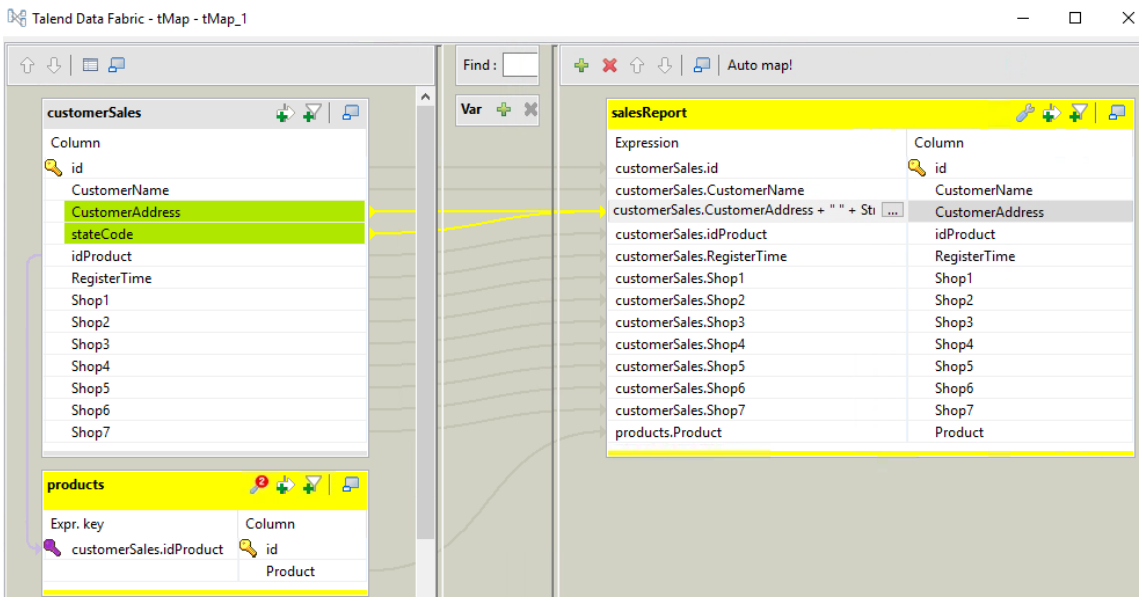
   To create a join:

   a. Select the **idProduct** column in **customerSales**.

   b. Drag and drop it on the left side of the **id** column in the **products** table. This associates the **id** columns from the two tables.



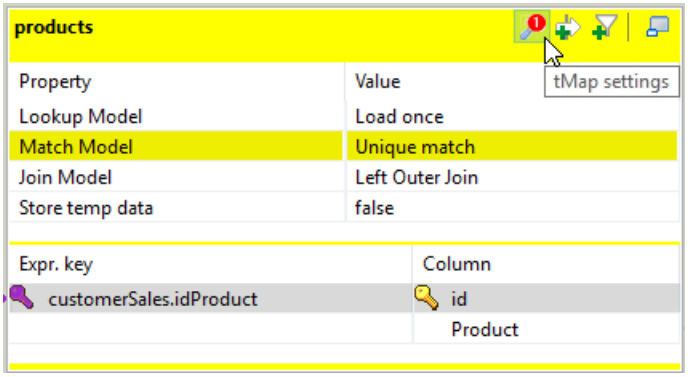
   c. Drag the **Product** column from the **products** table and drop it in the **salesReport** table.



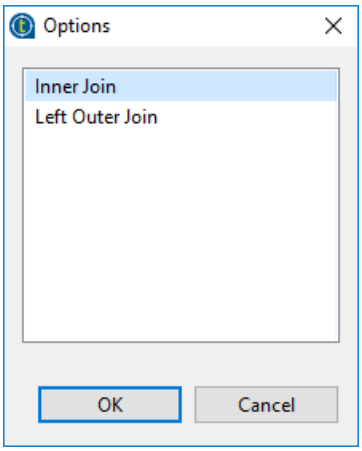
2. CONFIGURE THE JOIN MODEL

   To create a join:

   a. On the **products** table toolbar, click the **tMap settings** icon. Notice that the default **Join Model** is Left Outer Join.



   b. Click **Left Outer Join**. Click the **[...]** button that appears in the right and in the dialog box that opens, select **Inner Join** and click **OK**.



c. Make sure the **Join Model** is set to **Inner Join**.

d. In the tMap editor window, click **OK** to save the changes.

e. When asked if you want to propagate the changes, click **Yes**.

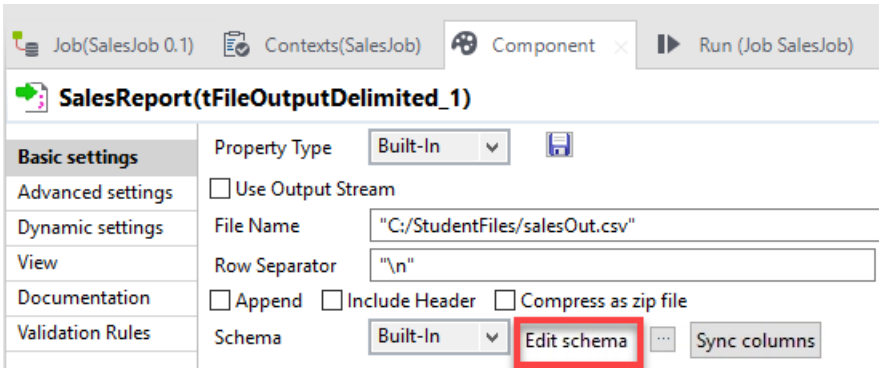
When you make changes to the schema of the output connection, you are prompted to propagate the changes (in other words, reflect those changes in the schema of the component on the other end of the connection). In the following you confirm that the schema was actually updated.

3. CHECK THE OUTPUT FILE SCHEMA

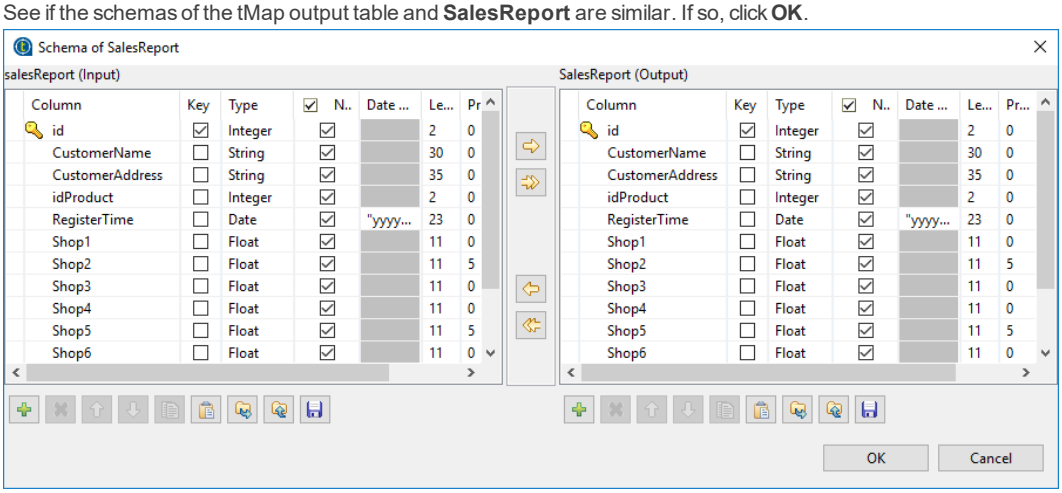
To check the schema:

a. Double-click the **SalesReport** component to open the **Component** view.

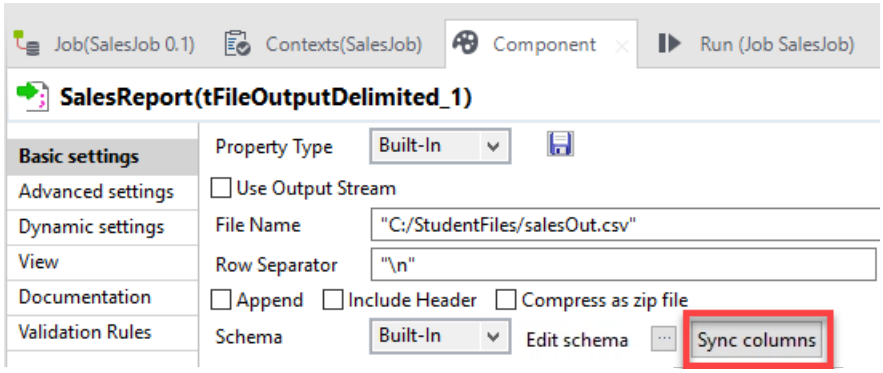
b. To the right of **Edit schema**, click **[...]**.



c. See if the schemas of the tMap output table and **SalesReport** are similar. If so, click **OK**.



NOTE:
The **Sync columns** button in the **Component** view allows you to synchronize two schemas in one click whenever necessary.

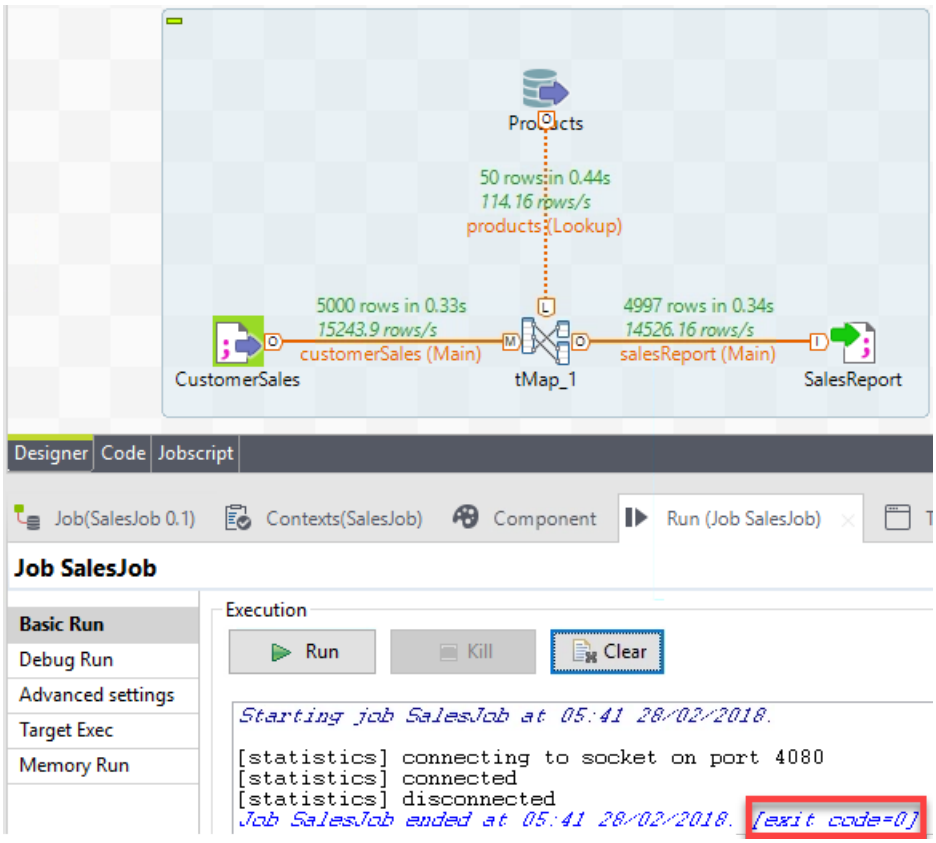


4. RUN THE JOB

To run the Job:

a. In the **Run** view, click **Run**

b. Make sure the Job executes correctly (**[exit code=0]**).



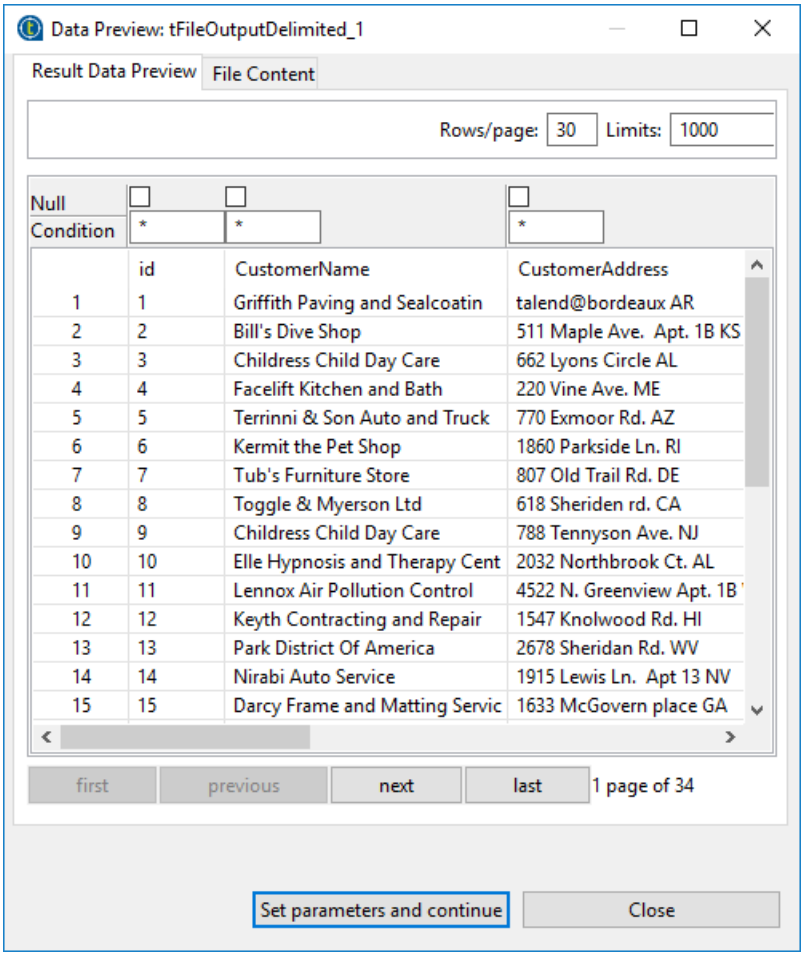
Notice that after execution, the number of processed rows transmitted from one component to another is displayed on the links. You can see that only 4997 of 5000 rows received at the tMap input were output in SalesReport. Some of the rows were rejected by the inner join.

5. VISUALIZE THE REPORT USING THE DATA VIEWER

   To visualize data using the **Data viewer**:

a. Right-click the **SalesReport** component.

b. Select **Data viewer**. A window displays the results of the mapping.



c. Look at the data and click **Close**.

NOTE:
Alternatively, you can use a text editor like Notepad++ to visualize the resulting file located at C:/StudentFiles/salesOut.csv.

Capturing the join rejects

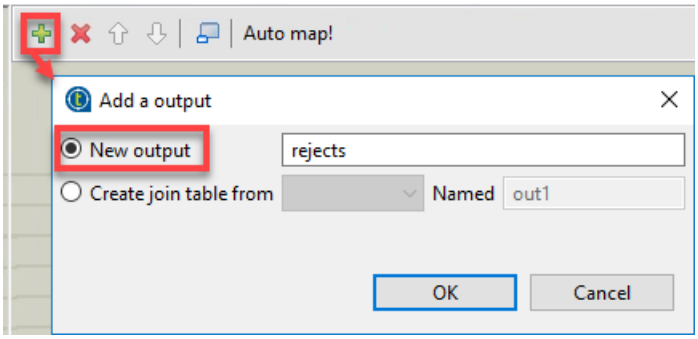
In the following section, you investigate which rows are rejected to make sure you did not exclude relevant data.

1. ADD AN OUTPUT TABLE TO CAPTURE JOIN REJECTS

   To capture join rejects, create a second output table in the tMap configuration:

   a. Double-click **tMap_1** to open the mapping editor. Remember that the output tables are listed to the right of the editor window

b. On the main toolbar on the right, click the **Add output table** button



c. In the **New output** text box, enter the table name *rejects* and click **OK**. The rejects table is created.

2. CONFIGURE THE REJECTS TABLE

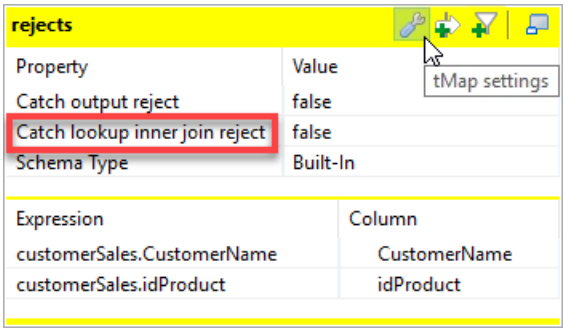
To configure the **rejects** table, you must define the table rows and set the **Catch lookup inner join rejects** property to *true*. You want the rejects table to contain the customer name and the product id columns. To define the table rows, you repeat the same process you used when creating the first output table.

Follow these steps:

a. In the **customerSales** table, select **CustomerName** and drag and drop it in the **rejects** table as soon as the blue arrow is positioned correctly.

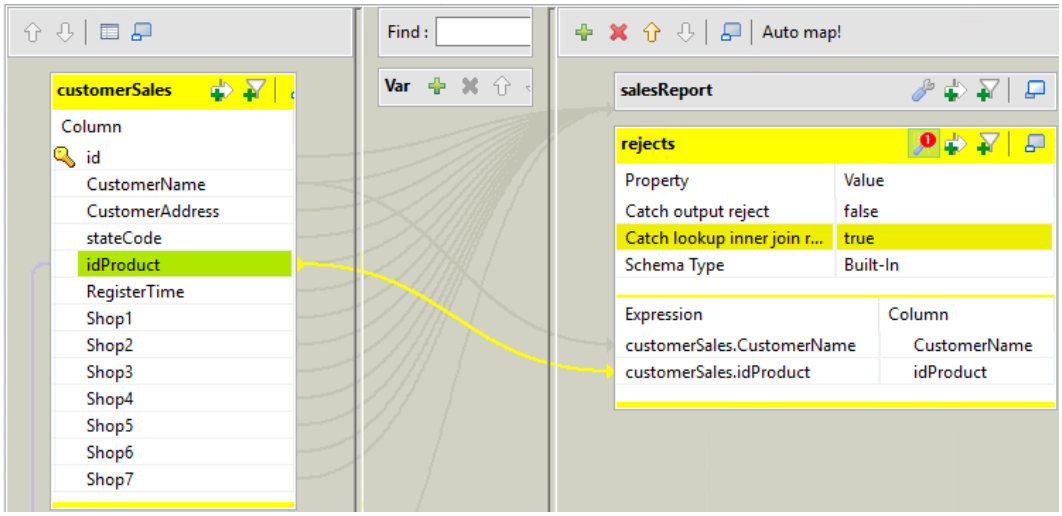
b. In the **customerSales** table, drag **idProduct** and drop it in the **rejects** table

c. On the **rejects** table toolbar, click the **tMap settings** icon. Notice that the **Catch lookup inner join rejects** is set to **false**.



d. To the right of **Catch lookup inner join reject**, click **false**. The **[...]** button appears to the right.

e. Click on **[...]** and in the dialog that opens, select **true** and click **OK**. This allows you to catch the unmatched rows in the rejects table. Your mapping looks like this:



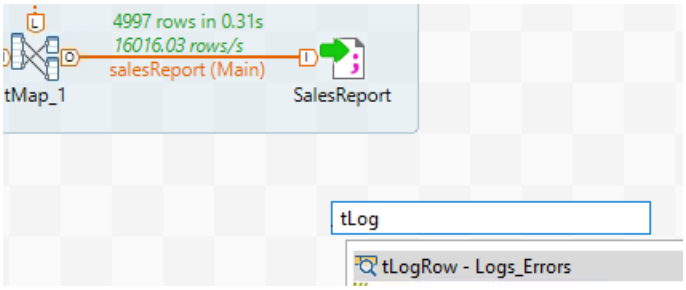
f. In the tMap editor window, click **OK**.

You have configured the rejects table. You are ready to send the output to the console.

3. LOG REJECTED ROWS TO THE CONSOLE

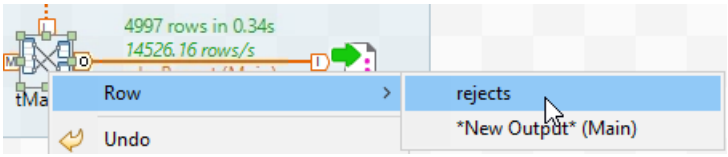
To log failures to the console, use a **tLogRow** component:

a. In the **Designer**, under the **SalesReport** component, start entering **tLogRow** to bring up the components matching your search.

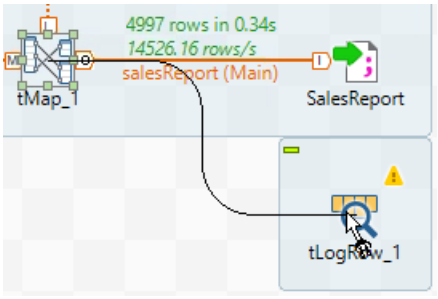


b. Double-click **tLogRow** on the list. The component appears in the workspace.

c. Right-click the **tMap_1** component and select **Row > rejects**.

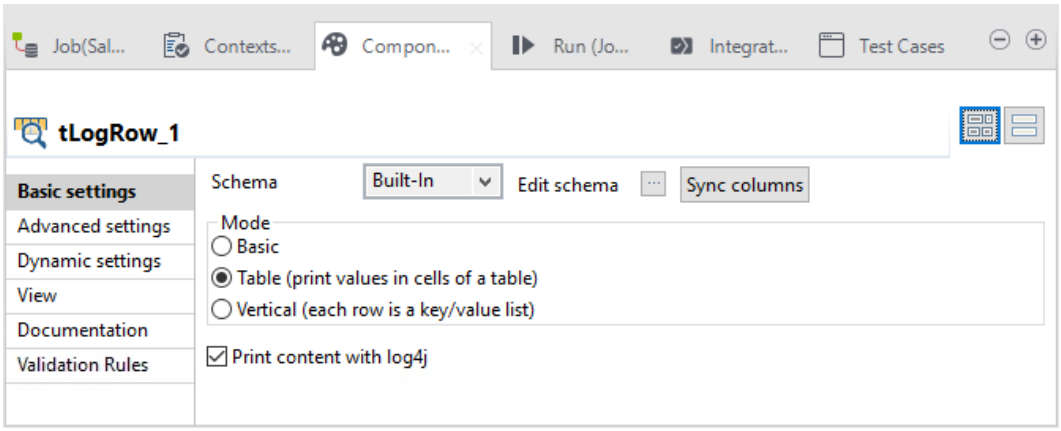


d. A connection link is visible between tMap and the mouse pointer. You can now drag the mouse pointer over to the output component.

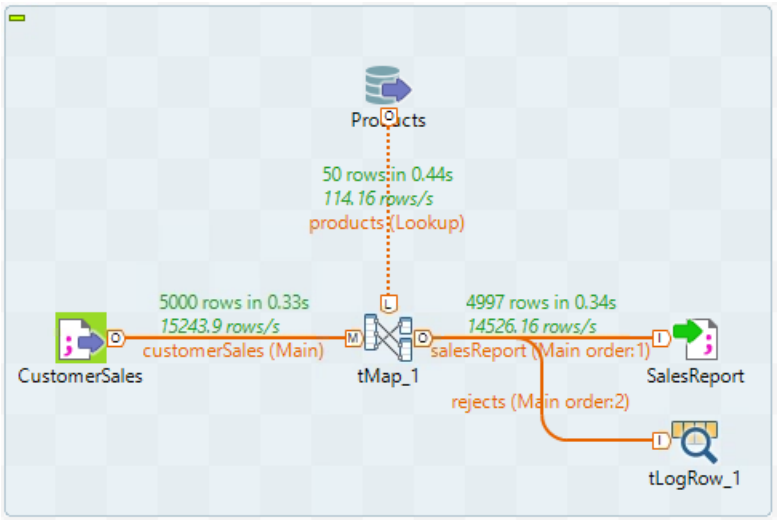


e. Click **tLogRow_1**. Notice that a **rejects** row is created between the two components.

f. Double-click **tLogRow_1** to open the **Component** view and select **Table**. This allows you to print results in a user friendly format.



Your Job looks like this:



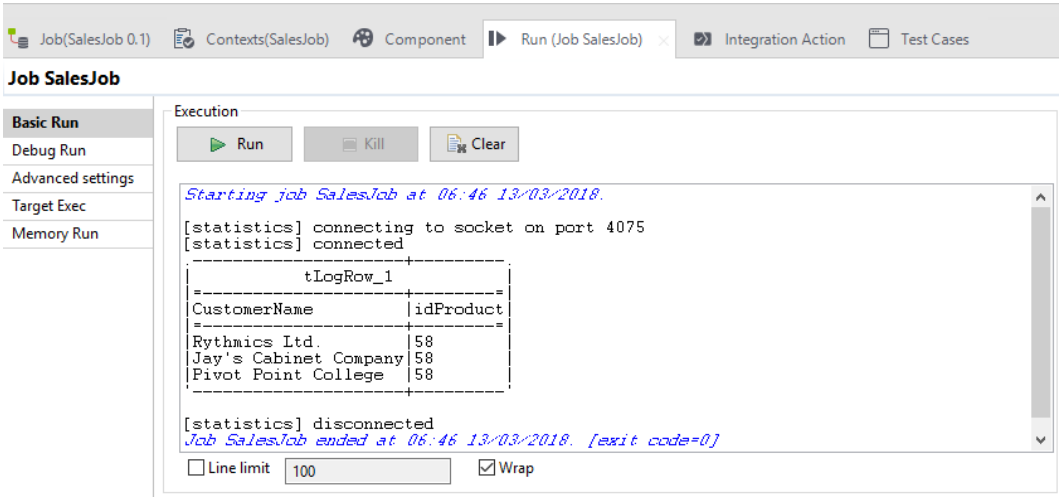
You can now run the Job and see the results.

4. RUN THE JOB

To run the Job:

a. On the **Run** tab, click **Run**

b. Make sure the Job executed correctly (**[exit code=0]**)



Notice that three rows were sent to the log component and displayed in the console. For each row, the name of the customer and product ID are shown. As in the Products database, you have no product with identifier 58, so you can now understand why these rows were rejected.

Next step

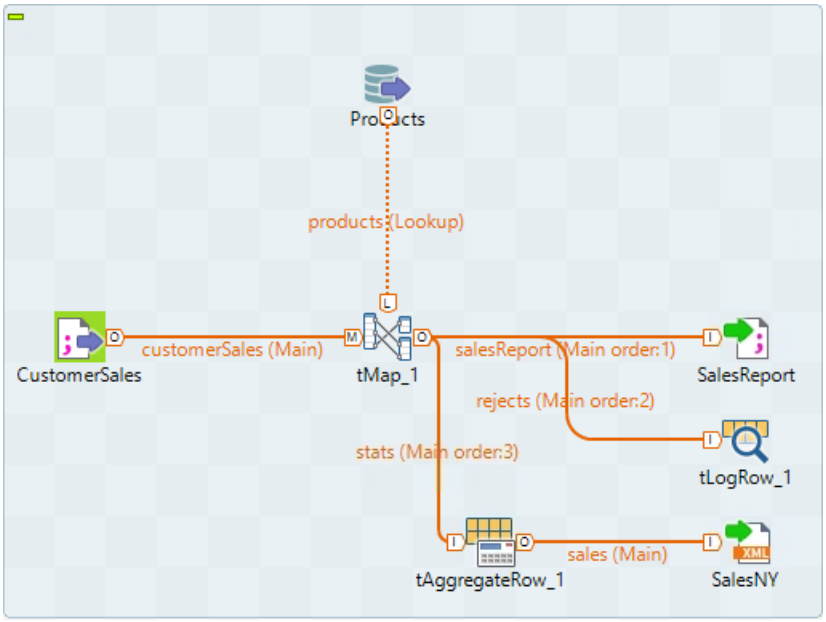
You have joined data using the tMap component and generated a report containing the join results. You have also investigated the rejects to make sure you do not omit relevant data. You are ready to learn more about transforming data.

Transforming Data

Task outline

In this exercise you learn to aggregate data, apply filters, and compute some statistics for the New York (NY) state in order to identify the best-selling products. At the end, you want to generate a report in a format that can be further processed by another system.

This section takes about 35 minutes to complete. At the end, the Job looks like this:



Aggregating and filtering data

You want to compute some statistics for the New York (NY) state in order to identify the best-selling products. To achieve this, you must create a new output table (*stats*) that contains the product name and customer ID coming from the input tables. However, you need to create a filter on the state so that only products sold in New York are considered. Then you use the **tAggregateRow** component to group data by product name and count the number of customer sales per product. You want to print the results in an XML file that can be transferred to another system for processing.

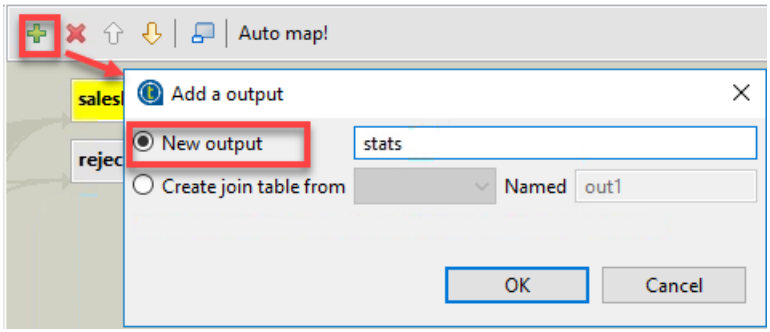
NOTE:
tAggregateRow: Receives a flow and aggregates it based on one or more columns. For each output line, are provided the aggregation key and the operations to compute (e.g. min, max, sum).

Before starting the exercise, double-click **tMap_1** to open the mapping editor. On the **salesReport** table toolbar, on the right, click the **Minimize** button to close the table editor. Repeat this step for the **rejects** table. Now you can focus on creating the new output table.

1. ADD THE STATS TABLE IN tMap

   To create the stats table in the mapping editor:

a. On the toolbar situated on the right, click the **Add output table** button.



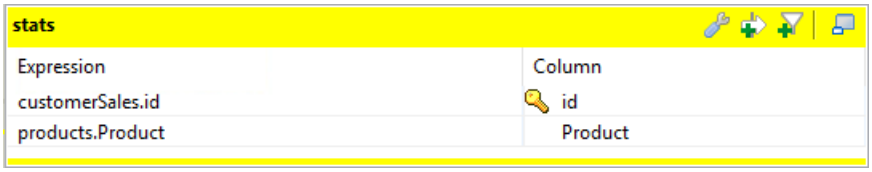
b. In the **New output** text box, enter the table name *stats* and click **OK**. The stats table is created.

2. CONFIGURE THE STATS TABLE

To configure the **stats** table, you must define the table rows:

a. In the **customerSales** table, select **id**. Drag it to the **stats** table and drop it as soon as the blue arrow is positioned correctly.

b. In the **products** table, select **Product** and drop it to the **stats** table.



You are ready to create a filter on the state column so that you only consider products sold in New York.

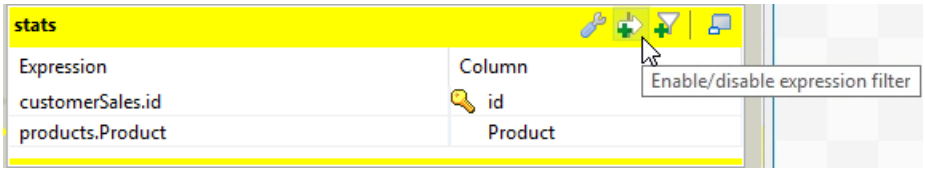
3. APPLY A FILTER USING THE FILTER EXPRESSION IN tMap

You want to create the filter expression:

"ny".equals(customerSales.stateCode)

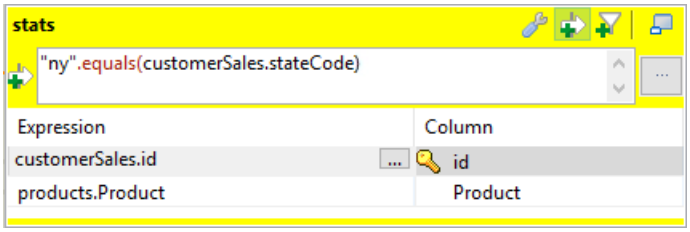
To apply filters to data:

a. On the stats table toolbar, click the **Enable/disable expression filter** button. The in-line editor opens.



b. In the **customerSales** table, select and drag the **stateCode** column into the in-line editor.

c. In the in-line editor, complete the expression *"ny".equals(customerSales.stateCode)*. Alternatively, you can click **[...]** on the right to open the Expression builder dialog box that you used earlier.



d. In the tMap editor window, click **OK**.

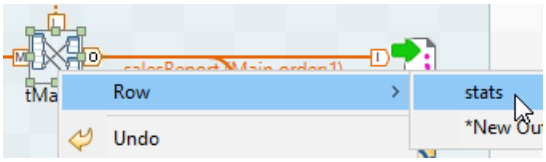
4. ADD A tAggregateRow COMPONENT

To add a **tAggregateRow** component:

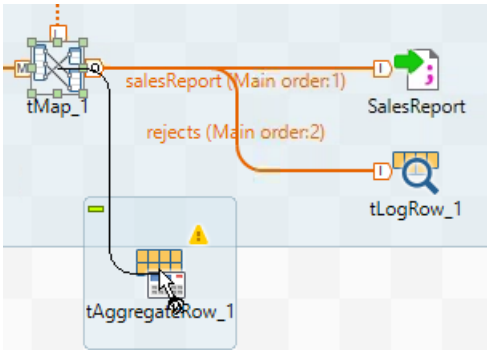
a. In the **Designer**, under the tMap_1 component, start entering **tAggregateRow** to bring up components matching your search.

b. Double-click **tAggregateRow** on the list. The component appears in the workspace.

c. Right-click the **tMap_1** component and select **Row > stats**.



d. A connection link is visible between tMap and the mouse pointer. You can now drag the mouse pointer over to the output component.



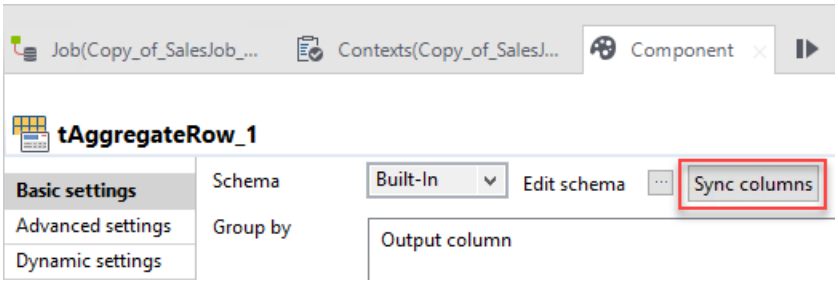
e. Click **tAggregateRow_1**. Notice that a **stats** link is created between the two components.

5. CONFIGURE THE tAggregateRow SCHEMA

To configure the **tAggregateRow** schema:

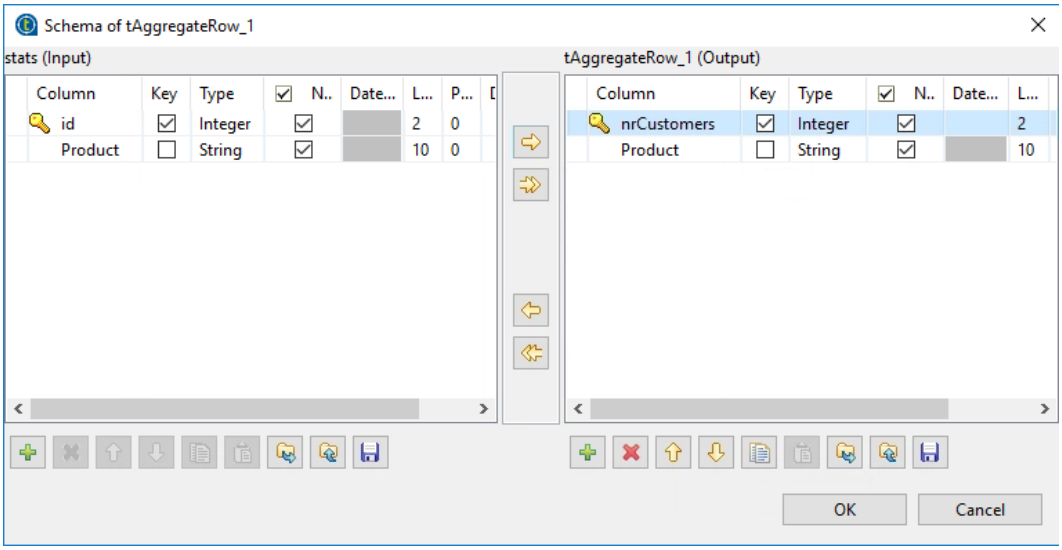
a. Double-click **tAggregateRow_1** to open the **Component** view.

b. Click the **Sync columns** button. Remember that this button allows you to synchronize the schema of your component with the schema of the input link (the stats link).



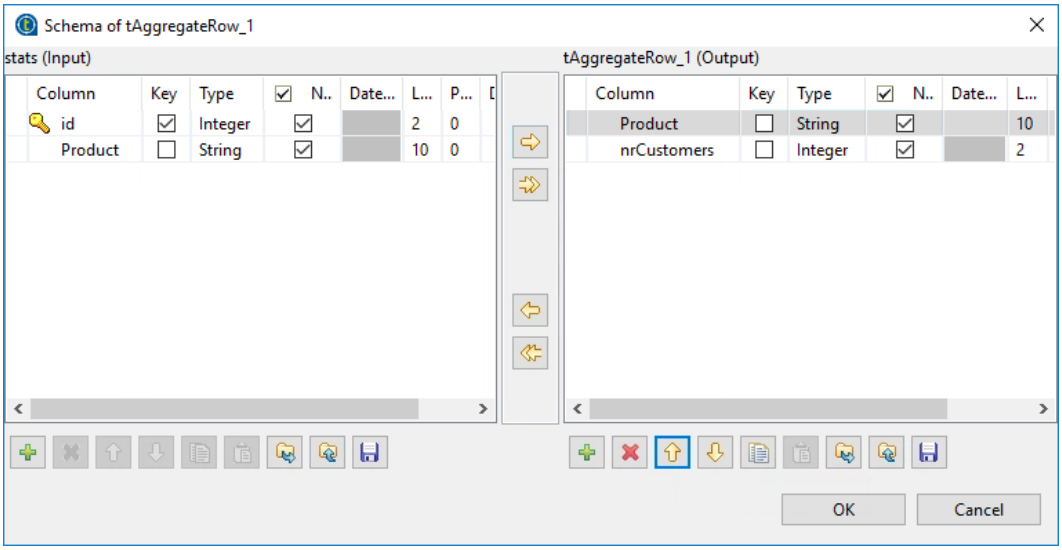
c. To the right of **Edit schema**, click **[...]** and notice that the two schemas are exactly the same.

d. In the Output schema **tAggregateRow_1**, click *id*. Rename the column *nrCustomers*. This gives you a more user friendly display in the file report you create.



e. To the right of the **nrCustomers** column, deselect **Key**. You don't need this column to be a primary key.

f. In the Output schema **tAggregateRow_1**, select the **Product** column. In the toolbar at the bottom, click the **Move up selected items** button, to place the Product column on top. This gives you a more user friendly display in the file report.



g. Click **OK** to close the window.

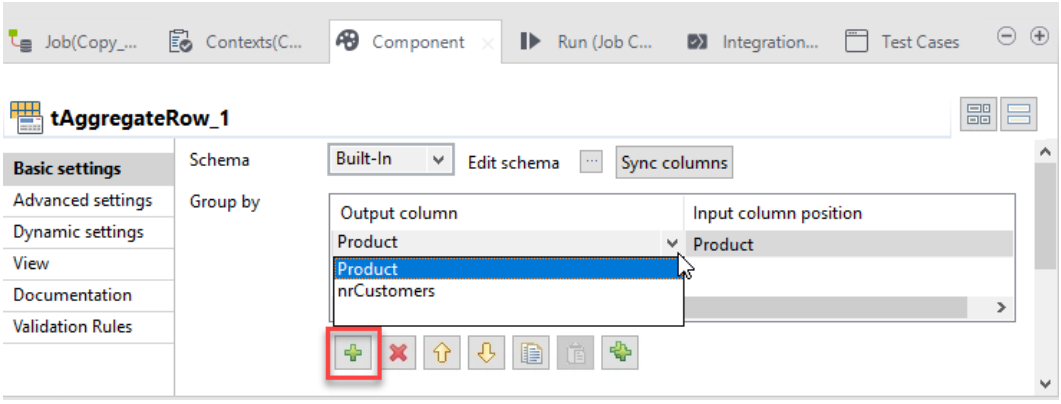
6. CREATE A GROUP BY CLAUSE WITH AGGREGATE FUNCTIONS

To create a GROUP BY clause with aggregate functions, you configure the Output column and the Input column position in the Group by and Operations tables.

Output column: allows you to select columns from the output schema

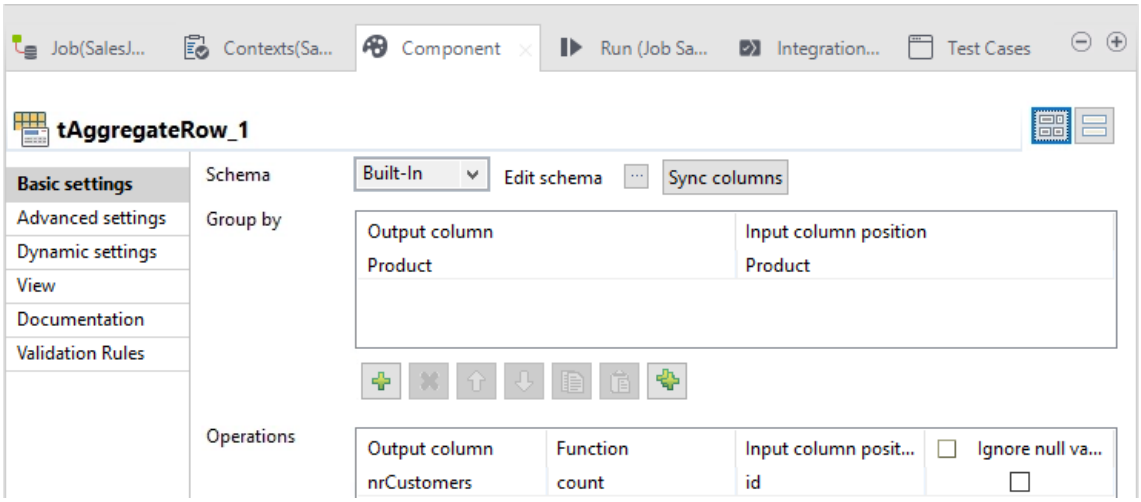
Input column position: allows you to select columns from the input schema

a. Below the **Group by** section, click the **[+]** button. Notice that the Output column and the Input column position both display the **Product** column by default. If this is not the case, just click on the column to select Product, as shown in the screenshot.



b. For **Operations**, click the **[+]** button. Make sure the **Output column** is set to *nrCustomers* and the **Input column position** to *id*

When you are finished, the configuration looks like this:



Generating the NY sales report

You want to print the results of the aggregation in an XML file that can be transferred to another system for processing. For this purpose you can use the **tFileOutputXML** component.

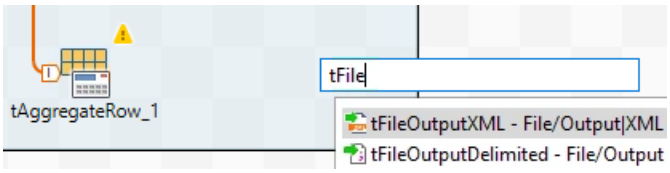
NOTE:
tFileOutputXML: Writes an XML file with separated data values according to a defined schema.

1. ADD AN XML FILE TO WRITE THE RESULTS

   To write the results to an XML file, your report, you can use a **tFileOutputXML** component.

   To add a **tFileOutputXML** component:

   a. In the **Designer**, to the right of the **tAggregateRow_1** component, start entering **tFileOutputXML** to bring up components matching your search.

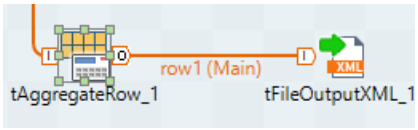


b. Double-click **tFileOutputXML** on the list. The component appears in the workspace.

c. Right-click the **tAggregateRow_1** component and select **Row > Main**.

d. A connection link is visible between tMap and the mouse pointer. You can now drag the mouse pointer over to the output component.

e. Click **tFileOutputXML_1**. Notice that a link is created between the two components.



f. Click on **row1** to edit and rename the link sales

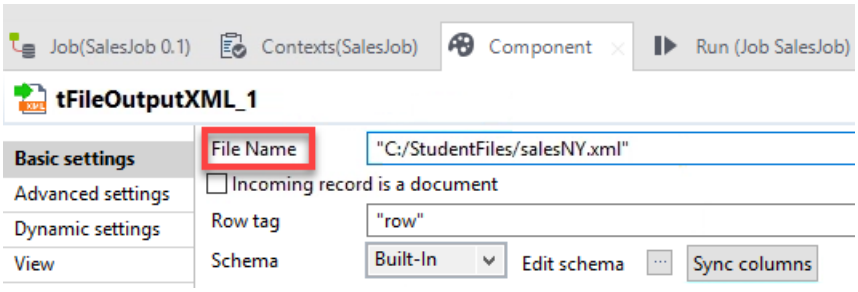
2. CONFIGURE THE OUTPUT COMPONENT

To configure the **tFileOutputXML_1** component:

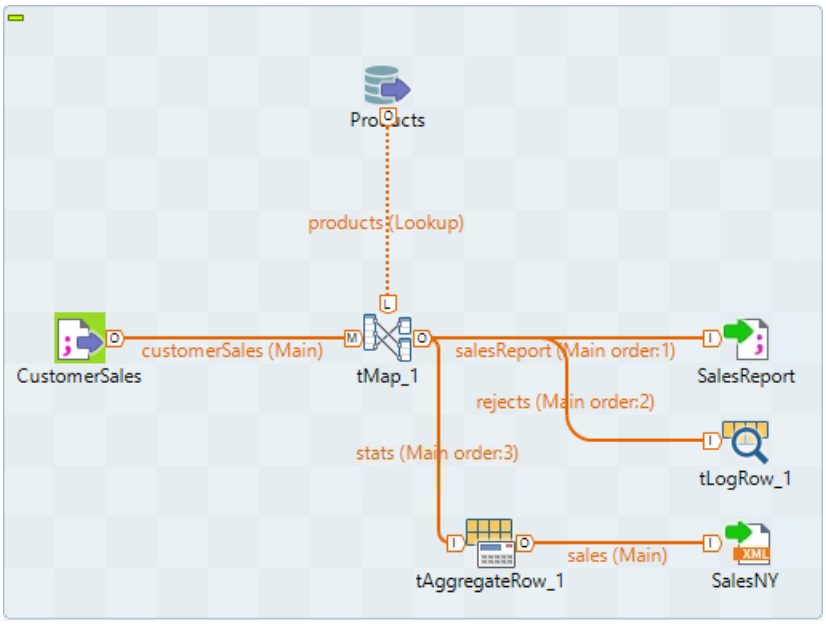
a. Double-click **tFileOutputXML_1** to open the **Component** view.

b. In the **File Name** text box, enter *"C:/StudentFiles/salesNY.xml"*

c. On the **View** tab, in the **Label format** text box, enter *SalesNY* to rename the file.



When you finish, your Job looks like this:



You can now run the job and generate the SalesNY. xml report.

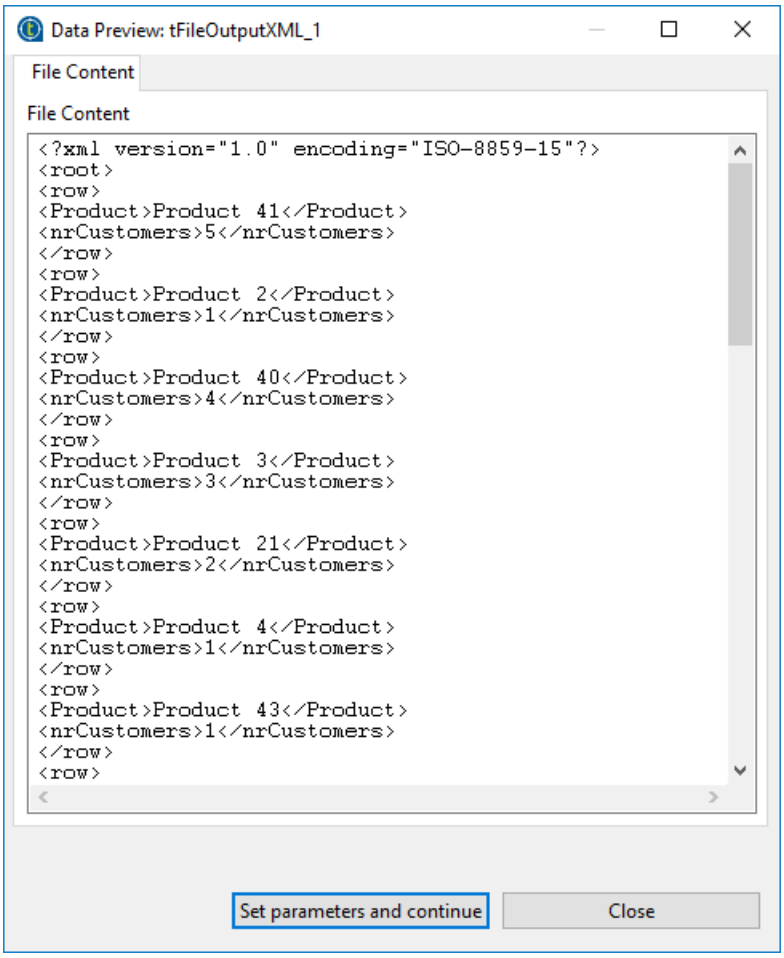
3. RUN THE JOB

   To run the Job:

   a. On the **Run** tab, click **Run**

   b. Make sure the Job executes correctly (**[exit code=0]**).

   To check the results, right-click the **SalesNY**component and select **Data viewer**.



Congratulations! You have designed and run your first data integration Job using Talend Studio.

Next step

You have almost finished this section. Time for a quick review.

Review

In this lesson, you have learned how to design and run your first data integration project using Talend Studio.

You followed the classical phases of a data integration chain. You extracted data from structured files using **tFileInputDelimited** and from databases using **tMysqlInput**. You explored the functionnalities offered by the **tMap** component to join and process data. You applied different transformations on the data such as filters and common aggregation techniques using the **tAggregateRow** component. You learned how to construct expressions using the inline editor and the Expression builder. Then you loaded the result in the appropriate target storage.

Intentionally blank